Computational methodology for enhanced sensitivity analysis of gene regulatory networks

A dissertation presented

by

Mattia Petroni

to

The Faculty of Computer and Information Science

in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in the subject of

Computer and Information Science

Ljubljana, 2018

The thesis was supported by the national postgraduate programme Higher Education National Scheme (*Inovativna shema za sofinanciranje doktorskega šudija za spodbujanje sodelovanja z gospodarstvom in reševanja aktualnih družbenih izzivov — generacija 2011 Univerza v Ljubljani*), financed by the European Union (EU), University of Ljubljana and Slovenian Ministry of Higher Education, Science and Technology.

APPROVAL

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

— Mattia Petroni —

April 2018

The submission has been approved by

dr. Miha Moškon
Assistant Professor of Computer and Information Science
ADVISOR

dr. Roman Jerala
Professor of Molecular Biology and Biochemistry
EXTERNAL EXAMINER

dr. Miha Mraz
Professor of Computer and Information Science
EXAMINER

dr. Bojan Orel
Professor of Mathematics
EXAMINER

dr. Nikolaj Zimic
Professor of Computer and Information Science
EXAMINER

PREVIOUS PUBLICATION

I hereby declare that the research reported herein was previously published/submitted for publication in peer reviewed journals or publicly presented at the following occasions:

[1] M. Petroni, N. Zimic, M. Mraz and M. Moškon. Stochastic simulation algorithm for gene regulatory networks with multiple binding sites. *Journal of Computational Biology*, volume 22, number 3, pages 218–226, 2015, doi: 10.1089/cmb.2014.0064, Mary Ann Liebert Inc.

[2] M. Moškon, J. Bordon, M. Mraz, N. Zimic and M. Petroni. Computational approaches in synthetic and systems biology. Chapter in *Recent Advances in Systems Biology Research*, Pages 131–156, Editors: André X. C. N. Valente, Abhijit Sarkar and Yuan Gao. 2014, ISBN: 978-1-62948-736-6, Nova Science Publishers.

[3] M. Petroni, M. Moškon. A nested stochastic simulation algorithm for the gene regulatory network of the Epstein-Barr virus. Abstract at *14th Workshop on Algorithms in Bioinformatics*, September 9, 2014, Wroclaw, Poland.

[4] M. Petroni, N. Zimic, M. Mraz and M. Moškon. Multiscale stochastic simulation algorithm for complex gene regulatory networks. Abstract in 9th CFGBC Symposium From arrays and sequencing to understanding diseases: book of abstracts, Pages 46. Ljubljana, 2014

[5] M. Petroni, N. Zimic, M. Mraz and M. Moškon. A parallel algorithm for stochastic multiscale simulation of gene regulatory networks with multiple binding sites. Book of abstracts at 10th CFGBC Symposium, pages 30. Ljubljana, June 30 - July 3, 2015

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Ljubljana.

University *of Ljubljana*
Faculty *of Computer and Information Science*
Mattia Petroni
Computational methodology for enhanced sensitivity analysis of gene regulatory networks

ABSTRACT

Biological computing is held towards a new era of processing platforms based on the bio-logical computer structures that are at the heart of biological systems with information processing capabilities. These bio-logical computer structures are mostly based on gene regulatory networks, mainly because their dynamics reminds the computer logic structures functioning. The use of these bio-structures is still in its early days since they are for the time being far less effective than their silicon counterparts. However, their use can be already exploited for a wide range of applications, covering pharmacological, medical and industrial. In order to develop such applications, a precise design that is based on computational modelling is vital in the process of their implementation.

Gene regulatory networks can be described as a chemical reacting systems. The dynamics of such systems is defined at the molecular level with a set of interacting reactions. The stochastic simulation algorithm can be used to generate the time evolution trajectories of each chemical species by firing each reaction according to a Monte-Carlo experiment. The main shortcoming of this approach is its computational complexity, which increases linearly with the total number of reactions that have to be simulated. When the number of reactions becomes too high, the stochastic simulation algorithm turns out to be impracticable. This is the case of certain gene regulatory networks, which can be either found in nature or can be artificially constructed. An additional problem lies in the fact that reactions in such networks can often occur at different time scales, which can differ by many orders of magnitude. Such scenario occurs when gene regulatory networks contain multiple cis-regulatory binding sites, on which different transcription factors are able to bind non-cooperatively. The transcription factors binding occurs much faster than the average reactions in the gene expression, therefore, this time-scale gap needs to be accounted into the simulation. Moreover, the transcription control can be affected by specific dispositions of the bound transcription factors,

which is only possible to simulate, if all the reactions that can produce the same dispositions are defined. The number of such reactions increases exponentially with the number of binding sites.

In order to decrease the time complexity of the stochastic simulation algorithm for such gene regulatory networks, an alternative algorithm called the dynamic multi-scale stochastic algorithm (DMSSA) is proposed, in which the reactions involved in the transcription regulation can be simulated independently, by performing the stochastic simulation algorithm in a nested fashion. This is conditioned by the property of the set of reactions, describing the gene regulatory network, being divided into two subsets, i.e. a set of "fast" reactions, which occur frequently in a short time scale, and a set of"slow" reactions, which occur less frequently in longer time scales. This thesis demonstrates the equivalence between this approach and the standard stochastic simulation algorithm and shows its capabilities on two gene regulatory models, that are commonly used as examples in systems and synthetic biology.

The thesis focuses on how to identify the most important input parameters of multi-scale models, that affect the system the most. This is a common practice during the design of bio-logical structures and can be achieved with the sensitivity analysis. It may be difficult to carry out such analysis for complex reaction networks exhibiting different time scales. In order to cope with this issue, an alternative computation of the elementary effects in the Morris screening method is proposed, which is able to sort all the model parameters, independently on their structural or time scale definitions, in order of importance, i.e. which parameter carries the largest influence on the response of the model.

To ease the use of the simulation algorithm and to perform the sensitivity analysis, the thesis presents *ParMSSA*, an OpenCL based engine for performing parallel stochastic simulations on multi-core architectures. *ParMSSA* aims to accelerate the simulations, performed with our approach. *ParMSSA* is capable to run concurrently multiple instances of DMSSA, which are usually needed for reducing the noisy results of stochastic simulations. *ParMSSA* provides also a framework for performing the Morris screening experiment on reaction networks, which allows users to carry out the sensitivity analysis of observed systems. The simulation results provided by the *ParMSSA* can be easily interpreted and can be used to assess the robustness of the bio-logical computer structures.

The proposed algorithms and the proposed simulation engine were applied on two

case studies, i.e. on the Epstein-Barr virus genetic switch and on the synthetic repressilator with multiple transcription factor binding sites. The results of the sensitivity analysis of the repressilator revealed that larger numbers of binding sites increase the robustness of the system and thus the robustness of the oscillatory behaviour.

Key words: stochastic modelling, multi-scale modelling, sensitivity analysis, systems biology, synthetic biology, stochastic simulation algorithm, multiple transcription factors binding sites

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Mattia Petroni

Računalniško podprta metodologija za analizo občutljivosti večnivojskih stohastičnih modelov bioloških preklopnih gradnikov

POVZETEK

Biološko računalništvo je zasnovano na procesnih platformah, ki izhajajo iz bioloških preklopnih struktur z zmožnostjo procesiranja informacij. Te strukture večinoma temeljijo na gensko regulatornih omrežjih (GRO). Njihova dinamika spominja na delovanje računalniških preklopnih gradnikov. Uporaba bioloških preklopnih struktur je trenutno še v povojih, saj je njihova učinkovitost neprimerno manjša od silicijevih ekvivalentov. Kljub temu njihove aplikacije že posegajo v farmakaloška, medicinska in industrijska področja. Natančno načrtovanje na podlagi matematičnega in računalniškega modeliranja je ključnega pomena za razvoj in implementacijo tovrstnih aplikacij.

GRO lahko opišemo s sistemom kemijskih reakcij. Njihovo dinamiko definiramo na molekularnem nivoju z množico kemijskih zvrsti in njihovih medsebojnih interakcij. Za generiranje časovne evolucije vsake kemijske zvrsti lahko uporabljamo stohastični simulacijski algoritem (SSA). V njem posamično reakcijo simuliramo na osnovi Monte Carlo eksperimenta. Glavna slabost te metode je računska kompleksnost, ki se poveča linearno s številom reakcij, ki jih je potrebno upoštevati v simulaciji. V primeru prevelikega števila reakcij postane stohastični simulacijski algoritem neobvladljiv. Tovrstne primere najdemo pri nekaterih GRO, ki vsebujejo več neekoperativnih DNA vezavnih mest transkripcijskih faktorjev in ki so pogosta pri sodobnih bioloških procesnih strukturah. Dodaten problem se pojavi, ko želimo vse reakcije sistema simulirati v enem samem časovnem okvirju. V GRO se nekatere reakcije izvedejo tudi za več redov velikosti hitreje od drugih. Pri GRO, ki vsebujejo več nekooperativnih DNA vezavnih mest transkripcijskih faktorjev, se pojavi ravno taka okoliščina. Vezava transkripcijskih faktorjev poteka mnogo hitreje kot reakcije genske ekspresije, zato je pogosto potrebno to razliko ustrezno obravnavati. Poleg tega je število vseh reakcij vezave in disociacije eksponentno odvisno od števila vezavnih mest.

V disertaciji smo razvili dinamični večnivojski stohastični simulacijski algoritem

(DMSSA), ki občutno zmanjša časovno kompleksnost stohastičnega simulacijskega algoritma pri tovrstnih GRO. DMSSA je zmožen reakcije vezave izvajati neodvisno od reakcij, ki zadevajo gensko ekspresijo. Izvedba reakcij vezave poteka znotraj vgnezdenega SSA. Prednost takega pristopa je v ločenju množice reakcij sistema na dve neodvisni podmnožici, in sicer na množico hitrih in množico počasnih reakcij. Počasne reakcije potekajo v časovni skali, ki sovpada z redom velikosti hitrosti reakcij v genski ekspresiji, kot sta reakciji transkripcije in translacije. Te reakcije DMSSA izvede manj pogosto. Hitre reakcije, kot sta reakciji vezave in disociacije tranksripcijskih faktorjev na/iz DNA vezavna mesta, se po drugi strani izvajajo bolj pogosto. V pričujoči disertaciji pokažemo, da se natančnost DMSSA ujema s SSA. Uporabo DMSSA pokažemo na dveh modelih s področja sistemske in sintezne biologije.

V disertaciji se dodatno osredotočimo na pomen ocenitve občutljivosti večnivojskih stohastičnih modelov GRO, ki vsebujejo več nekooperativnih DNA vezavnih mest. Z analizo občutljivosti lahko sortiramo vhodne parametre na osnovi največjega vpliva na izhode modela. Analiza občutljivosti stohastičnih modelov predstavlja računski izziv zaradi računske kompleksnosti algoritmov, ki se uporabljajo za pridobitev odzivov samih modelov. V disertaciji predlagamo uporabo spremenjene Morrisove metode na osnovi alternativnih elementarnih učinkov, s katerimi lahko ocenimo občutljivost parametrov modela neodvisno od njegove večnivojske razsežnosti.

Za pohitritev simulacij stohastičnih modelov smo v sklopu disertacije razvili orodje *ParMSSA*, ki vsebuje simulator za paralelno izvajanje stohastičnih simulacij z algoritmom DMSSA. Orodje *ParMSSA* je zmožno paralelno izvajati več instanc DMSSA algoritma z različnimi vhodnimi parametri za potrebe analize občutljivosti s spremenjeno Morrisovo metodo. Rezultate pridobljene na podlagi občutljivostne analize z orodjem *ParMSSA* je možno neposredno uporabljati za ocenjevanje robustnosti stohastičnih večnivojskih modelov GRO, ki vsebujejo večkratna nekooperativna DNA vezavna mesta trankripcijskih faktorjev.

Uporabo razvitih algoritmov skupaj z orodjem smo demonstrirali na dveh vzorčnih primerih, tj. na analizi mehanizma preklopa v virusu Epstein-Barr in na analizi sintetičnega represilatorja z več vezavnimi mesti za transkripcijske faktorje. Na slednjem smo pokazali, da povečevanje števila vezavnih mest za transkripcijske faktorje povečuje robustnost sistema in s tem oscilatornega delovanja.

Ključne besede: stohastično modeliranje, večnivojsko modeliranje, občutljivostna anali-

za, sistemska biologija, sintezna biologija, stohastično simulacijski algoritem, večkratna vezavna mesta transkripcijskih faktorjev

ACKNOWLEDGEMENTS

First, I have to thank my advisor Miha Moškon for the patience he always had in guiding me and motivating my research. His suggestions, corrections and his pioneering work in the computational biology group here in Ljubljana made this thesis possible, so thank you.

Special thanks to Luigi Canciani MD, for helping us to receive the necessary funding for our research and to Iztok Lebar Bajec for his innumerable tips about LaTeX and design. Thanks also go to Marc Leguen de Lacroix for proofreading my English writing. And a special thanks to my colleagues of the FRIdom band: Igor, Brane, Nejc, and Matevž. Thanks for enriching my time by playing together good music while I was immersed in writing.

I would like to thank all the current and former colleagues of the Computer Structures and Systems Laboratory and of the Laboratory for Adaptive Systems and Parallel Processing for all the funny moments that we spent during coffees and lunches. Thanks for all the endless discussions and the innumerable laughs.

A great thanks to my parents who always motivated my scientific curiosity both for biology and technology, and allowed me to study the things that I always loved.

And at last, with all my heart I want to thank my beloved Marija, without whom this thesis wouldn't have seen the light of the day. Thank you for the immense patience you had along the way and for your brilliant wisdom which constantly lights my path as a father and as a scientist. I love you.

— Mattia Petroni, Ljubljana, April 2018.

to Isobel and Marija

CONTENTS

I

Introduction

A senior professor once asked
a student of his: "Why computers can not have emotions?"
The student replied: "Because they are insensitive."

A fun fact during a lecture of algorithms and data structures.

What do a bacteria, a fishing hook and a lonely tree in an empty field have in common? Apart from some hilarious jokes that can be found on Internet forums frequented by young scientists in rough hours in the night, probably nothing would come in mind at first, and we may spend some time thinking before surrender to the peculiarity of this question. In fact, this question is indeed strange and someone may really doubt there is something that connects together all these animated and inanimate objects. For instance, any sceptic would be convinced that the only thing these objects have in common is a place in the Oxford dictionary, besides the basic atomic composition that is shared by all the matter in the universe of course. Probably with more conviction a religious person would say that all these three objects are creations of God. And an experienced sailor would say that all these things can be found on the Isle of Man. At last, a scientist may observe these objects with a deep introspective and determine a common shape that all these objects can have in specific circumstances.

Leaving comic fantasy, personal belief and imagination aside for a moment, it is really a challenge to find such a connection and by the time we read this first page someone would probably doubt the truthfulness of this question. However, there is nothing wrong with it and a plausible explanation of such a connection between these objects really exists, but without saying too much at this point we could provide a little suggestion, that would introduce the main topic of this dissertation.

When observing any natural event, we are usually interested to make some empirical statement about it (much like anything else). In order to do so, there are often required long term observations and large data gathering. But usually we are only interested in very simple conclusions or facts, such as the quantification of some measurable value, a rough explanation of some unmeasurable quality, or just a simple estimation or characterization of the event itself, i.e. how bad or good it is. For instance, if we observe a meteorological event, such as a North Atlantic storm hitting the western coast of Ireland, probably the first things we notice is the wind blowing at fifty (or more) miles per hour and possibly some gigantic waves breaking on the rocks. If we would like to collect some empirical information from such a meteorological phenomena, perhaps we will begin by measuring the speed of the wind or by estimating roughly the height of the waves. The intensity of the wind can be measured by searching for some trees in the nearby and watching how much their branches are bended. A simple conclusion statement will hold that *more the branches are bended, bigger is the force of the storm*. The quantity of *bending* can therefore tell us how powerful is this natural event (or either how brutal). A similar conclusion can be made by estimating the height of the waves by observing how much splash the waves produce when smashing on the rocks. Here the quantity of *splash* can address the same measure as the bending does for the wind. We may therefore continue by providing different metrics and measurements, until a robust confirmation will validate our conclusions.

By observing this storm for a long time, we may therefore gather a lot of measurements. However no matter what kind of natural event we are observing, we may always find out that the values of these measured quantities, may never cross certain limits. For instance, if the wind speed reaches more than hundred miles per hour, we would probably witness eradication of some trees and their inevitable fall. On the other hand, if the height of a wave exceed more than twenty meters we may witness not just a big splash on the rocks, but probably a devastating flood all around the point of impact. Of course in such an occasion we are observing an exceptional storm, but such abnormal events, as eradication and flooding, can be simply described as a deviation from a reference interval of our chosen metrics – wind and waves. We can therefore conclude that the bigger this deviation is, then bigger the damage will probably be.

In this dissertation we wanted to address a similar statement, but with one major difference. We would not observe huge North Atlantic storms neither similar brutal meteorological events. Instead, the main natural event of our study will be the gene

regulation in living organisms. We would like to, or at least try to, extrapolate similar conclusions as was done for the storm, by modelling the biochemical events that shape the gene regulatory networks of biological systems. More specifically we are interested in those gene regulatory networks that can perform information processing just like common computers.

1.1 Motivation

The ultra large scale integration (ULSI) and the increasing miniaturizing processes used by computer chips manufactures are pushing the boundaries of computing capabilities of modern computers. The semiconductor technology today enables placing billions of transistors on a single chip. The so-called Moore's law, which states that the number of placed transistors on a chip doubles in size every two years, is still valid after more than 40 years from its formulation. However several semiconductor manufacturers recently agreed, that we are about to reach the upper limit of the number of transistors that are physically possible to place on a single surface in an integrated circuit [1]. Despite the fact, that such limits and the eventual slowdown were already predicted several years ago [2], the development of complementary metal oxide semiconductor technology (CMOS), did not stop, and according to the International Technology Roadmap for Semiconductors (ITRS), the 5nm CMOS-based transistors[1] could be achieved for manufacturing commercial chips. At 5nm the CMOS transistor will be still immune from quantum tunneling, that may heavily affect the response of logic gates. Although ITRS predicts that use of novel materials and structural enhancements, such as 3-D circuits, can lead to improved chip fabrication, and enable further development and improvements in computer chip performances, a large number of physicists, chemists and computer scientists are searching for novel processing platforms able to perform logic computation. In this vast investigation and research, biological systems came up as a possible processing platform. In fact the advances in molecular biology and biotechnology of the last two decades allow certain CMOS logic gates and other logic structures to be re-visited from a biological point of view. Advances in synthetic biology have shown that implementation of robust *bio-logical gates* is possible [3–9]. As tempting this possibility is, problems arise when constructing complex circuitry. Biological gates do not always obey the same behaviour as CMOS gates. For instance,

[1] 14 nm transistor technology is currently in use at the time of this dissertation.

fan-out and logic input/output levels may differ profoundly within the same circuit. A huge endeavour has been tackled by biotechnologists, computational and synthetic biologists in order to develop robust CMOS-like bio-logical gates, capable to perform any logic function. One of the aim of synthetic biology is in fact to provide modular constructs to be used for creating scalable logic circuitry, such as logic gates, memory cells (i.e. *flip-flop*) and oscillators [10–13].

Synthetic biology has embraced this challenge in the last two decades, and the results are countless [14]. By exploiting the recombinant DNA technology, synthetic biologists can engineer new devices in living cells [15]. In light of these advances plenty of novel laboratory methods and techniques have been developed [16–20]. These methods are co-responsible for today first commercial products, e.g. in pharmacological, environmental and fuel production applications [14, 21–23]. Examples of such devices include single logic gates NOR [4, 7, 24], NOT, AND, NAND [4, 25, 26], XOR, XNOR gates [4, 24], logic operators [27, 28], toggle [6, 29, 30] and genetic switches [31], biocomputers and signal processing applications [25, 32], pattern formation functionalities [33], biomedical applications [21], genetic oscillators [34, 35], counters [36], memory storage [37–39], and also more complex logic circuits [40].

The majority of these devices is based on *gene regulatory networks* (GRNs) because of their programmable nature [29]. A GRN can be viewed as the basic framework used to implement biological systems with information processing functionalities. However, the capabilities of these systems in terms of complexity and size, i.e. to be used modularly in large scale circuits, are today still limited [41]. With the advent of synthetic *transcription activator-like effector* (TALE) [42–44] and the recent discovery of the CRISPR-Cas9 system [45–58], a wider range of more complex, modular and scalable GRNs can now be implemented and hence the ability to construct complex logic connected structures and circuits *in vivo*, such as those in modern computers, currently becomes no longer an unachievable goal [7]. Such GRNs are often referred as *synthetic gene regulatory networks* (SGRNs) [59].

The construction of SGRN has been performed in almost all types of cells used in bioengineering and biotechnology: yeast [60], bacteria [61] and mammalian [62] cells. The design of these systems is rarely straightforward and often requires the trial and error [59] or more selective strategies [63][2]. Mathematical and computational mod-

[2] The inverse problem is to discover the structure of GRN while having available a long DNA sequence of base-pairs. This is one of the main problems in the field of bioinformatics [64, 65].

elling techniques have proved crucial for helping the design process to minimize the costs and the time needed to develop these systems. Modelling provides a quantitative estimation of the dynamics of designed biological systems before their experimental implementation [66–68]. Current efforts emphasize the need of accurate modelling of gene expression dynamics and more specifically of large GRNs. The aim of modelling is to construct mathematical or computational models of biological systems by including all their physiological characteristics in a formal mathematical way. These models can be used not only to quantitatively validate the response of underlying systems, but also for predicting their behaviour through time. This is particularly important in the case of sequential logic circuits, where a certain logic level may be required for triggering the response of logic gates at specific time. Furthermore, the comprehensive view of the system's behaviour helps to quickly assess the logic correctness of the circuit at any time.

Biological systems are known to reflect robust behaviour, even in a noisy environment, which is a consequence of long evolutionary processes [69]. The robustness of a biological system (or model) is the capacity of the system to maintain intact its current operating mode (or state), despite the continuous perturbation of its internal and external inputs. Robustness needs to be estimated precisely and in different environmental conditions in order to optimize the design of SGRNs. Robustness is believed to be the key factor of adaptability in the evolutionary process of biological systems [70]. In cell biology, the robustness is a property that permits every single-cell system to adapt to unpredictable environment conditions, e.g. the bacterial chemotaxis dynamics [71]. Qualitative and quantitative robustness estimation can be therefore of crucial importance in the design of reliable and scalable biological systems with novel functionalities.

Although a general quantitative measure for robustness has still not been established, numerous efforts have come from various scientific disciplines, especially from the control theory. Many metrics for robustness have been proposed in the last decade [72–78]. Different studies show that the robustness can be effectively estimated with the aid of sensitivity analysis [79, 80], such as parameter sensitivity analysis [81–83]. Additional benefit of using the sensitivity analysis is its ability to guide experimental work, meaning that with its results, it is possible to identify the inputs of the model that affect the system's outputs the most. Such inputs should be a subject of further experimental investigation [84]. This property makes the sensitivity analysis suitable for estimating

the robustness of synthetic biology systems.

The majority of *state-of-the-art* sensitivity analysis approaches can be applied to deterministic models only, which are unfortunately unable to capture the noise induced effects, inherently present in biological systems. However few exceptions were reported in literature [85–87].

Switching mechanisms in biological systems are known to be subjected to large amounts of noise. This may represent a major problem in the design of robust logic structures with higher complexity. The design of biological systems with high scalability and reliability is therefore far from being straightforward. Stochastic modelling approaches have proven to be useful in this context because they are capable to model the noise induced dynamics in the underlying systems, by stochastically simulating the reactions occurring in the systems [88–90]. These techniques are unfortunately computationally demanding in most cases, especially, if the number of reactions is very high. This can occur in novel biological systems containing multiple non-cooperative transcription factor binding sites [7, 42, 91, 92]. One way to decrease the computational complexity of these modelling approaches is to use a multi-scale modelling approach [93–95], in which several dependent chemical reaction subsystems are presumed to be independent and can be therefore analysed and simulated separately in different time-scales. Stochastic multi-scale modelling maintains all the pros of stochastic modelling and furthermore, it allows to increase the computational performance of the simulation algorithm, and therefore increases the accuracy of the approximations of underlying chemical kinetics. Despite the rapid development of sensitivity analysis methodologies in recent years, there is still a lack of methods that can evaluate the sensitivity of stochastic multi-scale models, which can be sometimes the only way to obtain a valid dynamical response of the system *in-silico*. Establishing a methodology to overcome this problem is vital for the robust design of SGRNs and hence for further development of synthetic biological systems with information processing capabilities.

In this doctoral dissertation we developed a methodology for performing the sensitivity analysis of stochastic multi-scale models of GRNs containing multiple non-cooperative transcription factor binding sites. Such methodology can be applied along the design strategy of SGRNs with information processing capabilities.

1.2 Methodology

The development of the aforementioned methodology has been carried out through the following phases:

- In the first phase we reviewed and analysed the current *state-of-the-art* stochastic multi-scale simulation techniques and the sensitivity analysis methods. We favoured the computational algorithms that are able to deal with large data structures efficiently and allow a high level of parallelisation in order to be exploited in modern multi-core architectures. We then chose those sensitivity analysis methods, that require the minimum computational effort for obtaining the simulations' response, e.g. the Morris sensitivity analysis technique.

- In the second phase we proposed a variant of the stochastic simulation algorithm, called DMSSA, which is able to tackle the multi-scale nature of GRNs, that contain multiple non-cooperative transcription factor binding sites. The validity of the responses obtained with this algorithm and its equivalence with the stochastic simulation algorithm were proved on a simple multi-scale model, that is being used constantly in systems and synthetic biology. We implemented DMSSA alongside a simulation engine, called *ParMSSA*, which is capable to perform multiple parallel simulations.

- In the third phase we completed our methodology by implementing the Morris sensitivity analysis technique with few major enhancements, for obtaining the parameters sensitivities of the multi-scale models. These sensitivities allow to classify the model parameters by influence upon the models' outputs.

- In the fourth phase we applied the methodology on two case studies: on the genetic switch of the Epstein-Barr virus and on a genetic oscillator.

1.3 Scientific contributions of this thesis

Overall we achieved the following contributions:

- *A new simulation algorithm for GRNs with multiple non-cooperative transcription factors binding sites.* A variant of the commonly used stochastic simulation algorithm is proposed for performing accurate simulations of GRNs containing

multiple non-cooperative transcription factor binding sites. The algorithm is used inside a simulation engine called *ParMSSA*, built for simulating systems and synthetic biology models of GRNs with any feasible number of binding sites. This contribution is presented in chapter 4.

- *Establishment of a methodology for enhanced sensitivity analysis of stochastic multi-scale models.* The *state-of-the-art* sensitivity analysis approaches lack the capability to investigate stochastic multi-scale models. Establishing a methodology that will fill this gap is essential for the robust design of SGRNs and hence for the development of synthetic biological system with information processing capabilities. We enhanced a sensitivity analysis technique, based on the Morris screening experiment, which is capable to estimate the parameters sensitivities of stochastic multi-scale models that are frequent in both systems and synthetic biology research. These parameter sensitivities can be directly used as an estimator of the model's robustness. As such they can be used as metrics for the design of robust biological systems with information processing capabilities. This contribution is presented in chapter 6. An extension of the metrics for the design of robust biological systems with information processing capabilities can be therefore established by exploiting the sensitivity values from the proposed methodology.

1.4 Dissertation overview

The dissertation is organized as follow.

Chapter 2 presents an introduction of the foundations of molecular biology, emphasizing the information processing mechanisms inside the cell, especially those that are of close interest for computer engineers. The main topics of systems and synthetic biology are also introduced, as well as an overview in gene regulatory networks with multiple non-cooperative transcription factor binding sites, which represent the main objects of study of the thesis.

Chapter 3 contains a basic introduction in the field of reaction kinetics and a brief overview of the *state-of-the-art* modelling and simulation techniques. These short introduction is essential for understanding the basics of the multi-scale computational modelling presented in chapter 4.

Chapter 4 presents a computational solution to the exponential complexity prob-

lem of describing the transcription factors-DNA binding and unbinding events over a cluster of multiple binding sites, through simple reactions. The solution is then implemented in an algorithm called DMSSA, which can be used generally for simulating the dynamics of any GRN containing multiple transcription factor binding sites. The correctness of the algorithm is then validated with a sample model of GRN containing multiple transcription factors binding sites.

Chapter 5 reviews the most commonly used approaches to perform the sensitivity analysis in systems and synthetic biology, with a major focus on screening experiments, more precisely the Morris method.

Chapter 6 presents a set of alternative output functions to be used for computing the elementary effects in the Morris screening experiment, in order to enhance the ability of the sensitivity analysis method to cope with the complexity of stochastic multi-scale models of GRNs containing multiple non-cooperative transcription factor binding sites. The Morris sensitivity method is adapted to handle the complex outputs of such models.

A simulation engine called *ParMSSA* is presented in appendix A.3, which implements among other things, the algorithm DMSSA, and exploits the coarse parallelism of modern computer architectures, to ease the computational expensiveness of the screening experiments.

The full capabilities of the engine *ParMSSA* are presented in chapter 7 on two case studies: the model of the Epstein-Barr virus' genetic switch and a SGRN of a robust biological oscillator. In this chapter is shown that by analysing in both models the parameter sensitivities obtained with the method proposed in chapter 6, it is possible to sort the models' parameters in order of importance. Such information can be useful to understand:

- how the system in reality would behave, if perturbed through one of its inputs (a valuable information in pharmacological research), and
- what the designer must be aware of during the design process of SGRNs, in other words, which model parameter can be neglected.

Chapter 8 contains the final remarks of the dissertation and a short analysis of the future research directions.

2

Information processing in gene regulatory networks

Lt. Cmdr. Matt T. Sherman: Where is Lt. Holden?
Lt. Watson: When the air raid started they took off. All he said was - "In confusion there is profit."

Operation Petticoat, 1959

The most peculiar characteristic of any biological system is the inherent ability to process information at a molecular level. The cell itself is a prodigious machinery where thousands of different chemical reactions take place. Each of this reactions is a small piece of a complex set of biochemical processes that ultimately take the semblance of life. Cellular *respiration* and *replication* are two common examples. The physiology of the cell is shaped by its intrinsic and complex dynamics. *Metabolic*, *signalling pathways*, and *gene regulatory networks* define some of the main mechanisms that drive these large and complex processes.

2.1 Information processing machinery in the cell

All of these mechanisms in different ways perform biochemical transformations over complex organic molecules, which result in more complex or smaller biochemical compounds. The chemical reactions that involve the *deoxyribonucleic acid* (DNA) and the *ribonucleic acid* (RNA) play a central role, not only for control, guidance and driving the cell physiology, but mainly for expressing the organism's phenotypes. In fact the information about an organism's phenotype is encoded in the DNA molecule. Long sequences of nucleotides called *genes* can encode the information regarding structure and composition of *proteins*. When the genes are expressed, the proteins are assembled via the concatenation of *amino acids*, following the nucleobases sequences that

are found inside the genes. Proteins have different functions inside the cell: they can work as reactions catalysts, control the gene expression and DNA replication, drive the intra and extracellular signalling response and facilitate the transport of molecular compounds inside and outside the cell [96].

2.2 Gene expression

DNA can contain several hundred of thousands of different genes, i.e. the entire genetic material of the organisms. In this case it is also referred as the "*genome*". Each gene in the genome can encode the information regarding a specific phenotype, DNA regulatory protein, metabolites, enzymes, or basically everything composed of amino acids. The gene expression is indeed one of the most remarkable information processing events occurring in the cell. The information inside a gene, that is coded with a unique sequence of nucleotides, is processed in stages. First, it needs to be copied out

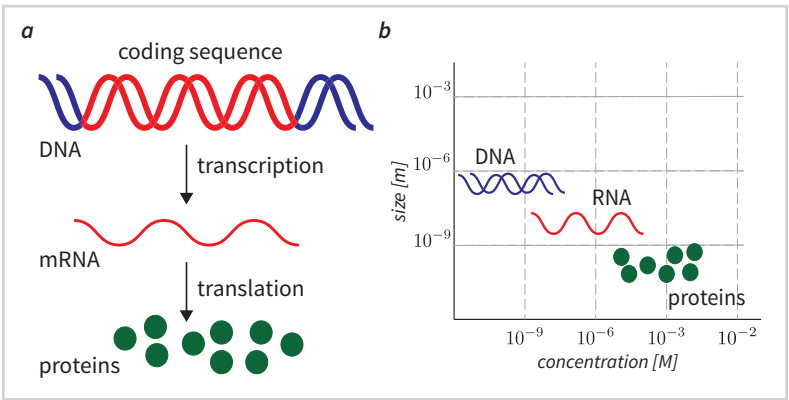


Figure 2.1

The three main components of the gene expression: DNA, RNA and proteins. *a)* The gene expression scheme. The coding sequence of a gene is transcribed into the mRNA strand, which in turn transports the coded information inside the ribosome, where the protein can be finally assembled. *b)* The ratio between the size and the concentration of the three components of gene expression inside the cell [97].

from the DNA double strand. This process is called gene *transcription* and the product of this stage is a single stranded ribonucleic acid called *messenger RNA* (mRNA). In the next stage the copied information, carried by the mRNA, is transported into the ribosomal site. Inside the ribosome another ribonucleic acid called *transfer RNA* (tRNA) is responsible to decode each adjacent triplet of the nucleotide sequence into a specific amino acid. This process is called *translation*. In the last stage the amino acids are chained together in the same order as in the nucleotide sequence transcribed from the gene. The product of this assembly is the *synthesis* of the final protein, see Fig. 2.1.

Transcription

The transcription can be described as a three stage process. The first stage is called *initiation* and begins with a binding of an enzyme called *RNA polymerase* (RNAP) to a specific site inside the gene called *promoter*[1]. When the RNAP is bound to the promoter, it opens the double helix of the DNA in two strands by separating the nucleobases bond, and begins sliding along the coding sequence, which usually lies in proximity or next to this promoter site, separating the two strands of DNA furtherer. This stage is called *elongation*. This sliding is always directed in the direction 3' → 5'[2]. On each open nucleotide of the open DNA strand, the transcription machinery[3] facilitates the binding of a free complementary nucleotide (see Fig. 2.2). Once all the nucleotides of the open strand are bound with their complementary bases, they form a new ribonucleic backbone strand that is the basis of the mRNA. The formation of this complementary strand ends, when the RNAP reaches a site along the gene called *terminator*. Every gene has its own terminator site. When the mRNA is fully assembled and detached from the original DNA strand, the transcription of the coded information is complete and the two original DNA strands can be tied together again in the classic double helix formation. This last event represents the final stage of transcription and is called *termination*.

Translation

The mRNA then travels[4] towards the ribosome. In eukaryotes, the mRNA leaves the nucleus, where the transcription occurred. In prokaryotes the nucleus is absent, therefore the mRNA can reach the ribosome faster. Once the mRNA binds the ribosome, the translation process can begin. Similarly as transcription, also the translation can be divided in stages [96]. Inside the ribosome, the ribonucleic acid tRNA

[1] Note: the RNAP binds in the close proximity of the promoter. The promoter itself is in the position on the DNA sequence, where the genetic information of the gene starts and hence is the location, where the transcription should begin.

[2] The 3' and 5' notations are used to indicate the orientation of the carbon atoms in the deoxyribose sugar ring, that composes the DNA backbone (see orientation/directionality in [96, 98] for more details).

[3] Here the transcription machinery is referred as the complex group of transcriptional initiator complexes, transcriptional cofactors and transcription factors that drive the transcription along all the three stages. A more complete and exhaustive explanation of these machinery can be found in any molecular biology textbook (see [96, 98] as examples).

[4] More precisely, the mRNA, like every other molecule inside the cell, floats inside the nuclear matrix and cell's cytosol.

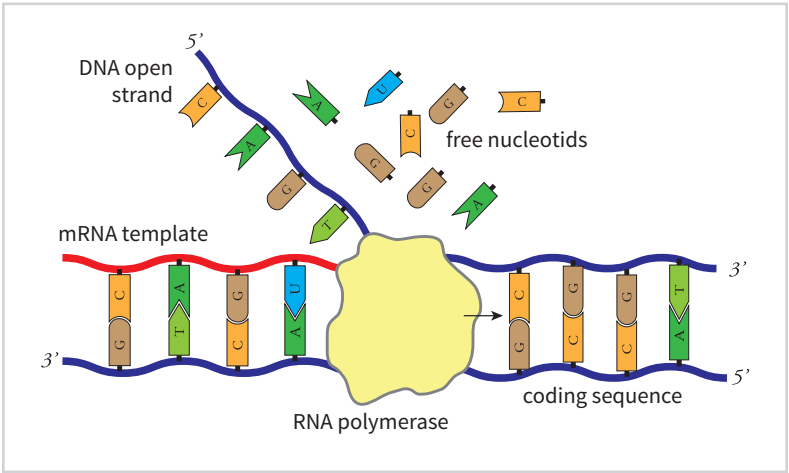


Figure 2.2

A simple schematic of the transcription process. The RNAP after binding on the promoter region starts sliding along the coding sequence of the gene and opens the DNA double helix, separating the two strands, i.e. the elongation stage. The mRNA template begins to form on the open strand directed 3' → 5'.

starts reading the coding sequence of the mRNA by binding to a specific triplet called *start codon*, which is the nucleobases sequence AUG. The ribosome slides forward the mRNA strand by three nucleotides at a time, enabling the tRNA to read the entire coding sequence of the mRNA, also known as the *open reading frame* (ORF). The reading ends when the tRNA binds over a particular triplet called *stop codon*. The tRNA itself consists of a complementary anticodon (a complementary triplet which enables to bind the tRNA to a specific triplet on the mRNA), and an amino acid. This amino acid is tightly bound to the anticodon sequence according to the genetic code [96]. Each tRNA anticodon carries its own specific amino acid. Thus the tRNA represents the link between the genetic information and the protein structure. When the tRNA is bound to the mRNA inside the ribosome, the carried amino acid is detached from the tRNA and it is chained together with the amino acid of the previous bounded tRNA. By repeating this process over all the codons of the mRNA, a polypeptide is obtained. This chaining process ends when the stop codon is reached, for which the complementary tRNA anticodon does not provide any amino acid. The chain of amino acids that is formed in this fashion represents the primary structure of the new synthesized protein.

2.3 The control of gene expression

The expression of proteins is usually regulated. There are several mechanisms that prevent the genes to be expressed unsupervised. These control mechanisms are known as *gene regulation* [99].

Post-translational regulation for instance ensures, that the concentration of a free synthesized protein never surpasses a certain limit. This can be achieved by protein structural modifications, e.g. with *phosphorylation*, which can switch on or off the functionality of individual protein. Other gene expression control mechanisms involve modifications of the RNAs in various stages of the gene expression. Some of this modifications facilitate the RNA and proteins degradation, or the mRNA inhibition via the *microRNA* post-transcriptional regulation [100], which can cause the loss of the gene product information. If the control of gene expression is performed at the gene transcription stage, then the control mechanism is called *transcriptional regulation* and the involved proteins (or RNAs in some cases) are called transcriptional *regulators*. The aim of transcriptional regulation is to induce or inhibit the gene transcription by starting or blocking the initiation stage.

2.3.1 Transcriptional regulation

In order to initiate the transcription, the RNAP needs to bind on the gene promoter region. This process is driven by the transcription machinery, a complex set of DNA binding proteins, enzymes and signalling proteins that actively participate in the gene expression[5]. Some of this proteins have the capability to facilitate (or even recruit) the nearby RNAP and guide it to its binding site on the gene's promoter. Such proteins are called transcription *activators*, because they not only augment the chance of initiating the transcription correctly, but also sometimes to enable it at all. Other proteins have the opposite function i.e. they prevent the RNAP to bind on the promoter. Such proteins are called gene *repressors*. Both activators and repressors are known as *transcription factors* (TF).

[5] Now, in the majority of cases, especially in mammalian cells, this scenario is highly unlikely to happen independently, because the DNA is packed inside a complex structure called *chromatin*, which usually prevents the interactions of the DNA with other compounds [96]. Therefore, in order to initiate the transcription of a gene, first the chromatin complex needs to be opened, so the DNA inside can be accessed. This event concerns *epigenetics* and we refer to [96] for more insights about this topic. Later in this dissertation we will assume, that this task is (eventually) always performed, regardless the complexity of the observed gene regulation.

The majority of TFs are complex and often large molecules, usually composed by a functional and a binding domain. The former performs one of the two regulatory functions – activation or repression, while the latter allows the TFs to bind to a specific *binding site* close to the promoter region on the DNA. These *transcription factor binding*

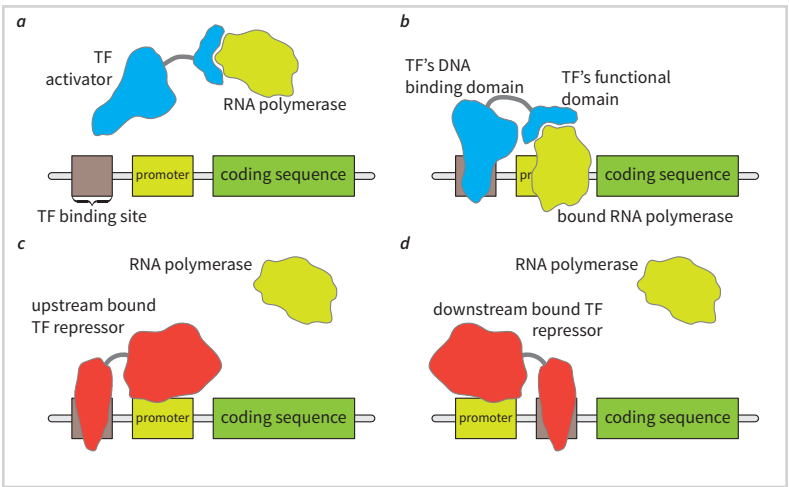


Figure 2.3

A simple scheme of transcriptional regulation possibilities. *a)* A transcription factor (TF) activator, composed by a DNA binding and a functional domain, recruits the RNAP in the nearby of a gene that needs to be transcribed. *b)* The activator binds on its binding site close to the promoter and enable the RNAP to initiate the transcription. *c)* A repressor binds to an upstream binding site close to the gene's promoter and prevents the RNAP to initiate the transcription. *d)* A similar situation than (c) whereas the binding site is located downstream to the promoter region.

sites (TFBSs) are in literature known as *cis-regulatory modules* or *elements*. If the TF is an activator then, when it is attached to its binding site, the functional domain recruits the RNAP and facilitates its binding to the promoter (see Fig. 2.3 *a,b*). TFs can have a higher binding affinity than the RNAP, and therefore they can find the right gene location inside the DNA more quickly. Repressors can bind strongly over similar locations of activators (sometimes even on the same sites, including downstream to the promoter region), but instead to facilitate, they prevent the RNAP to initiate transcription (see Fig. 2.3 *c,d*). A concise representation of activation and repression mechanisms is depicted on Fig. 2.4.

Not all the genes have TFBS in the promoter's vicinity. Many promoters do not have associated transcriptional regulators at all. In such cases these genes can be transcribed by the RNAP without negative (or positive) influence from TFs. The promoters in such genes are called *constitutive promoters*.

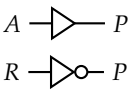
2.3.2 *Gene switching*

When the activators are present, the gene can be expressed. We can therefore say, that the activators trigger the production of proteins. The counterpart in digital circuits of such a device is the *driver* (or *buffer*). On the other hand the repression works exactly as an inverter, because the presence of TF repressors inhibits the gene transcription and therefore prevents the protein synthesis. A short schematic representation of these two switches is depicted in Fig. 2.4, while the symbols used in literature to represent both activation and repression of proteins are shown in Tab. 2.1. These two gene switching mechanisms allow TFs, i.e. *input signal*, to enable or disable the presence of a protein, i.e. *output signal*, inside the cell. This holds of course only if the protein in question can be synthesized solely from the coding sequence of the gene that these TFs regulate.

As the voltage switching in CMOS transistors is the foundation of all logic gates operations in digital circuits, the gene switching paradigm represents the foundation for implementation of information processing applications in biological systems [15, 59].

Table 2.1

Symbols used in literature to represent the protein production dependency from transcription factor activation and repression.

type	*symbol*	*logic*
activation	$A \longrightarrow P$	
repression	$R \dashv P$	

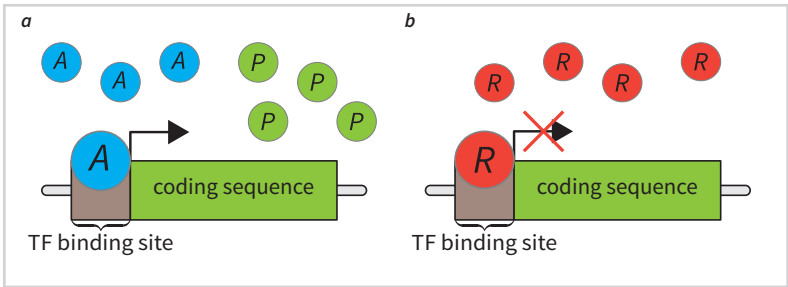


Figure 2.4

The two gene regulation motifs: *a)* activation and *b)* repression. Both activators (A) and repressors (R) can be seen as an input signal in the gene expression mechanism, while the product protein (P) of the transcribed coding sequence can be seen as the output signal.

2.3.3 Complex cis-regulatory modules

It is not always possible to describe the gene regulation in terms of digital switches as the gene promoter being switched on/off by the TF binding in its closed proximity. The gene regulation generally exhibits a more variable behaviour that is not necessary binary in the value of the gene product, i.e. only a high or low concentration of the output protein. This of course prevents any interpretation of gene regulation as a digital switch. Moreover, close to the promoter region, there could be more than one TFBS. In such scenario the binding dynamics of TFs to their cis-regulatory elements becomes much more complicated to describe than the case depicted in Fig. 2.4. Furthermore, several different types of TFs exists in nature that can influence each other even in the smallest concentrations. An extensively large spectrum of DNA binding possibilities may also exist even for the same TFs, which enriches further the complexity of the transcription regulation. Here we shortly present the main TFs binding scenarios that can occur during gene regulation.

Clusters of TF binding sites

Genes can be often regulated with multiple TFBSs, which are known as *clusters* of cis-regulatory elements [101]. Multiple TFBSs for a single promoter allow a variety of possible *regulation dynamics*, *competition* and *cooperativity*. Examples of clusters of cis-regulatory elements can be found in the well studied *Drosophila melanogaster*, in the yeast *Saccharomyces* [102] as well as in viruses, such as in the Epstein-Barr virus [103, 104]. A simple schematic of a cluster of TFBSs is depicted in Fig. 2.5.

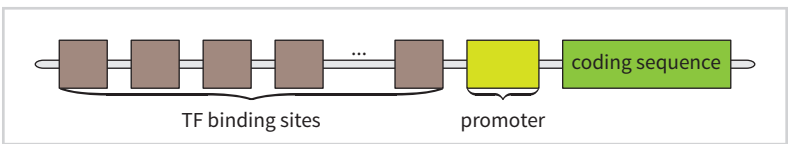


Figure 2.5

The multiple (or *clustered*) TFBSs-cis-regulatory module motif. On each adjacent binding site can bind one TF.

The cluster is not necessary structured with equally spaced binding sites locations (see for instance the NF-κB binding sites in the *NFKBIA* gene [105]). It is important however, that these binding sites are located in the proximity of the promoter, so the TF activator can recruit the RNAP in such a way, that the polymerase will be still able to bind to the promoter, when the TF is bound to one of these sites. Multiple TFs can therefore bind to these sites, and act together to recruit or inhibit the RNAP binding.

However, the dynamics of competition and cooperativity on such clusters are still not well understood and are a current topic of research in molecular biology.

Cooperativity

Normally a TF is a simple protein that binds directly to its TFBS upstream or downstream of the promoter region (Fig. 2.4). In this scenario the gene expression is known to be gradual in respect to the input TFs concentration [105]. The bound TF can either recruit or inhibit the RNAP to bind over the promoter of the coding sequence. If on a cluster of binding sites, multiple TFs bind independently without affecting each other or the subsequent regulation (i.e. their binding reactions are independent events), then the transcription rate increases proportionally to the amount of bound TF activators and inversely proportional to the amount of bound TF repressors. Such dynamics defines non-cooperative TF binding and it will be one of the main objects of study through this dissertation.

In the opposite scenario, when the binding affinity of a TF increases with the number of already bound TFs on the cluster of multiple binding sites (i.e. their binding reactions are not independent events), then the transcription rate increases (or decrease) non-linearly to the amount of bound TFs. This TF binding dynamics is called *cooperative binding*. The term cooperativity is normally used to indicate the type of non-linear effect that the TFs binding have over the transcription products.

Competition between different TFs

Two or more different types of TFs, either activators or repressors can also bind to the same binding site. This is a scenario where TFs "compete" for binding on the regulatory site. Which TF would prevail depends on different conditions, from binding affinity to the concentration of individual type of TFs. The transcription regulation driven by TFs competition becomes exceptionally complex to describe, if multiple adjacent binding sites are present in the promoter proximity. In such a scenario activators and repressors may bound at the same time on different but adjacent binding sites. In this case the regulation effect can be difficult to predict and usually experimental verification is needed. We refer to [106, 107] for more insights and for a more complete review of the TFs' competition dynamics.

Rules of activation

Usually clusters of TFBSs allow the binding of only one of the two types of TFs, either activators or repressors. However, there are exceptions as in the case of synthetic genes, produced from the designs in synthetic biology (see section 2.6), that can be regulated by synthetic proteins (activators and repressors), which bind over the same cluster of TFBSs [4, 6, 7], creating thus a binding and regulative competition. This is then a scenario of TFs competition over multiple binding sites, that deserves to be investigated. Even if only one type of TF has the ability to bind over a cluster, there are some questions that arise and need to be answered. For instance, how multiple bound activators recruit the RNAP? How the regulation behaves in the presence of just one bound repressor in the cluster? In the work of Giorgetti et al. [105] and Lengyel [108] there were investigated possible regulation rules of activations of genes with multiple bound TFs in their cis-regulatory elements [105]. A regulation rule is an hypothesis of activation or repression for a specific combination of bound TFs over the cluster of binding sites. Since the combinations that have to be evaluated can exponentially grow with the number of binding sites in the cluster, it is obvious that this is not a simple problem to tackle. However some of these combinations can be intelligently grouped in a way, so that each group is composed only by rules with identical effect on gene regulation. In the work of Giorgetti et al. [105] the following rules for activation were analysed:

- additive,
- all-or-none and
- singular.

These rules are also schematically depicted in Fig. 2.6. The additive rule states that the transcription of the regulated gene is activated with a rate, that is proportional to the number of bound TFs activators. Bound TFs hence contribute in an additive manner to the promoter's activation. In the all-or-none rule, the transcription of the gene is activated only, if every binding site of the cluster is occupied by a TF activator. In this scenario the RNAP is recruited by all the bound activator together. The singular rule states contrary that, at least one activator needs to be bound, so that the RNAP can be recruited.

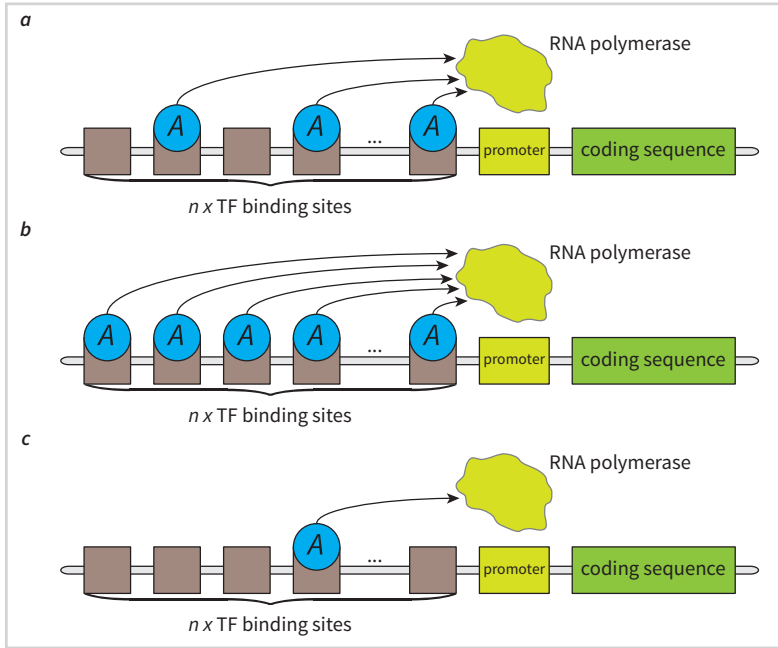


Figure 2.6

The possible rules that can govern the gene activation in the scenario where multiple TFBSs are located in the proximity (upstream) of the promoter region. *a)* The scheme showing the additive rule. In this scenario each bound activator contributes equally to the recruitment of the RNAP and eventually to the rate of transcription. *b)* The all-or-none rule. In this case all the binding sites need to be occupied so the RNAP can be successfully recruited. *c)* The singular rule. At least one activator needs to be bound on the cluster of binding sites, so the RNAP can be recruited.

2.4 Gene regulatory networks

A gene regulated by a TF may encode a protein, which in turn can be a TF itself. If multiple genes are regulated by TFs that are expressed within the same set of genes, then these genes form a *gene regulatory network* (GRN). In the literature GRNs are also called *transcriptional* or *cis-regulatory networks*. Since the gene regulation, as we showed earlier, performs like a switch, GRNs can be seen as a network of connected switches, similarly as a digital circuit, and hence they can be considered as a processing platform. We often refer to the network of genes' switches also as a *gene* or *genetic circuit* [109]. We will see later in this chapter, that through this network of switches, biological systems can store and process logic information in a way that is analogous to digital computers.

Since GRNs often exhibit one or more logic switch motifs (they are often composed of cascade sequences of gene switches), their dynamics can be described in terms of logic

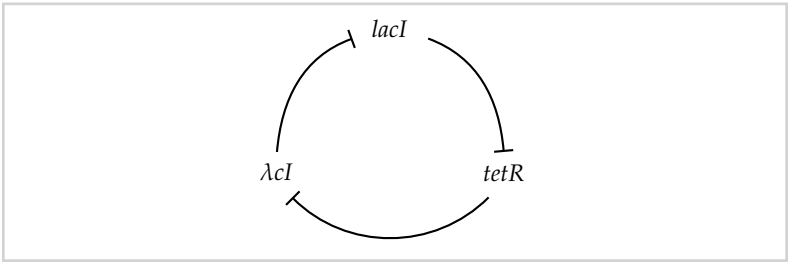


Figure 2.7

An example of a GRN-graph, i.e. the *repressilator* circuit [34], which has an oscillatory response in the concentration of all the three TFs that are involved in the gene regulation: *λcI*, *lacI* and *tetR*.

variables and functions [110–112]. A key element for discovering such logic motifs in GRNs is the searching and analysis of TFBSs [113, 114].

2.5 Representation and visualization of GRNs

Any GRN can be represented as a *directed graph*. This graph consists of nodes, that depict the gene products or the active TFs, and of directed edges, which represent the interactions between TFs and genes. These actions can be either *inducible* (activation) or *inhibitory* (repression). The directed edges are depicted with the same symbols as shown in Tab. 2.1. A basic example is shown in Fig. 2.7, where three TFs inhibit each other in a cycle.

2.6 Synthetic GRNs

While systems biology tries to characterized the multitude of GRNs hidden in the genome of living cells, synthetic biology exploits the technology of recombinant DNA to construct GRNs that do not occur naturally [115], hence the name *synthetic GRN* (SGRN).

Synthetic biologists construct GRNs with natural occurring genes that encode specific TF proteins or to design and construct SGRNs with genes that encode synthetic TFs, such as TALEs [116–119]. These synthetic TFs are usually *chimeric proteins*, which consist of a *DNA binding domain* and a *functional domain*, i.e. the functionality of the TF can be either activation or repression (see Fig. 2.3). The use of natural occurring TFs is highly suitable for constructing relatively small GRNs with simple functionalities. The proved robustness and expression efficiency of such genes make these TFs good candidates for implementation of logic gene circuits. The most prominent

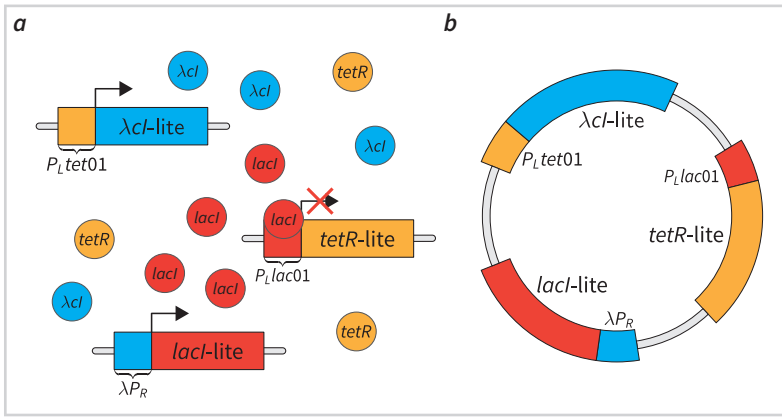


Figure 2.8

The repressilator circuit, a SGRN with an oscillatory response [34]. *a)* A simple schematic of the dynamics of the SGRN. *b)* The repressilator SGRN implemented on the plasmid pSC101.

example of such circuit, built with natural occurring TFs is the *repressilator* network [34], which is depicted in Fig. 2.8.

However, any natural occurring TF used in the designed GRNs, can potentially interfere with the host DNA, and hence their application is usually limited. Another drawback of the utilization of natural TFs is that, there are only few of them that can be used simultaneously in a SGRN. This enormously limits the capabilities of such SGRNs for implementation of logic circuits.

Therefore, in order to construct SGRNs with a large number of switches, a large number of orthogonal TFs[6] is required to prevent possible TFs overlapping (i.e. crosstalk) [120]. This is where the synthetic TFs come to the rescue. By constructing a set of independent and orthogonal TFs, it is possible to broaden the spectrum of gene circuits that can be implemented with SGRNs [7, 120].

2.6.1 *Building blocks of SGRN circuits*

The fundamental building block of any SGRN is the set of transcription factors that is used to implement the genetic switches. Since the number of natural TFs is limited, various synthetic TFs have been developed in the last two decades. The aim is to provide a large set of "easy-to-implement" orthogonal TFs, which could be used to

[6] Orthogonal TFs are here defined as TFs that do not interact between each other (they do not affect the binding or unbinding of other TFs), and do not have any influence to the cell metabolism.

construct any feasible SGRN. Today the most potential candidates for such set are:

- *zinc finger transcription factors* (ZFTFs) [121],
- *microRNA*-TFs [100, 122, 123]
- *transcription activator-like effectors* (TALE) [124, 125] and
- the CRISPR/Cas-9 system [7, 46, 56, 126].

Each of these TFs has its own *pros* and *cons*, but all of them have been used extensively to implement SGRNs with information processing capabilities. Which TF group to choose for a particular design, depends on the design itself [7, 127], however a recent research showed that the effectiveness and orthogonality of TALEs seems to be ahead of the other TFs [7]. Anyway both TALEs and CRISPR/Cas-9 are regarded today as the best choice of orthogonal TFs to construct multi-level logic circuits [7, 126, 128].

TALEs and CRISPR/Cas-9 based TFs can benefit from their strong DNA binding domain fusion with an arbitrary functional domain, such as the *Krüppel associated box repressor* (KRAB) [129] or the powerful activator functional domain VP16 [130]. The main benefit of this fusion is that the binding domain can be the same for both the activator and repressor TFs [7]. This means that the affinity of binding on the DNA site is the same for both activators and repressors[7]. The TALEs TFs are known to bind the TFBSs on the DNA as monomers non-cooperatively. The non-linear response that is required for the implementation of logic gates can be achieved with different modifications of the circuit, such as multiple TFBSs or competitive binding between activators and repressors [7].

In order to simplify the development and the construction of genetic circuits based on SGRNs, a database[8] of already characterized constructs (which includes plasmids, promoters, primers, etc.) was founded in the early days of synthetic biology when the first possibilities for building genetically based computers appeared [133]. This database allows synthetic biologists to quickly find the component of a SGRN that they need, thus improving the effectiveness of the entire design process and also the

[7] As in digital electronics, this is fundamental for designing a circuit with the same type of wires. No circuit designer wants a digital circuit where the signals travel with different speed on different routes, because this can quickly lead to possible malfunction and unexpected switching [131, 132].

[8] Also called *registry of standard parts*. Available at `http://parts.igem.org/Main_Page` [133].

quality of the final SGRN. These components are often referred as *devices*. A hard effort is today invested in synthetic biology towards the standardization of these devices [134, 135].

In order to detect the output signal proteins *in vivo* from a SGRN, synthetic and systems biologists use fluorescence proteins that can be easily detected under the confocal microscope. An example of such protein is the *green fluorescence protein* (GFP), as used for instance in [34] for detecting the oscillations in the repressilator circuit.

2.6.2 Logic gates based on SGRNs

Logic gates are the building blocks of all digital circuits, and they are the foundation of any modern computing platforms. Implementing logic gates with SGRNs then, means to allow digital computation also in the cell.

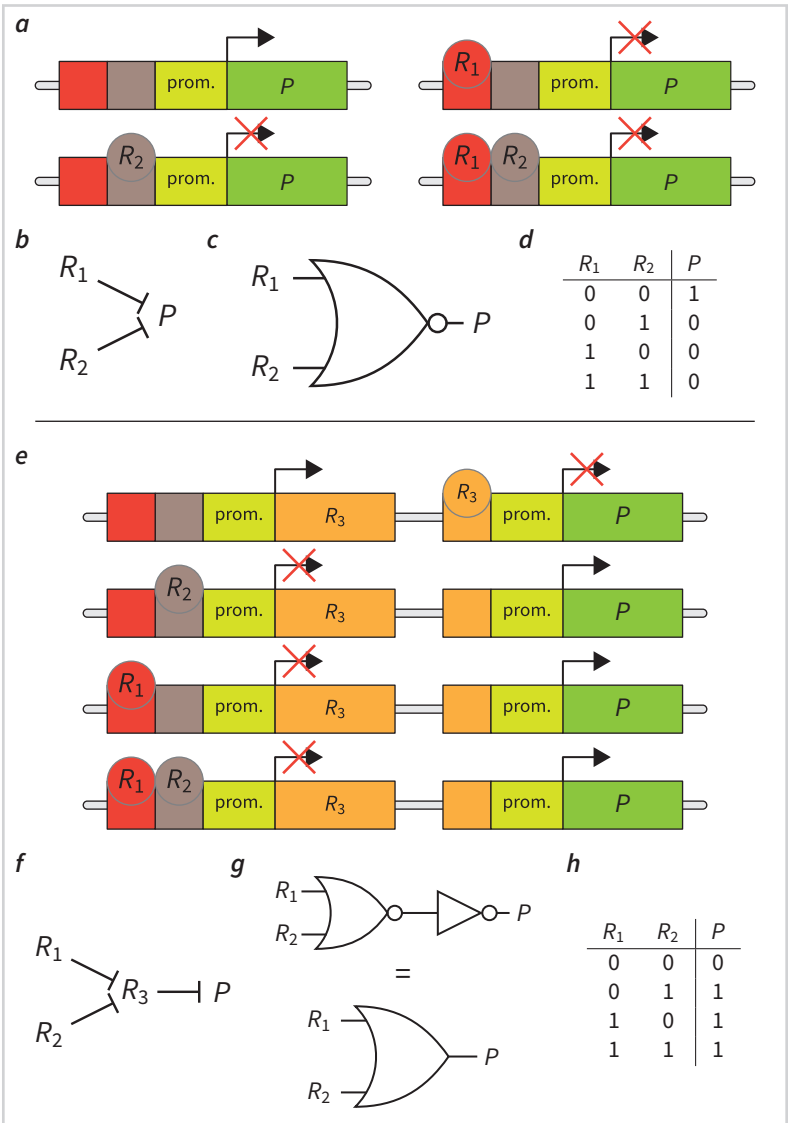
The simplest example is the *NOT gate* (symbol "$\neg$"), which is depicted in Fig. 2.4. The inverter can be implemented with a SGRNs motif that contains only one regulation site, where an input signal, i.e. a repression protein, can inhibit the transcription of the output protein. With a simple extension of the NOT motif, we can construct a two input *NOR gate* (symbol "$\downarrow$"), by adding an additional binding site next to the promoter for the second input signal, see Fig. 2.9 a). We will refer to the group composed by the TF's binding site, the promoter and the output gene as a SGRN construct. From the NOR motif it can be constructed also the *OR gate* (symbol "$\vee$") by adding to the output of the NOR gate an additional inverter as depicted in Fig. 2.9 e), g).

To construct the *AND gate* (symbol "$\wedge$") we can use the same motif of the NOR gate by inverting the two inputs signals, see Fig. 2.10. In a similar way, as depicted for the AND gate, it is possible to construct the *NAND gate* (symbol "$\uparrow$"), or any other basic logic gate. The principal characteristic of these implementations is that all these logic gates are founded from the basic gene repression. In fact, as proven in recent studies [4], any type of logic gates can be implemented from the repression motif.

Once we have available any *functional complete set* (FCS) of operators[9], in this case SGRNs constructs from Fig. 2.9 and Fig. 2.10, we can combine them to implement any logic function. Typical FCS are the following:

$$\{\neg, \vee, \wedge\}, \{\neg, \vee\}, \{\neg, \wedge\}, \{\downarrow\}, \{\uparrow\} \tag{2.1}$$

[9] This is the set of operators with which we can implement any arbitrary logic function.



R_1	R_2	P
0	0	1
0	1	0
1	0	0
1	1	0

R_1	R_2	P
0	0	0
0	1	1
1	0	1
1	1	1

Figure 2.9

Representation of a possible implementation of the NOR and OR gates with SGRNs as proposed in [4, 7]. *a)* The all possible combinations of bound repressors over the regulatory sites in the SGRNs construct that implements the NOR gate. The repressors R_1 and R_2 can bind only on the red and brown binding sites, respectively. Only one bound repressor is needed to stop the transcription of the output protein P. However, the cis-regulatory modules proposed in [4, 7] of all the depicted constructs are composed by multiple non-cooperative TFBSs. *b)* The GRN-graph representing the SGRN of the NOR gate. *c)* The logic schematic of the NOR gate used in electronics. *d)* The truth table of the NOR gate for two input variables. *e)* The combinations of bound repressors over the regulatory sites in the SGRNs constructs that implements the OR gate. The OR gate is implemented via a two input NOR gate with the inverted output. Two logic gates are therefore needed to implement the OR logic. The first construct is a NOR gate, for which the output is used as the input in the second construct, i.e. the inverter. The output of the inverter, i.e. the protein P, is the output of the equivalent OR gate. *f)* The GRN-graph of the SGRN implementing the OR gate. *g)* The logic schematic of the OR gate used in electronics. *h)* The truth table of the OR gate.

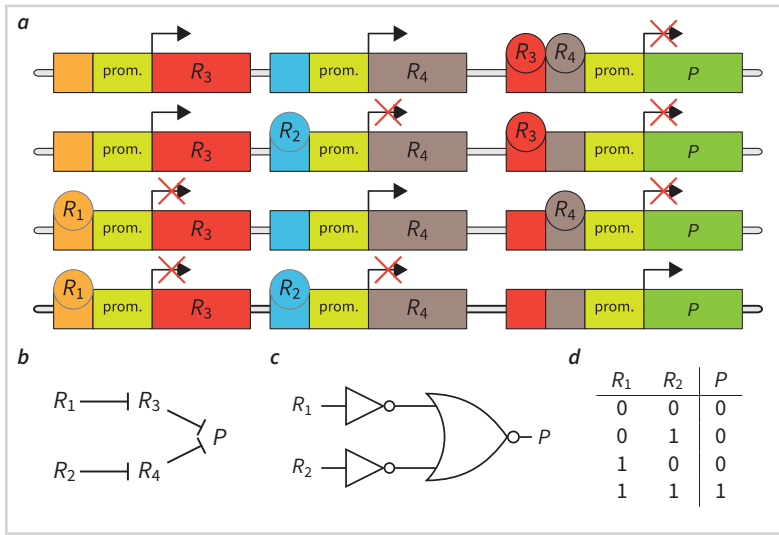


Figure 2.10

A possible implementation of the AND gate with a SGRN as proposed in [7]. *a)* The all possible combinations of bound repressors over the regulatory sites in the SGRNs construct that implements the AND gate. The AND logic is implemented with a NOR gate having both the inputs inverted. *b)* The GRN-graph representing the SGRN of the AND gate. *c)* The logic schematic of the equivalent AND gate. *d)* The truth table of the AND function for two input variables.

Generally the design of logic gates based on SGRNs is a complex task. Many motifs have been designed through years with more or less success. Some of them involved cross-regulation, protein-protein interactions, inductor activation, external light and signalling triggering and even quorum sensing [136–138]. This is today a hot research topic, because logic gates would someday allow implementation of "intelligent" drugs, so any active principle may be produced by SGRNs only on demand and possibly only in the targeted cells. A more complete review of possible information processing structures based on GRNs for basic digital operations can be found in [67] and we refer to [4, 139, 140] for a wider spectrum of possible logic gate designs with SGRNs.

In digital electronics, the switching between logical levels in logic gates needs to happen rapidly and robustly, which means, that the switching response must be nonlinear. A robust nonlinear response in GRNs can be on the other hand achieved with genes regulated by clusters of TFBSs, as a recent research demonstrated [7].

3

Modelling approaches in systems and synthetic biology

Lt. Cmdr. Matt T. Sherman: Sir, Sea Tiger was built to fight. She deserves a better epitaph than "Commissioned 1940, sunk 1941, engagements none, shots fired none." Now, you can't let it go that way. That's like a beautiful woman dying an old maid, if you know what I mean by old maid.

Capt. J.B. Henderson: Did you ever sell used cars?

Lt. Cmdr. Matt T.Sherman: No, Sir.

Capt. J.B. Henderson: I've got a hunch you missed your calling.

Operation Petticoat, 1959

Mathematical and computational modelling coupled with computer simulations are today the two pillars that hold the scientific discovery in almost all the branches of science. There is no scientific field that does not benefit of an *a priori* modelling of the objects of study [141]. Modelling biological systems allows a deeper and a wider view over the dynamics of a particular biochemical process in the cell, and helps to identify the most important properties that govern the behaviour and the development of the cell. In synthetic biology, an *a priori* modelling and simulation of the dynamics of GRNs or SGRNs, is fundamental for a correct design of synthetic gene circuits [67].

Here we present a short introduction to the modelling techniques and simulation algorithms, that are widely used in systems and synthetic biology. We refer to a vast literature that covers this topic, since it is almost impossible to include all the research developments in this field in a short review [71, 142–147].

3.1 Basics of chemical kinetics

Biochemical processes of the cell can be formally described in the same way as any other chemical process: with a system of *chemical reactions*. The natural laws that guide these reactions are the chemical thermodynamic laws. They define exactly when and in which conditions these reactions may occur, based on the thermodynamic properties of the reactants and of the environment. On the other hand, if we want to answer the question: "what is the speed of the reaction?", we have to dive in the field of reaction kinetics [148].

When modelling biological systems, the set of reactions is usually already identified. However the dynamics and kinetics of these reactions often need to be evaluated. Before introducing the topic of chemical reaction kinetics and the modelling techniques, we first present some of the basic chemical formalisms.

Having an environment with volume V, containing N different species, the set of all chemical species is denoted as

$$\mathbf{S} = \{S_1, S_2, \dots S_N\}. \tag{3.1}$$

The preferred measurement unit used for the concentration of chemical species is usually the number of moles per litre, which is also called *molarity* or *molar concentration* and is usually denoted with $M = mol/L$.

A *chemical reaction system* in a closed environment of volume V with M different chemical reactions and N species can be defined as per Def. 3.1[1].

Definition 3.1: A chemical reaction system $\mathbf{R}$ is a set of reactions

$$\mathbf{R} = \left\{R_1, R_2, \dots R_j, \dots R_M\right\}, \tag{3.2}$$

in which the j-th reaction is defined as:

$$R_j = \left\{\sum_{i=1}^{N} m_{ij} S_i \xrightarrow{k_j} \sum_{i=1}^{N} n_{ij} S_i\right\}, \tag{3.3}$$

where k_j is the *reaction rate constant*, and m_{ij} and n_{ij} denote the number of

[1] In the thesis we also used the synonym *chemically reacting system* for referring to this definition.

molecules of the i-th reactant and the number of molecules of the i-th product respectively.

The reaction rate constant is also referred as the *reaction rate coefficient* or simply the *kinetic constant*. The measurement unit of the kinetic constant k depends on the type of reaction, more precisely on the coefficients m_{ij}, by the relation $M^{1-(\sum_{i=1}^{N} m_{ij})}\ s^{-1}$. The m_{ij} and n_{ij} are also known as the *stoichiometric coefficients* and are defined inside the system's input/output matrices:

$$\mathbf{I} = \left[m_{ij}\right]_{M\times N} \quad \text{and} \quad \mathbf{O} = \left[n_{ij}\right]_{M\times N}. \tag{3.4}$$

The reaction R_j from Eq. (3.3) can be defined also with its *stoichiometric vector*.

Definition 3.2: The stoichiometric vector, with length N, of the j-th reaction in the chemical reaction system $\mathbf{R}$ is denoted by ν_j and is defined as follows:

$$\nu_j = \left[n_{1j} - m_{1j}, n_{2j} - m_{2j}, \dots n_{Nj} - m_{Nj}\right]. \tag{3.5}$$

Each element of the stoichiometric vector, which we will denote by ν_{ij}, represents the difference between the two stoichiometric numbers of the i-th species in the j-th reaction. The difference is calculated by subtracting the number of molecules of the species that is consumed in the reaction from the number of molecules of the same species that is produced by the reaction.

Given a system of M different reactions, we can define the *stoichiometric matrix*, denoted by $\mathbf{N}$, as

$$\mathbf{N} = \mathbf{O} - \mathbf{I} = \left[\nu_{ij}\right]_{M\times N}, \tag{3.6}$$

which is a matrix of all the stoichiometric vectors of the chemical reaction system .

3.2 Reaction kinetics in the cell

Given any reaction in the form depicted in Eq. (3.3) we are interested in how fast the reaction can occur, given a specific set of conditions. The answer is represented by the kinetic rate law of the reaction. Let be G_1 and G_2 two reactants, and P the product of reaction R with a kinetic constant k:

$$m_1 G_1 + m_2 G_2 \xrightarrow{k} nP, \tag{3.7}$$

and let be $[G_1]$, $[G_2]$ and $[P]$ their concentrations. Then the *rate* of the reaction R is denoted by v and is defined as

$$v = k\, f\left([G_1], [G_2]\right).$$

The function f is called the *rate function*. Usually the speed is proportional to the species concentrations, i.e.

$$v = k\ [G_1]^{m_1}\, [G_2]^{m_2}.$$

The reaction rate can be generalized for any arbitrary reaction from Eq. (3.3) as

$$v = k \prod_{i=1}^{r} [G_i]^{m_i}. \tag{3.8}$$

By using an infinitesimal notation, we can write the *reaction rate equation* (RRE) as

$$v = \frac{\mathrm{d}\,[P_1]}{\mathrm{d}t} = k\ [G_1]^{m_1}\, [G_2]^{m_2}. \tag{3.9}$$

The RRE from Eq. (3.9) is also known as the *law of mass-action* and essentially is a differential equation that describes how fast the product is produced from the reactants. Moreover it tells us how the concentration of the product changes with time. Eq. (3.9) can be generalized for reactions of the type depicted in Eq. (3.3) as

$$\frac{\mathrm{d}\,[P_i]}{\mathrm{d}t} = k \prod_{i=1}^{r} [G_i]^{m_i}. \tag{3.10}$$

3.3 Reaction kinetics of gene regulatory networks

The dynamics of GRNs, as biological systems, is described through the gene regulation and expression. This dynamics can be conveniently formalized as reaction networks, in which the fundamental reactions of gene regulation and expression take place. GRNs can be viewed as collections of TFs, genes, promoters and binding sites that are interacting (or reacting) together, hence forming a reaction network.

Definition 3.3: The rate-limiting step reactions that can uniquely define the dynamics of a GRN, are of the following types:

1. TF-DNA binding,
2. gene transcription,
3. mRNA translation,
4. protein and mRNA degradation.

Gene regulation can be effectively formalized with reactions of the first type, while the gene expression can be represented with the remaining three types.

3.3.1 Reactions in gene regulation

The reactions describing the binding of one or multiple TFs on one or multiple specific DNA-binding sites, represent the key of the transcriptional regulation in GRNs. The activation and repression of a gene can be systematically formalized with the binding reactions between TFs and the TFBSs. If a TF activator binds to its TFBS close to the promoter region, then the promoter will become "activated" (see chapter 2). The RNAP will hence quickly bind to the promoter and it will initiate the transcription of the gene. Contrary, if a TF repressor binds to the same site, then the promoter will become "repressed" and the gene transcription will not occur. This simple behaviour can be formalized with Def. 3.4.

Definition 3.4: Let be BS_{pr} the cis-regulatory promoter region, i.e. the TFBS of a gene, TF_{act} a TF activator and TF_{rep} a TF repressor. Then the bidirectional reactions

$$\mathrm{BS}_{pr} + TF_{act} \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} \mathrm{BS}^{+}_{pr} \tag{3.11}$$

$$\mathrm{BS}_{pr} + TF_{rep} \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} \mathrm{BS}^{-}_{pr} \tag{3.12}$$

represents the promoter activation BS^{+}_{pr} and the promoter repression BS^{-}_{pr}, respec-

tively. The kinetic constants k_{on} and k_{off} refer to the binding and the unbinding rate of the TF on the DNA binding site, respectively.

Usually in the case of DNA binding, the values k_{on} and k_{off} differ by multiple orders of magnitude. In the case of *E. coli*, the RNAP binding k_{on} is in the range of $10^{10}\ M^{-1}s^{-1}$ [149], while the kinetic rate constant for the unbinding k_{off} is around $2.9\ s^{-1}$ [150]. The affinity of a particular TF to a DNA binding site can be represented by the ratio between k_{on} and k_{off}. Contrary, the ratio between k_{off} and k_{on}, i.e. $\frac{k_{off}}{k_{on}}$ is called the dissociation constant K_d, and it denotes the propensity of the binding TF to dissociate from the DNA.

3.3.2 Reactions in gene expression

A good approximation of the gene expression can be obtained with the separation of the gene transcription and translation as two rate-limiting step reactions.

Transcription

In the previous chapter, we described the gene transcription as a complex biochemical process, that produces a mRNA template from the gene coding sequence. Its dynamics can be approximated with a rate-limiting step reaction.

Definition 3.5: Let be pr^+ an activated promoter as the result of an activator binding in its proximity and let be $RNAP$ a free RNA polymerase enzyme. Then the reaction:

$$pr^+ + RNAP \xrightarrow{k_{trsc}} mRNA + pr^+ + RNAP. \tag{3.13}$$

is the rate-limiting step reaction of the transcription process.

The reaction from Eq. (3.13) depicts a scenario when a free RNAP enzyme binds to the activated promoter pr^+. This initiates the transcription, for which the product is a $mRNA$ molecule, while the $RNAP$ at the end detaches from DNA and leaves the gene coding region. Sometimes it is convenient to assume, that the concentration of RNAP is constant through the entire cell life time. Such assumption implies that the propensity of the transcription becomes conditioned solely by the amount of activated promoters. This leads to a simplification of the reaction in Eq. (3.13), which by

omitting the $RNAP$ becomes

$$pr^{+} \xrightarrow{k_{\text{trsc}}} mRNA + pr^{+}. \tag{3.14}$$

The rate-limiting step reaction in Eq. (3.14), despite having a simple mass action kinetics, conveniently represents the process of transcription. The main shortcoming of this simplification is the kinetic rate k_{trsc} of this reaction. Kinetic rates for rate-limiting step reactions are known to be hard to establish or estimate, and often they are calculated as simple time-delay inverses. In this case the value of k_{trsc}, should take into account all the complex kinetics of transcription. And since the transcription process is composed of three main phases, namely *initiation*, *elongation* and *termination*, it is obvious that the value of k_{trsc} must be dependent on the kinetics of all these three stages.

It is logical then to assume that the entire transcription time can be derived as the sum of all the three stages (often the RNAP-DNA binding can be included as part of initiation). In ideal conditions, this time should be equal for all the same long coding sequences. However we should remember that transcription is a complex biochemical process that is still under research focus, and making an assumption of such ideal conditions would be an oversimplification. In fact, not only all known RNAP enzymes have different binding dynamics to the promoter of eukaryotic and prokaryotic cells, but also can elongate with a variable rate. This is due to the multiple pausing that may keep the polymerase to stall during the formation of mRNA transcripts [151, 152].

Table 3.1

The time-speed ranges for the three transcription phases of the most common eukaryotic and prokaryotic RNAP reported in literature (notation: [nt] nucleotids, [kbp] kilo-base pairs, [bp] base pairs).

	initiation	elongation	termination
RNAP II	0.0216 s^{-1} [151]	0.4 ± 0.08 kb min^{-1} – 4.3 kb min^{-1} [151, 153–155]	0.0016 – 0.0024 s^{-1} [151]
E. coli RNAP	6.2 min – 12.6 h [156]	4.8 nt s^{-1} [157]	0.016 s^{-1} [157] – 30 s^{-1} [158], 2–3 s^{-1} [159]
T7 RNAP	0.36 s^{-1} [150]	43 ± 3.2 nt s^{-1} [150]	5 – 630 s^{-1} [160]

Despite this complexity, there is a large effort in the literature to measure the values

of the kinetic rates of all the three transcription stages, see Tab. 3.1.

By taking into account all these considerations, the authors in [160] proposed to define the reaction's kinetic rate constant, depicting the effective transcription rate k_{trsc} of a gene, as

$$k_{\text{trsc}} = \frac{1}{\frac{1}{k_{init}} + \frac{N_l}{k_{el}} + \frac{1}{k_{term}}}, \tag{3.15}$$

where k_{init} is the kinetic rate of transcription initiation, k_{el} is the elongation speed, N_l is the length in nucleotides of the coding sequence being elongated by the RNAP, and the k_{term} is the kinetic rate of termination.

The Eq. (3.15) allows a representation of the transcription in the form of a rate-limiting step such as the Eq. (3.13) and Eq. (3.14).

Translation

The translation machinery involved in the process, that we explained earlier in section 2.2, can be difficult to model exactly. The translation process is often seen as only an intermediate state in the path from the transcription to the final synthesized product. In this scenario the translation can be defined as per Def. 3.6.

Definition 3.6: Let $mRNA$ be the product of transcription, rB the ribosome and P the final synthesized protein from the translation process. Then the reaction

$$mRNA + rB \xrightarrow{k_{\text{trsl}}} mRNA + rB + P \tag{3.16}$$

represents the rate-limiting step reaction of translation for the mRNA template.

It can be assumed that the total number of ribosome in the cytosol is constant through time. This means that the rB concentration affects the propensity of the translation reaction, but can be included in the reaction rate constant and can be therefore omitted as the $RNAP$ in the transcription reaction. Eq. (3.16) then becomes

$$mRNA \xrightarrow{k_{\text{trsl}}} mRNA + P. \tag{3.17}$$

Sometimes the reaction in Eq. (3.17) can be included directly in the transcription re-

action from Eq. (3.14) in the form

$$pr^{+} \xrightarrow{k_{trsc+trsl}} P + pr. \tag{3.18}$$

The mRNA molecule is subjected to a multitude of transformations during its way from the nucleus to ribosome, such as microRNA interactions and splicing, which in turn make the value $k_{trsc+trsl}$ in Eq. (3.18) hard to estimate accurately.

Using the reaction depicted in Eq. (3.16) as the rate-limiting step reaction for translation, we need to estimate the value of k_{trsl}. The kinetic rate constant k_{trsl}, similarly than k_{trsc} is dependent upon several sub-reactions. In fact the translation process is composed of similar phases as transcription: initiation, elongation and termination. Hence, it should be possible to describe the rate k_{trsl} in the same terms as transcription, such as in Eq. (3.15).

The first modelling proposal of the translation kinetic dates back to the work of Bergmann and Lodish [161], which elucidated the importance of all the three phases of translation, and proposed a simple linear model at steady state. Contrary, a recent study in [162] proposed a non-linear approach, which allowed estimation of the overall translation elongation rate also in terms of the initial association rate. Despite that authors in [163] suggested, that the elongation and initiation are too fast (at least in the *E. coli* K12) to be considered as rate-limiting step reactions for translation, in [164] it was pointed out, that without folding errors, the total translation time can be estimated as the sum of initiation t_{init} (which can be assumed sometimes as the rate-limiting step [165]) and elongation time t_{el}. For long encoding sequence of mRNA it is fair to assume that $t_{el} \gg t_{init}$, so the rate k_{trsl} can be approximated as

$$k_{\text{trsl}} = \frac{1}{t_{el}} = \frac{k_{el}}{N_a}, \tag{3.19}$$

where k_{el} is the total elongation rate of the transcript and N_a is the length in amino acids of the mRNA transcript. For the *E. coli* the value of k_{el} can be in the range between 10 and 20 amino acids/s [164]. We refer to [166] for a thorough review and analysis of translation kinetics.

It is important to highlight that, Eq. (3.13) and Eq. (3.16), from which we derived Eq. (3.14) and Eq. (3.17) respectively, are rate-limiting step reactions, and although they seem to break the conservation laws, because of the presence of an extra product

(mRNA in the case of transcription and P in the case of translation), they are perfectly valid, if we assume to omit all the additional reactants involved in both transcription and translation, which are present in the reaction environment in constant amounts and are therefore non interesting to the external observer. These additional reactants, such as the free amino acids that provides the nucleotides during the formation of the mRNA strand in the transcription process, ensure the compliance with the conservation laws in both the limiting-step reactions depicted in Eq. (3.13) and Eq. (3.16).

Degradation

Degradation is a physical phenomenon in which the concentration exponentially decreases with time. In cell biology, the degradation of organic compounds is often increased by co-factors and enzymes, which prevent a single substance to be present in excessive concentrations. The intracellular degradation process occurs for every chemical compound. Specific enzymes called *proteases* are responsible for fast degradation of large macromolecules, such as proteins. They bind to the target macromolecule and facilitate the break down of the protein's amino acid chain. In GRNs the two species that are more susceptible to degradation are proteins (including TFs) and mRNAs [167].

Definition 3.7: The degradation of a chemical species S can be defined as

$$S \xrightarrow{k_{\text{deg}}} \varnothing. \tag{3.20}$$

Note that the symbol $\varnothing$ denotes the *null* product.

Without losing generality we can expand the definition of degradation also for proteins. Let P_0 be the initial concentration of a protein (or mRNA) in the cell and let denote with P the concentration of the same protein at a particular time t. By applying a simple mass-action kinetic to the reaction above, the degradation can be described with the following differential equation

$$\frac{\mathrm{d}P}{\mathrm{d}t} = -k_{\text{deg}} P, \tag{3.21}$$

Table 3.2

The time ranges for degradation half-life of the majority of proteins and mRNAs found in the most common hosts studied in systems biology (table adapted from values listed in [164]). Note, that there are some proteins and mRNAs found especially in yeast and human cells, that have higher upper limit values (see [164, 168]).

	E. coli	*S. cerevisiae*	*H. sapiens*
mRNA	1 – 15 (min)	5 – 60 (min)	1 – 30 (h)
proteins	20 – 60 (min)	0 – 3 (h)	0 – 70 (h)

where k_{deg} is the kinetic constant of the degradation. By rearranging and integrating both sides of the Eq. (3.21), we obtain:

$$\int_{P_0}^{P} \frac{\mathrm{d}P}{P} = -k_{\text{deg}} \int_0^t \mathrm{d}t, \tag{3.22}$$

and finally

$$\ln\left(\frac{P}{P_0}\right) = -k_{\text{deg}} t. \tag{3.23}$$

From the Eq. (3.23) we can express the solution for $[P]$ and for time t, as

$$P = P_0 \mathrm{e}^{-k_{\text{deg}} t}, \tag{3.24}$$

$$t = \frac{\ln(P_0) - \ln(P)}{k_{\text{deg}}}. \tag{3.25}$$

With Eq. (3.25) we can evaluate the half-life time $t_{\frac{1}{2}}$, which is the time that needs to elapse for the initial concentration to be halved, i.e. $P = \frac{P_0}{2}$:

$$t_{\frac{1}{2}} = \frac{\ln(P_0) - \ln\left(\frac{P_0}{2}\right)}{k_{\text{deg}}} = \frac{\ln(2)}{k_{\text{deg}}}. \tag{3.26}$$

Half-life is much easier to obtain than the DNA binding kinetic constants for TFs. Moreover, by knowing the value of half-life we can obtain the kinetic constant k_{deg} from Eq. (3.26). Typical values for mRNA and protein half-life are listed in Tab. 3.2.

3.3.3 Reaction rate's dependencies

The kinetic rate constants of all the reactions described in this section are usually hard to obtain. Specific experimental techniques can measure the value of these rates in particular environments, but unfortunately they are not always accurate. Alternative approaches include

1. the exploitation of parameter estimation techniques, that can help to overcome the lack of data, or
2. the decomposition of the kinetic rate constants as functions of other "more handleable" parameters, that can be determined effortlessly.

Physical chemistry is a branch of chemistry that aims to solve the latter. In fact several (if not all) kinetic constants can be written as a function of sub-parameters, such as the diffusion coefficient in cell cytoplasm, the molecular mobility, the temperature of the environment, the UV radiation intensity, etc. In the light of these relations, it is sometimes convenient to define the kinetic rates of GRNs reaction networks as composite parameters according to Def. 3.8.

Definition 3.8: A kinetic rate constant, or any reaction-related parameter which is not a species concentration, is called a *composite parameter*, if it can be decomposed either analytically or numerically as an expression of sub-parameters, that can be directly measured or estimate experimentally.

The view of a kinetic rate constant as a composite parameter can be found useful in the case, where the constant is unknown and the sub-parameters can be instead measured experimentally. An example of a composite parameter is the transcription rate k_{trsc}, which is defined in Eq. (3.15).

3.4 Modelling approaches and simulation techniques

One of the main aims of *in silico* modelling and simulation is to determine the response of a reaction network or system, given a set of observed chemical reactions and a set of initial parameters' values. The first step is to formalise the GRNs as reaction networks, according to Def. 3.9.

Definition 3.9: Let's observe the cell reactive environment, which can be approximated as a homogeneous mixture of different molecules, with fixed volume V. Let be M the number of different reactions (or reaction channels) in the system, which are denoted by the column vector $\mathbf{R} = [R_1, \dots R_M]^\top$, with their kinetic constants $\Theta = [k_1, \dots k_M]^\top$. These reactions have N different chemical species, denoted by the vector $\mathbf{S} = [S_1, \dots S_N]$. The value $x_i(t)$ defines the concentration (or sometimes the number of molecules, i.e. the population) of the species S_i at time t. The *state vector*

$$\mathbf{X}(t) = [x_1(t), \dots x_N(t)] \tag{3.27}$$

describes the current state of the system at time t, while the initial state can be denoted by $\mathbf{X}(t_0) = \mathbf{x}_0$. Often it is convenient to rewrite the notation $x_i(t)$ simply as x_i and the vector $\mathbf{X}(t)$ as $\mathbf{x}$ or simply as x.

The reactions $\mathbf{R}$ and the species $\mathbf{S}$ are linked together via the stoichiometric matrix, which is denoted by $\mathbf{N} = [\nu_{ij}]$, of size $M \times N$. The vector

$$\nu_j = \left[\nu_{1j}, \dots, \nu_{Nj}\right] \tag{3.28}$$

is called the *state transition vector* or also *state-change vector*, because it defines the changes in concentrations for all the chemical species involved in the reaction j. The execution of reaction R_j changes the system to the new state $\mathbf{X} + \nu_j$.

The second step consists in choosing the simulation method for deriving the response of the system dynamics instead. One can choose between deterministic, stochastic or multi-scale approaches.

3.4.1 Deterministic approaches

Deterministic approaches provide an average response of the population of each species of the reaction network. Such description is usually obtained with a set of *ordinary differential equations* (ODEs). Any GRN can be represented by a system of ODEs, since the reaction network can be described in terms of mass action kinetics, see Ex. 3.1.

Example 3.1: Let be a reaction network composed by the following two reactions:

$$R_1: \quad S_1 + S_2 \xrightarrow{k_1} P_1,$$
$$R_2: \quad P_1 \xrightarrow{k_2} \varnothing.$$

with the initial concentrations $[S_1] = 2\,\mu M$, $[S_2] = 2\,\mu M$ and the kinetic constants $k_1 = 10\,M^{-1}s^{-1}$ and $k_2 = 2.3\,s^{-1}$. From Def. 3.9 it follows:

$$\mathbf{R} = \begin{bmatrix} R_1 \\ R_2 \end{bmatrix}, \qquad \mathbf{S} = [S_1 \;\; S_2 \;\; P_1],$$
$$\mathbf{K} = \begin{bmatrix} 10 \\ 2.3 \end{bmatrix}, \qquad \mathbf{X}_0 = \begin{bmatrix} 2 \cdot 10^{-6} & 2 \cdot 10^{-6} & 0 \end{bmatrix},$$

$$\mathbf{N} = \begin{bmatrix} -1 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}.$$

The dynamics of this reaction network can be fully described by the following system of ODEs:

$$\frac{\mathrm{d}P_1}{\mathrm{d}t} = k_1 S_1 S_2 - k_2 P_1,$$
$$\frac{\mathrm{d}S_1}{\mathrm{d}t} = -k_1 S_1 S_2,$$
$$\frac{\mathrm{d}S_2}{\mathrm{d}t} = -k_1 S_1 S_2.$$

Systems of ODEs can be solved analytically only for simpler biochemical systems. For reaction networks of large GRNs, numerical integration is usually preferred.

In systems and synthetic biology, ODEs are usually used for accurate modelling of the dynamics of entire populations of cells exhibiting the same pattern of species concentrations. The simulation carried by a numerical solution of ODEs, represents the average response in the population.

The main shortcoming of deterministic approaches is their lack to handle efficiently

the noise that is inherently present in every biological system [88]. As such, the cellular dynamics is often regarded as noisy. If the noise has a large influence on the cellular response, then it should be included in the model definition. In order to simulate the response at a molecular level, other approaches should be used instead of simple ODEs. Stochastic methods are usually preferred in such cases.

3.4.2 *Stochastic approaches*

Stochastic modelling is a prominent computational approach to analyse dynamical systems, which are highly affected by noise and for which the system's response cannot be analytically derived. Stochastic chemical kinetics allows to take into account such constraints implicitly in the model definition.

Stochastic chemical kinetics is based on the Def. 3.9. Let $\Theta = \{\mathbf{S}, \mathbf{X}, \mathbf{R}\}$ be a system from the definition Def. 3.9 and let $\mathbf{X}(t) = x$ describe the current state of the system at the time t. Each reaction R_j of the vector $\mathbf{R}$ can be linked with a specific *propensity function*, which is denoted by $a_j(x)$. The value $a_j(x)\mathrm{d}t$ is the probability, that the reaction R_j will occur in the next infinitesimal time interval $[t, t + \mathrm{d}t)$. The propensity function can be calculated according to the type of reaction:

- for monomolecular reactions it holds

$$S_i \xrightarrow{k_j} \dots \qquad a_j(x) = k_j x_i, \tag{3.29}$$

- for bimolecular reactions

$$S_i + S_{i'} \xrightarrow{k_j} \dots \quad a_j(x) = \frac{k_j}{\Omega} x_i x_{i'}, \qquad i \neq i' \tag{3.30}$$

$$a_j(x) = \frac{k_j}{\Omega}(x_i - 1)x_i, \qquad i = i' \tag{3.31}$$

- and generally we can extend these rules with the following equation [67, 169]:

$$a_j(\mathbf{X}(t)) = \frac{k_j}{\Omega^{n_2 - 1}} \prod_{i=1}^{n_1} \frac{x_i(t)!}{(x_i(t) - l_i)!}, \tag{3.32}$$

where n_1 is the number of different chemical reactants in the reaction R_j, n_2 is the total number of consumed molecules in the reaction R_j, l_i is the number of consumed

molecules for the i-th species in the reaction R_j and Ω is the volume of the environment solution defined as $\Omega = V \cdot N_A$, where N_A is the Avogadro constant.

Chemical Master Equation

The probability that the system will be in a particular state at a specific time can be formalized as a Markov jump process

$$\begin{aligned} P(x, t + \mathrm{d}t|x_0, t_0) \; = & P(x, t|x_0, t_0)\left(1 - \sum_{j=1}^{M} a_j(x)\mathrm{d}t\right) + \\ & \sum_{j=1}^{M} P(x - \nu_j, t|x_0, t_0)a_j(x - \nu_j)\mathrm{d}t, \end{aligned} \tag{3.33}$$

where $P(x, t|x_0, t_0)$ denotes the probability of the system to be in state x at time t, with t_0 and x_0 representing the initial time and state. $P(x, t + \mathrm{d}t|x_0, t_0)$ denotes the probability that the system will be in state x in the next infinitesimal time $t + \mathrm{d}t$.

Rearranging the Eq. (3.33) and limiting the differential time $\mathrm{d}t \to 0$ we obtain

$$\frac{\mathrm{d}P(x, t|x_0, t_0)}{\mathrm{d}t} \; = \; \sum_{j=1}^{M} \Big[P(x - \nu_j, t|x_0, t_0)a_j(x - \nu_j) - P(x, t|x_0, t_0)a_j(x)\Big], \tag{3.34}$$

which is the formal notation of the *chemical master equation* (CME) [88]. Note that the reaction rate equation from Eq. (3.10) can be derived from Eq. (3.34) by simply assuming that there are no molecular fluctuation in the reaction environment, i.e. making the stochastic reacting system deterministic [88]. More details about the formulation of CME can be found in [170, 171].

The CME is hard to solve in practice. The problem is, that we cannot express the probabilities $P(x, t|x_0, t_0)$ in terms of concentrations. Moreover, the number of possible states, in which the system could be at a determined time, is enormous and hence also the number of (stochastic) differential equations cannot be efficiently solved.

The solution of the CME can be obtained computationally with the *stochastic simulation algorithm* (SSA), which can precisely simulate (and not only approximate) the sequence of reactions that occur in the system, and hence equivalently depict the same dynamics described with the CME.

Stochastic simulation algorithm

SSA is a widely used method for simulating the dynamics of a well-stirred system of molecules [172]. Let's denote the probability, that the reaction R_j will take place in the system at the time interval $[t+\tau, t+\tau+\mathrm{d}\tau)$ with $p(\tau, j|x, t)\mathrm{d}\tau$. This probability is associated to the joint event of two random variables

- the time τ elapsing between two reactions and
- the index of the next reaction j.

We can write the probability $p(\tau, j|x, t)\mathrm{d}\tau$ as the product of probabilities of these two random variables

$$p(\tau, j|x, t)\mathrm{d}\tau = P(\tau|x, t)P(j|\tau, x, t), \tag{3.35}$$

and the same holds for their *probability density functions* (PDFs)

$$p(\tau, j|x, t) = p(\tau|x, t)p(j|\tau, x, t). \tag{3.36}$$

The PDF $p(j|\tau, x, t)$ represents the scenario when the j-th reaction will occur among a total of M different reactions and it can be calculated as the ratio between the j-th propensity function $a_j(x)$ and the cumulative propensity function, namely $a_0(x)$. Shortly

$$a_0(x) = \sum_{j'=1}^{M} a_{j'}(x), \tag{3.37}$$

$$p(j|\tau, x, t) = \frac{a_j(x)}{a_0(x)}. \tag{3.38}$$

On the other hand, $p(\tau|x, t)$ is more difficult to estimate, since the distribution of the time τ is unknown. We can evaluate the probability $p(\tau, j|x, t)\mathrm{d}\tau$ from Eq. (3.36) in a different way. $P(\tau|x, t)$ refers to the probability that the next reaction will occur in exactly τ. This is equal to the probability $P_0(\tau|x, t)$ that no reaction will occur before τ. We can therefore calculate the probability $p(\tau, j|x, t)\mathrm{d}\tau$ as the probability that no reaction will take place till the time $t+\tau$, times the probability that exactly one reaction R_j will occur in the infinitesimal time interval $[t+\tau, t+\tau+\mathrm{d}t]$. The latter

probability is formally represented by the propensity value $a_j(x)\mathrm{d}\tau$

$$p(\tau, j|x, t)\mathrm{d}\tau = P_0(\tau|x, t)a_j(x)\mathrm{d}\tau. \tag{3.39}$$

The only unknown term in the right side of Eq. (3.39) is $P_0(\tau|x, t)$. One way to evaluate this probability is by extending the event to an infinitesimal time $\mathrm{d}\tau$. The probability that no reaction will occur in $\tau + \mathrm{d}\tau$ is a joint probability between the case that no reaction occurs till time τ, i.e. $P_0(\tau|x, t)$ and that no reaction will occur during the infinitesimal time $\mathrm{d}\tau$. This can be formalized as

$$P_0(\tau + d\tau|x, t) = P_0(\tau|x, t)\left[1 - \sum_{j'=1}^{M} a_{j'}(x)\mathrm{d}\tau\right]. \tag{3.40}$$

The first term on the right in Eq. (3.40) is the same as in Eq. (3.39), while the second term stands for the probability that no reactions will occur during the infinitesimal time $\mathrm{d}\tau$. Considering that $a_0(x) = \sum_{j'=1}^{M} a_{j'}(x)$, the Eq. (3.40) can be rewritten as

$$P_0(\tau + d\tau|x, t) - P_0(\tau|x, t) = -P_0(\tau|x, t)a_0(x)\mathrm{d}\tau,$$

which we can rearrange furtherer as

$$\frac{P_0(\tau + d\tau|x, t) - P_0(\tau|x, t)}{\mathrm{d}\tau} = -P_0(\tau|x, t)a_0(x), \tag{3.41}$$

and assuming that $d\tau \to 0$, we can write Eq. (3.41) as a simple first order differential equation using the basic law of infinitesimal calculus

$$\dot{f} = \lim_{\triangle t \to 0} \frac{f(t + \triangle t) - f(t)}{\triangle t},$$

which bring us to the form

$$\lim_{\mathrm{d}\tau \to 0} \frac{P_0(\tau + \mathrm{d}\tau|x, t) - P_0(\tau|x, t)}{\mathrm{d}\tau} = -P_0(\tau|x, t)a_0(x),$$

or equivalently

$$\dot{P}_0(\tau|x, t) = -P_0(\tau|x, t)a_0(x). \tag{3.42}$$

By rearranging and integrating both sides of Eq. (3.42) we get

$$\int_0^\tau \frac{\dot{P}_0(\tau|x,t)}{P_0(\tau|x,t)}\mathrm{d}\tau = -a_0(x)\int_0^\tau \mathrm{d}\tau.$$

Solving the integrals leads us to

$$\ln\left(P_0(\tau|x,t)\right) = -a_0(x)\tau,$$

from which $P_0(\tau|x,t)$ can be expressed as

$$P_0(\tau|x,t) = \mathrm{e}^{-a_0(x)\tau}. \tag{3.43}$$

Now, if we put the definition of $P_0(\tau|x,t)$ in the Eq. (3.39), we get

$$p(\tau,j|x,t) = a_j(x)\mathrm{e}^{-a_0(x)\tau}. \tag{3.44}$$

The equation Eq. (3.44) is the foundation for stochastic simulation of the reacting system. If we combine the Eq. (3.44) with Eq. (3.38) in Eq. (3.36), then we get

$$p(\tau|x,t) = a_0(x)\mathrm{e}^{-a_0(x)\tau}. \tag{3.45}$$

The dynamics of the system can be simulated, by firing the j-th reaction at each time interval τ. Since we have defined the PDFs for both of the two random variables τ and j, through Eq. (3.45) and Eq. (3.38), we can now approximate these values by using the *inverse generating method*, which is one of the fundamental tools of the *Monte Carlo simulation* approach [172]. The time τ in which the next reaction will occur in the system can be inferred knowing its PDF, i.e.

$$P(\tau \leq t|x,t) = \int_{-\infty}^{t} p(\tau|x,t)\mathrm{d}\tau = \mathrm{e}^{-a_0(x)t}. \tag{3.46}$$

By replacing t with τ in Eq. (3.46) or by simply reversing Eq. (3.43), the time τ can be expressed as

$$\tau = \frac{1}{a_0(x)}\ln\left(\frac{1}{r_1}\right). \tag{3.47}$$

Similarly the index of the next reaction can be determined by using a selection process

that resembles the *roulette wheel selection*:

$$\min_{j} \left\{ \sum_{k=1}^{j} a_k(x) \geq r_2 a_0(x) \right\}. \tag{3.48}$$

Here r_1 and r_2 are two random numbers uniformly distributed in the interval $(0, 1)$. Eq. (3.47) and Eq. (3.48) represent the *Monte Carlo* step in the stochastic simulation algorithm.

Algorithm 3.1

The *stochastic simulation algorithm* (SSA).

Input: ▪ a model from Def. 3.9.

Output: ▪ the model responses x over time t.

procedure SSA

1. Determine all the propensities values $a_j(x)$ and their cumulative sum $a_0(x)$.
2. Monte Carlo step: compute τ and the index j from Eq. (3.47) and Eq. (3.48).
3. Increase the time step $t \leftarrow t + \tau$ and update the system state $x \leftarrow x + \nu_j$.
4. Record the current pair (x, t) and go back to step 1 if $t < t_{max}$.

end procedure

The Alg. 3.1 is known as the Gillespie's *stochastic simulation algorithm* (SSA), or the Gillespie's *direct method* (DM) [172, 173]. The main advantage of the SSA is that, the average of multiple runs of the simulation, will provide a convergence to the analytical solution of the CME [67, 88, 172, 173]. However, the SSA is known to be computationally expensive. The computational cost of the algorithm depicted in Alg. 3.1 can be tied to these two main factors:

- the determination of the propensities $a_j(x)$ and
- the computation of the index j.

Both of these costs are linearly dependent on the number of reactions (hence $\sim \mathcal{O}(M)$) [172]. By using optimized data structures for storing species and propensities, we can

decrease the overall time complexity of the SSA [174].

Variants and improvements of the SSA

The SSA can be used to model any arbitrary reactions, however, for larger systems, it should be rather replaced by one of its improvements, since the DM implementation can be computationally expensive. However, it is still widely used in systems and synthetic biology, due to its simplicity and reliability. SSA has been greatly studied and tuned since its original formulation in 1976 [175]. Several variants have been developed, such as the

- next reaction method (NRM) [176],
- optimized direct method (ODM) [177],
- sorting direct method (SDM) [178],
- logarithmic direct method (LDM) [174],
- partial propensities direct method (PDM) [179] and
- composition rejection (SSA-CR) [180].

The first reaction method (FRM) is also considered as an alternative improvement, despite being developed along with the DM [175]. The majority of these methods mainly aims to improve the computational performance of the simulation algorithm, by integrating either reduction assumptions, or improved data structures, without loosing generality or quality of the simulations. Authors in [181] provide a short review as well as an efficient benchmark study of the algorithms listed above.

3.4.3 Multi-scale approaches

The multi-scale approaches refer globally to the modelling techniques that deal with models that are composed of multiple sub-systems. Their dynamics usually occur on different scales: spatial, temporal, or both. In chemically reacting systems some reactions occur faster than others. If this difference in speed spans at least one order of magnitude, then the system is considered to have a *multiple time scale* dynamics[2]. On

[2] The terms "multi-scale" and "multiple-time scale" have been used in literature often to represent different concepts. We refer to [182] for a general overview of terminology and definition of the multi-scale concept.

the other hand, if a model contains sub-systems that differ in size by several orders of magnitude, then the system is told to be defined in *spatially* different scales. For instance, modelling an entire organism (such as the human body) will require to differentiate the system in various sub-groups, such as organs, tissues and cells. In such scenario the sub-systems' dynamics can profoundly differ from each other, because of the different spatial scales in which the functionalities of the organism are defined.

GRNs, that are described with reactions defined in Def. 3.3, are an example of chemically reacting systems that exhibit multiple time scale reactions. For instance, the TFs binding reactions are known to be much faster than rate-limiting step reactions of transcription and translation. Their occurrence may be described in a different time scale.

The difference in reaction speeds is not the only multi-scale aspect of GRNs. The populations of species can differ by several orders of magnitude. For example, promoters are structurally a part of the DNA. If we consider the genome of the cell, then its promoters will be probably present only in one specific region of the genome. On the other hand, free TFs can have population of way over 10^5 (see Fig. 2.1)[3]. Combined with the multiple time scale of reaction networks, a large population difference between species, create the so-called *multi-scale problem* of chemically reacting systems [95].

A similar notion of spatial multi-scale model is compartmentalization. The cell reacting environment is usually divided in smaller sub-spaces (or sub-volumes) in which particular reactions take place. Transcription and translation rate-limiting step reactions in eukaryotes are two typical examples. The former occurs inside the nucleus, while the latter occurs in the cytosol. The nucleus and cytosol can be spatially segregated sub-systems, with only TFs and mRNA as common species. The transition between these two "compartments" can be simply modelled with reaction delays [183]. However, in most practical uses, the compartmentalization of the cytosol and nucleus can be omitted. We refer to the review in [184, 185] for a detailed summary of other spatial multi-scale approaches in systems biology.

Hereafter we will mostly focus in the multiple time scale dynamics, by introducing the most used multi-time-scale techniques that are able to tackle the multi-scale problem of chemically reacting systems.

[3] In SGRNs however, promoters are usually located on a plasmid vector in multiple copy numbers and hence their population would be identical to the population of the vector in the cell times the promoter copy number.

Stiffness in chemically reacting systems

Despite the simplicity of the SSA, any implementation of the Alg. 3.1 will perform slowly in multiple time scale systems. Such scenario can be described with the following condition:

Condition 3.1: A chemically reacting system from 3.9 can contain a combination of two types of reactions, which can be either exceptionally slow (occurring less frequently or sporadically) or exceptionally fast (occurring more frequently or all the time).

If Cond. 3.1 is met, then the SSA will suffer by the constant firing of the "fast" reactions (the reactions having a high kinetic constant value, i.e. higher speed, will have by definition a higher propensity function and therefore a higher probability to be fired), making the simulation extremely slow. This multi-scale problem is known as *stiffness* [94].

In order to allow a mathematical sounded representation of the stiffness occurring in GRNs, we need to separate reactions among fast and slow [186, 187]. A short redefinition of the system defined in Def. 3.9 is therefore needed.

Consider that the set of reactions $\mathbf{R}$ can be always divided in two subsets, the set of *fast reactions* R^f and the set of *slow reactions* R^s, i.e.

$$R^f = \left\{R_1^f, R_2^f, \dots R_{M_f}^f\right\}, \tag{3.49a}$$

$$R^s = \left\{R_1^s, R_2^s, \dots R_{M_s}^s\right\}. \tag{3.49b}$$

Being $\mathbf{R} = \left\{R^f \cup R^s\right\}$ the union of disjoint reactions sets R^f and R^s, for which $M = M^f + M^s$. The criterion for creating these two subsets is usually heuristic, however the easier way to separate the reactions in fast and slow is by discriminating the value of their propensity function.

Similarly, the set of species can be also partitioned into two disjoint subsets, i.e.

$$S^f = \left\{S_1^f, S_2^f, \dots S_{N_f}^f\right\}, \tag{3.50a}$$

$$S^s = \left\{S_1^s, S_2^s, \dots S_{N_s}^s\right\}. \tag{3.50b}$$

A *fast species* is defined as any species which gets changed by at least one fast reaction. A *slow species* on the other hand is defined as the species which gets changed by solely slow reactions. Here, one would observe an interesting asymmetry: a slow species cannot get changed by a fast reaction, whether a fast species can get changed by a slow reaction [94]. The total number of different species in the system are therefore

$$N = N_f + N_s. \tag{3.51}$$

The state vector takes the form

$$X(t) = \left(X^f(t), X^s(t)\right), \tag{3.52}$$

or more compactly $x = \left(x^f, x^s\right)$, where $X^f(t)$ and $X^s(t)$ denote the fast and the slow process respectively. Consequently, the stoichiometric matrix and the state transition vectors need to be redefined for each reaction and species partition, as

$$\nu_j^f = \left(\nu_{1j}^{ff}, \nu_{2j}^{ff}, \dots \nu_{N_f j}^{ff}\right), \qquad j = 1, \dots, M_f, \tag{3.53a}$$

$$\nu_j^s = \left(\nu_{1j}^{fs}, \nu_{2j}^{fs}, \dots \nu_{N_f j}^{fs}, \ \nu_{1j}^{ss}, \nu_{2j}^{ss}, \dots \nu_{N_s j}^{ss}\right), \qquad j = 1, \dots, M_s, \tag{3.53b}$$

where

- ν_{ij}^{ff} denotes the change of the i-th fast species in the j-th fast reaction R_j^f,
- ν_{ij}^{fs} denotes the change of the i-th fast species in the j-th slow reaction R_j^s and
- ν_{ij}^{ss} denotes the change of the i-th slow species in the j-th slow reaction R_j^s.

Note that by definition ν_{ij}^{sf} (the change of the i-th slow species in the j fast reaction) is zero, and hence the vector ν_j^f can be shorten to the size of N_f.

The corresponding propensity functions can be defined as functions of both partitioned subsets of species and reactions, i.e.

$$\begin{aligned} a_j^f(x) &= a_j^f\left(x^f, x^s\right) \quad j = 1, \dots, M_f, \\ a_j^s(x) &= a_j^s\left(x^f, x^s\right) \quad j = 1, \dots, M_s. \end{aligned} \tag{3.54}$$

Having redefined the basic structures from Def. 3.9, we can now define a stiff chemically reacting system limited by the multiple time scale constraints.

Definition 3.10: A well stirred chemically reacting system, subjected to Cond. 3.1 can be formalized as an extension of Def. 3.9, in which the reaction channels, the species, the state vector, the state transition vectors and the propensity functions are partitioned according to Eq. (3.49), Eq. (3.50), Eq. (3.52), Eq. (3.53) and Eq. (3.54), respectively.

Approximation techniques

Several multi-scale computational approaches have been developed, which can be efficiently applied to the multi-scale models. The most representative computational methods are the *slow-scale stochastic simulation algorithm* (ssSSA), the *multi-scale stochastic simulation algorithm* (MSSA) and the *nested stochastic simulation algorithm* (nSSA), which are described thoroughly in Appendix (see section A.1). Another approach called the *implicit tau-leaping*, described first in [188] as a solution of the "explicit" tau-leaping method, is also known as a viable approach to model multi-scale reacting systems. We will omit here a detail description and refer to the original paper [188] for more details about its definition and implementation.

3.5 Common issues in modern approaches

The stochastic simulation algorithm (SSA) [172] and its improvements, have become popular for modelling natural and synthetic GRNs in both the fields of systems and synthetic biology. Stochastic simulations are able to account for heterogeneity [189] and the intrinsic noise, inherent in the chemically reacting systems.

Multi-scale approximation techniques can be used to simulate the behaviour of stiff chemically reacting systems. However, there are scenarios in which the existing methods are not able to describe efficiently the gene regulation dynamics of the biological

system under study. Consider for instance a GRN or a SGRN that contains promoters with clusters of multiple non-cooperative TFBSs, such as one depicted in Fig. 2.5. According to the reaction network definition in Def. 3.9, such system may be complicated to formalize. The reason is, that it is unclear how to represent the activated and the repressed promoters. Def. 3.9 does not provide any criterion or rule on how to define a promoter with multiple bound TFs in its cis-regulatory module. If we consider all the possible promoter's states as a specific species, then we will have to deal with a combinatorial explosion of states. It is obvious then, that the Def. 3.9 should be enlarged and generalized to include and allow also this complex scenario.

Current literature is rather poor in the formalization of such GRNs, however there were some efforts to fill this gap, although only for few specific models, as investigated in the work of Giorgetti et al. and Lengyel et al. [105, 190]. In both works, authors considered only deterministic simulation approaches. In this dissertation we wanted to address this problematic and in chapter 4 we provide a solution for this type of GRNs, by proposing a multi-scale stochastic algorithm, able to precisely simulate the dynamics of complex cis-regulatory modules containing multiple non-cooperative TFBSs.

4

Modelling GRNs with complex cis-regulatory modules

Lt. Cmdr. Matt T. Sherman: Molumphry, will this boat go down?
Chief Molumphry: Like a rock sir.
Lt. Cmdr. Matt T. Sherman: Mr. Watson, how are the plates?
Lt. Watson: Tight as a drum sir.
Lt. Cmdr. Matt T. Sherman: And the engines, Tostin?
Chief Mechanic's Mate Sam Tostin: Factory fresh, sir.
Lt. Cmdr. Matt T. Sherman: Well how about it?
Capt. J.B. Henderson: I say take your thieves and these liars here and get the hell out. Oh there's one stipulation, you'll engage no enemy shipping and that includes lifeboats. Even if you see one of them swimming in the water, avoid him. He might kick a hole in your side. Good luck Matt.

Operation Petticoat, 1959

In nature, it is possible to come across several GRNs containing complex cis-regulatory modules. Such modules are often composed of multiple TFBSs that regulate the gene expression in complex ways, as we briefly reviewed in section 2.3.3. An example of such cis-regulatory module can be found in the Epstein-Barr virus (EBV) [191]. It allows competitive TF binding that results in a robust gene switching mechanism. Multiple non-cooperative TFBSs are also found in SGRNs [6, 7, 42, 91]. In recent years it has been shown that such systems reflect a robust behaviour. A high number of TFBSs for the same promoter can provide a more stable and robust gene regulation. Such property is currently exploited through novel synthetic DNA binding domains, in the design process of complex gene circuits with information processing capabilities

[4, 6, 7]. Modelling approaches for such systems have already been proposed [105, 191], although mainly based on deterministic thermodynamics.

In this chapter we explore the possibility to describe GRNs containing clusters of cis-regulatory elements by the aid of the multi-scale stochastic modelling, which allows one to perform highly accurate simulations at molecular level and, at the same time, manage efficiently the chemical stiffness of these GRNs. Hereafter we present a variant of SSA for the accurate modelling of GRNs containing promoters with multiple non-cooperative TFBSs. With a relative small computational complexity, the adapted algorithm is able to simulate efficiently the stiff chemically reacting system of these GRNs.

4.1 Chemically reacting systems of complex cis-regulatory modules

Describing a chemically reacting systems, e.g. a GRN, in which there is at least one promoter that is being regulated by multiple TFBSs, can pose a challenging modelling task. The reason is, that the functional state of a promoter, (free, active or inhibited) is defined indirectly by bound TFs. If multiple TFs are bound close to the promoter, then it depends on the activation (or inhibition) rules (described in section 2.3.3) to decide whether the promoter will be active or inhibited. But modelling such decision requires to deal with a high number of permutations of bound activation and repressor TFs. In fact the number of possible promoter's binding sites states increases exponentially with the number of binding sites, and hence it can be extremely hard to construct an accurate model for such systems. Moreover, if the cis-regulatory modules, containing these multiple binding sites, are present in extremely low concentration (low copy number), then a high level of stiffness may occur, especially for those reactions concerning these TFBSs, i.e. the TFs binding and unbinding reactions.

In the following section we will provide a more detailed formalization of a stochastic simulation algorithm capable to tackle both of these issues.

4.1.1 Binding and unbinding reactions

Let's presume a promoter being regulated by n consecutive (clustered) TFBSs as depicted in Fig. 2.5. The reactions of TF binding and unbinding are

$$\begin{aligned} b_i + TF &\xrightarrow{k_{\text{on}}} b_i^{\text{TF}}, \\ b_i^{\text{TF}} &\xrightarrow{k_{\text{off}}} b_i + TF, \end{aligned} \tag{4.1}$$

where b_i and b_i^{TF} denote an empty and a occupied binding site.

If two types of TFs can bind competitively to the same binding site, e.g. activators T_a and repressors T_r, then the reactions that change each binding site's states can be described per Eq. (3.11) and Eq. (3.12), as

$$\begin{aligned} b_i + T_a &\xrightarrow{k_{\text{on}}} b_i^{+}, \\ b_i^{+} &\xrightarrow{k_{\text{off}}} b_i + T_a, \\ b_i + T_r &\xrightarrow{k_{\text{on}}} b_i^{-}, \\ b_i^{-} &\xrightarrow{k_{\text{off}}} b_i + T_r, \end{aligned} \tag{4.2}$$

where b_i refers to the i-th non-occupied binding site, b_i^{+} refers to the same binding site on which an activator T_a is bound and b_i^{-} denotes a binding site on which a repressor T_r is bound.

The number of reactions of Eq. (4.1) and Eq. (4.2) increases drastically with the number of binding sites. Let's denote with k, the number of different types of TFs, that can bind to n consecutive binding sites. The total number of possible promoter states is then

$$(k+1)^n. \tag{4.3}$$

The total number of all the possible binding and unbinding reactions increases consequently. With k different TF types that can bind to n available binding sites,

$$\sum_{i=1}^{n} \binom{n}{i} i\, k^{n-i+1} \tag{4.4}$$

different binding reactions can occur.

4.1.2 A model for multiple binding sites representation

Consider a cis-regulatory module of a promoter that is composed of 10 TFBSs on which two TFs can bind competitively. Consider also that such promoter is used in a SGRN that we want to simulate through the classic SSA. In such scenario there would be

$$\sum_{i=1}^{10} \binom{10}{i} \cdot i \cdot 2^{11-i} = 393,660 \tag{4.5}$$

possible combinations of binding and unbinding reactions. It is clear that, if one would choose the SSA (or any other of its variants) to simulate the system, the model would have to describe 393,660 different reactions for TF-promoter interactions only. Considering that the binding and unbinding reactions between DNA and TFs can be presumed to be faster than transcription and translation, one can simply adopt the ssSSA approach proposed by Cao et al. [94, 95], which suggests to threat the virtual fast system containing all the fast reactions, as a stochastic birth-dead process. However, because of the exponential number of TF's binding and unbinding reactions, such solution (or any other solution which would require to represent analytically the fast system of binding reactions as a finite Markov process) would be difficult to implement, at least for the systems with a large number of reactions as per Eq. (4.5).

In order to make the system in question manageable, we propose to handle all the promoters states dynamically, without the need to store statically all the $(k+1)^n$ promoter species. By denoting with M_p the number of all the different promoters in the GRN, we can construct a time dependent *binding site matrix* $\mathbf{B}_m(t)$ for each of the promoters with multiple TFBSs. The matrix $\mathbf{B}_m(t)$ has dimensions $c_m \cdot n_m$, where c_m is the copy number of the m-th promoter in the GRN and n_m is the number of TFBSs in the cis-regulatory module of the same promoter, whereas $m = 1, \dots, M_p$. Each element of $\mathbf{B}_m(t)$ represents one binding site and is defined as

$$b_{i,j}^m(t) = \begin{cases} 0; & \text{if the binding site } \textit{is empty} \text{ at the time } t, \\ 1; & \text{if the binding site is occupied by an activator at the time } t, \\ -1; & \text{if the binding site is occupied by a repressor at the time } t, \end{cases} \tag{4.6}$$

where $b_{i,j}^m(t)$ (also written simply as b_{ij}^m or b_{ij}) denotes the state of the j-th binding site on the i-th copy of the m-th promoter at time t. We will also use the notation $b_{i,j}^{m+}(t)$

and $b^{m-}_{i,j}(t)$ (or simply b^{+}_{ij} and b^{-}_{ij}) for referring to a binding site occupied by an activator and a repressor respectively.

Matrix $\mathbf{B}_m(t)$ represents the entire DNA binding space for all the copies of the m-th promoter inside the GRN at time t. In order to account for all the m different promoters of the GRN, M_p matrices $\mathbf{B}_m$ needs to be defined. The matrix $\mathbf{B}_m$ can be also conveniently represented as a column vector of rows $\mathbf{b}_i, i = 1, \dots, c_m$ as

$$\mathbf{B}_m = \left[\mathbf{b}^m_1, \mathbf{b}^m_2, \dots, \mathbf{b}^m_{c_m}\right]^\top, \tag{4.7}$$

where each row $\mathbf{b}_i$ is defined as

$$\mathbf{b}_i = \left[b^m_{i,1}, b^m_{i,2}, \dots, b^m_{i,c_m}\right]. \tag{4.8}$$

With the aid of the matrix $\mathbf{B}_m$, one can simply retrieve the state of any promoter in the GRN. This allows a more compact management of the TFs' binding reactions. One can simply fire a binding or unbinding reaction by changing the state of a binding site b_{ij} directly in the matrix $\mathbf{B}_m$. Instead of dealing with an exponential sized data structures, as would be the stoichiometric matrix in such a scenario, we only need to evolve M_p different matrices of size $c_m \cdot n_m$. Obviously there would be cases in which the value c_m will be high[1], however, that size will be still incomparably smaller than the stoichiometric matrix that could be used instead.

4.1.3 *Activation and inhibition rules*

Such representation allows us to effectively represent all the states of the promoter through time. However, different binding sites configurations needs to be classified among three possible states:

- idle (or free/empty),
- active or
- repressed (or inhibited),

[1] e.g. $c_m \approx 1000$ would be the number of plasmids, that can be ingested and maintain by a mammalian cell, which means that 1000 promoters of the same type will be present in the GRN (under the assumption that on each plasmid there is only one copy of such promoter). In such scenario the matrix $\mathbf{B}_m$ will have a size of $1000 \cdot n_m$

which describe the actual activity of the promoters. This is determined by specific rules that take into account the number of bound repressors and activators in the cis-regulatory module, as we shortly reviewed in section 2.3.3. Examples in [105] are

- all-or-none,
- singular and
- additive.

It is important to clarify, that while authors in [105] describe these rules more as an interaction behaviour between the bound TFs and the RNAP, by attributing to each scenario from 2.6, the corresponding level of promoter activation, a slightly different interpretation should be considered for stochastic simulations. The bound TFs configuration defines the overall logic state of the promoter instead of a specific level of activation (or inhibition).

The rules that determine if a specific promoter, i.e. a specific row of the binding site matrix, is activated or inhibited, may be based, in a case of competitive TFs, on a simple *majority* rule, i.e. simply by counting the number of bound activators and repressors.

Such rules can be defined specifically for each type of promoters. For example, if the experimental findings show that some repressor TFs have a stronger effect than the activators, this would suggest that the rule governing the promoter activation may be based on whether just one repressor is bound on a TFBS. Another realistic rule may also consider the fact, that bound TFs on those binding sites that are closer to the promoter (i.e. their position in the rows of matrices $\mathbf{B}_m$ matters) may have a stronger effect on the promoter activity than the other TFs bound far away from the promoter. Bound activators that are close to the promoter may in fact facilitate the recruiting of the RNAP. Such scenarios should be properly investigated, since they would provide detailed rules to be used for determining the exact activation state of each promoter.

To find out how many promoters of the m-th type are activated and how many are inhibited in the observed GRN, one can then simply apply the activation (or the inhibition) rules, for each of the promoter states, i.e. for each row of the matrix $\mathbf{B}_m$. This can be done with the time dependent *activation matrix* $\mathbf{A}(t)$, which contains the activation state of every promoter. The matrix $\mathbf{A}(t)$ can be defined as a function of

$\mathbf{B}_m$ as

$$\mathbf{A}_m(t) = h\big(\mathbf{B}_m(t)\big) = h\left(\left[\mathbf{b}_1^m,\ \mathbf{b}_2^m,\ \dots\ ,\ \mathbf{b}_{c_m}^m\right]\right), \tag{4.9}$$

where $h()$ is the *rule function*. It computes the state of each row $\mathbf{b}_i^m$ of the matrix $\mathbf{B}_m$, by setting the respective row of the matrix $\mathbf{A}_m$ to a value $a_i^m(t)$ (also written as $a_i(t)$ or simply a_i), defined as

$$a_i^m(t) = \begin{cases} 0; & \text{if } \mathbf{b}_i^m \text{ encodes an idle state at the time } t, \\ 1; & \text{if } \mathbf{b}_i^m \text{ encodes an active state at the time } t, \\ -1; & \text{if } \mathbf{b}_i^m \text{ encodes an inhibited state at the time } t. \end{cases} \tag{4.10}$$

By counting the frequencies of each state in the matrix $\mathbf{A}_m$, it is then straightforward to set the global concentrations for the species representing each of the three possible promoter states.

We illustrate this approach in the following example.

Example 4.1: Let's presume a promoter pr with two binding sites (b_1 b_2), on which an activator A and a repressor R can bind competitively. The promoter can be thus free (pr), inhibited (pr_R) or activated (pr_A). Let presume a copy of the promoter on 4 different plasmids. All four promoters are activated according to a simple majority rule, i.e. if the number of bound activators is higher than the number of bound repressors, then the promoter will be activated. In a case of an equal amount of both TFs, repressors will prevail. Given any possible configuration of the binding sites matrix $\mathbf{B}(t)$, we can compute the current status of the activation matrix $\mathbf{A}(t)$, e.g.

$$\mathbf{B}(t) = \begin{bmatrix} 0 & 0 \\ -1 & 1 \\ 1 & 0 \\ -1 & -1 \end{bmatrix} \rightarrow \mathbf{A}(t) = \begin{bmatrix} 0 \\ -1 \\ 1 \\ -1 \end{bmatrix}.$$

By counting the frequencies of each state: idle (f_0), active (f_1) and inhibited (f_{-1}) in the matrix $\mathbf{A}$, we can then set the global concentration of the promoter species at the current time t: $\left[pr\right] = f_0 = 1$, $\left[pr_A\right] = f_1 = 1$ and $\left[pr_R\right] = f_{-1} = 2$.

4.1.4 *Time evolution of TF binding*

The simulation of the promoters regulation can be performed by simplifying the reactions that describe the activators and repressors binding, i.e. by interpreting the promoter binding reactions from Def. 3.4, as simple binding site reactions similarly as depicted in Eq. (4.2). This simulation can be performed with a nested SSA subroutine, inside the main stochastic simulation (which can also be performed with a SSA and is used to simulate the time evolution of non-TF related, binding and unbinding, reactions). Binding and unbinding reactions can be straightforwardly translated to the binding site matrix $\mathbf{B}_m$ for any free, activated and repressed binding site (b_{ij}, b_{ij}^{+} and b_{ij}^{-} respectively). The binding site matrix $\mathbf{B}_m$ is therefore used inside the inner SSA instance, similarly as the nested SSA subroutines in the nSSA (see Alg. A.3 in chapter A). When simulating a binding reaction in the inner SSA, a free binding site is randomly chosen from the matrix $\mathbf{B}_m$ and its value is changed according to Eq. (4.6), i.e. the chosen matrix element is set from 0 to 1 for activator binding and to -1 for repressor binding. On the other hand, if an unbinding reaction is simulated, an occupied site is randomly chosen from the matrix $\mathbf{B}_m$ and its value is changed to 0, meaning that the binding site is being released. At each step of the outer SSA, the configuration of the promoter is evaluated. The promoter is activated, if the TFs' occupancy configuration in its cis-regulatory module represents an activated configuration, according to the activation rule of the promoter.

To make the things more clear, we illustrate the time evolution for the matrix $\mathbf{B}_m$ in the example Ex. 4.2.

Example 4.2: Let's presume the same scenario as in 4.1, i.e.

$$\mathbf{B}(t) = \begin{bmatrix} 0 & 0 \\ -1 & 1 \\ 1 & 0 \\ -1 & -1 \end{bmatrix} \rightarrow \mathbf{A}(t) = \begin{bmatrix} 0 \\ -1 \\ 1 \\ -1 \end{bmatrix}.$$

Now, we can drive the time evolution of binding and unbinding reactions by simply taking into account these two types of reactions

$$BS + A \rightleftharpoons BS_A \quad \text{and} \quad BS + R \rightleftharpoons BS_R$$

where the BS denotes the count of free binding sites (the number of 0's), BS_A denotes the count of activated binding sites (the number of 1's) and BS_R denotes the count of inhibited binding sites (the number of -1's) in $\mathbf{B}$. These values are used to calculate the propensities of each reaction. Further on, if for example, a repressor binding reaction

$$BS + R \longrightarrow BS_R \tag{4.11}$$

is chosen by the SSA, the matrix $\mathbf{B}(t)$ will evolve, by randomly picking a free binding site (for instance the second element in the 3-th row of $\mathbf{B}$) and by changing its value to -1. This will result in a new state of the matrix

$$\mathbf{B}(t+1) = \begin{bmatrix} 0 & 0 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix}.$$

By applying again the majority rule over all the rows of the matrix $\mathbf{B}(t+1)$, we can compute the activation matrix $\mathbf{A}(t+1)$, i.e. the amount of activated and inhibited promoters in the GRN at the time $t+1$, i.e.

$$\mathbf{B}(t+1) = \begin{bmatrix} 0 & 0 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix} \rightarrow \mathbf{A}(t+1) = \begin{bmatrix} 0 \\ -1 \\ -1 \\ -1 \end{bmatrix}.$$

By counting the frequencies of each state in the matrix $\mathbf{A}(t+1)$ we can set the global concentrations of the promoter species at time $t+1$

$$[pr] = 1, \qquad [pr_A] = 0, \qquad [pr_R] = 3.$$

With such approach it is possible to simulate any arbitrary GRN and SGRN despite the complexity of the promoters' cis-regulatory modules. Hereafter we propose a stochastic simulation algorithm that successfully deals with such scenarios, by includ-

ing the time evolution of the matrices $\mathbf{B}_m$ and $\mathbf{A}_m$ in a multi-scale SSA algorithm, similar to the nSSA.

4.2 A dynamic multi-scale stochastic simulation algorithm

In order to apply a stochastic SSA-based algorithm that would include the same time evolution of the promoters with multiple TFBSs, as we described in the previous section, we have to provide a more strict definition of the system that we aim to simulate.

Definition 4.1: Let Ω be a chemically reacting system of a GRN, containing M_p different promoters with multiple TFBSs, in which only the reaction from Def. 3.3 takes place. The types of reactions that control the gene expression in Ω are

$$\mathrm{pr}_m^+ \longrightarrow \mathrm{mRNA}_m + \mathrm{pr}_m^+, \tag{4.12a}$$

$$\mathrm{mRNA}_m \longrightarrow \mathrm{mRNA}_m + \mathrm{P}_m, \tag{4.12b}$$

$$\mathrm{mRNA}_m \longrightarrow \varnothing, \tag{4.12c}$$

$$\mathrm{P}_m \longrightarrow \varnothing, \tag{4.12d}$$

where mRNA_m is the gene product expressed by the activated promoter pr_m^+, and P_m refers to the group of translation products of the m-th gene, with $m = 1, \dots, M_p$. On the other hand, let the following reaction types represent the gene regulation

$$\mathrm{pr}_m + \mathrm{T_A} \rightleftharpoons \mathrm{pr}_m^+, \tag{4.13a}$$

$$\mathrm{pr}_m + \mathrm{T_R} \rightleftharpoons \mathrm{pr}_m^-, \tag{4.13b}$$

where pr_m is the m-th promoter of the system, pr_m^+ and pr_m^- are its activated and inhibited state, $\mathrm{T_A}$ and $\mathrm{T_R}$ are general TF activators and repressors respectively, capable to bind to any of the m-th promoter's TFBSs (where $m = 1, \dots, M_p$).

Without loosing generality we can say that this is a viable representation of the key reactions in GRNs, which would allow one to simulate its dynamics accurately.

Now, suppose the system Ω is affected by stiffness, where the reactions occur in different time scales, as per Cond. 3.1. Assume also that the reactions in Eq. (4.12) of transcription (Eq. (3.13)), translation (Eq. (3.16)) and degradation (Eq. (3.20)) are slow compared to the reactions defined in Eq. (4.13) of TFs binding and unbinding (as defined in 3.4). We can thus partition the set of reactions in two different sets: the slow and the fast. We propose the following arrangements, in order for Def. 3.10 to be able to represent also the scenario in which the promoters contain complex cis-regulatory modules with multiple non-cooperative TFBSs.

Definition 4.2: The Def. 3.10 is modified to be capable of dealing with a chemically reacting system as depicted in Def. 4.1 by including the following arrangements:

1. Let c_m be the copy number and n_m the number of binding sites of each of the promoters in the system, $m = 1, \dots, M_p$.

2. Reactions can be partitioned according to Eq. (4.12) and Eq. (4.13).

3. Let $\mathscr{P}^s$ and $\mathscr{P}^f$ be the sets of the promoter species used in the slow and in the fast reactions, i.e.

$$\begin{aligned} \mathscr{P}^s &= \left\{ pr_1^{s+}, \dots, pr_{M_p}^{s+} \right\}, \\ \mathscr{P}^f &= \left\{ pr_1^f, pr_1^{f+}, pr_1^{f-}, \dots, pr_{M_p}^f, pr_{M_p}^{f+}, pr_{M_p}^{f-} \right\}, \end{aligned} \tag{4.14}$$

   where pr_i^{s+} denotes the i-th active promoter, involved in the slow reaction of transcription, and $pr_i^f, pr_i^{f+}, pr_i^{f-}$ denote the i-th idle, active and inhibited promoter respectively, involved in the fast reactions of TF binding and unbinding, where $i = 1, \dots, M_p$.

4. Let $\mathscr{T}^f$ and $\mathscr{T}^s$ denote the sets of the TFs involved in the fast and in the slow reactions respectively

$$\mathscr{T}^s = \left\{ T_1^s, \dots, T_{N_s^{\mathrm{TF}}}^s \right\} \quad \text{and} \quad \mathscr{T}^f = \left\{ T_1^f, \dots, T_{N_f^{\mathrm{TF}}}^f \right\}, \tag{4.15}$$

where N_s^{TF} and N_f^{TF} denote the number of different TFs involved in the slow and in the fast reactions respectively[a].

5. The slow and the fast species vectors can be redefined as

$$S^s = \left\{\mathscr{P}^s, \mathscr{T}^s, \mathscr{U}^s\right\} \quad \text{and} \quad S^f = \left\{\mathscr{P}^f, \mathscr{T}^f\right\}, \tag{4.16}$$

where $\mathscr{U}^s$ denotes the slow species, that are neither TFs, nor promoters (e.g. mRNAs), where $|S^s| = N^s$ and $|S^f| = N^f$. Hence the state vector from Eq. (3.52) is composed by the fast and slow species amounts accordingly.

6. Let $\mathscr{B}$ be the set with cardinality M_p of all the binding site matrices $\mathbf{B}_m$, i.e.

$$\mathscr{B} = \left\{\mathbf{B}_1, \ \ldots \ \mathbf{B}_{M_p}\right\}. \tag{4.17}$$

[a] Usually $\mathscr{T}^f = \mathscr{T}^s$, since TFs are involved in both the gene expression and regulation reactions.

4.2.1 *A simulation for the fast system*

The basic approach to simulate the dynamics of multi-scale chemically reacting systems, as in the nSSA, is to perform two SSAs, one for the fast and one for the slow reactions of the system. The slow reactions are performed in the main (outer) SSA, while the fast reactions are performed as an SSA subroutine inside the main SSA routine. Now the two reactions sets share a subset of common species which need to be updated before both SSAs can use them, i.e. these are the components in the system $X^s(t)$ from Eq. (3.52) that are changed by the stoichiometric vector's elements ν_{ij}^{fs} defined in Eq. (3.53). These species are in the case of a GRN defined as per Def. 4.1, the TFs lying in the intersection

$$\mathscr{T}_{s\cap f} = \mathscr{T}^s \cap \mathscr{T}^f. \tag{4.18}$$

Apart from $\mathscr{T}_{s\cap f}$, the two scales also share the active promoters species pr_m^{s+} of the set $\mathscr{P}^s$, for which their amount needs to be computed, as we shown in the Ex. 4.2 and Ex. 4.1. This can be done by computing the frequencies of appearance of each promoter state (active, repressed or idle) in the activation matrix $\mathbf{A}$ for all the M_p

different promoters in the GRN

$$pr_m^s = f_0\Big(\mathbf{A}_m(t)\Big), \tag{4.19a}$$
$$pr_m^{s+} = f_1\Big(\mathbf{A}_m(t)\Big), \tag{4.19b}$$
$$pr_m^{s-} = f_{-1}\Big(\mathbf{A}_m(t)\Big), \tag{4.19c}$$

where f_0, f_1 and f_{-1} denote the total number of idle, activated and repressed promoters, respectively, and where $m = 1, \dots, M_p$.

We can adapt the SSA from Alg. 3.1 to perform the fast reactions from Eq. (4.13) having multiple TFBSs. Let's have a system as per Def. 3.10 with the arrangements listed in Def. 4.2. The system state is defined as

$$X(t) = \Big(X^f(t),\ X^s(t)\Big), \tag{4.20a}$$
$$X^f(t) = \Big(X_{\mathscr{P}^f},\ X_{\mathscr{T}^f}\Big), \tag{4.20b}$$
$$X^s(t) = \Big(X_{\mathscr{P}^s},\ X_{\mathscr{T}^s},\ X_{\mathscr{Y}^s}\Big), \tag{4.20c}$$

where the stoichiometric vectors can be defined accordingly (see Eq. (3.53)). Therefore, the SSA can evolve the system $X^f(t)$. The only difference is that the reactions need to occur over the matrix $\mathbf{B}_m$, instead on the species S^f directly. This can be done efficiently by following the proposed approach in the previous section, i.e. by applying the fast reactions from Eq. (4.13) directly on $\mathbf{B}_m$ and by describing the concentration of each fast promoter species (pr_m^f) as

$$\Big[pr_m^f\Big] = \Big|\big\{b_{ij}^m\big\}\Big|, \tag{4.21a}$$
$$\Big[pr_m^{f+}\Big] = \Big|\big\{b_{ij}^{m+}\big\}\Big|, \tag{4.21b}$$
$$\Big[pr_m^{f-}\Big] = \Big|\big\{b_{ij}^{m-}\big\}\Big|, \tag{4.21c}$$

where b_{ij}^m, b_{ij}^{m+} and b_{ij}^{m-} denote the elements of the matrix $\mathbf{B}_m$ as in Eq. (4.6) and where the notation $|\,|$ denotes their cardinality. These values are used to calculate the propensities of four reactions that are observed at this point, i.e. binding and unbinding of activator, and binding and unbinding of repressor.

If a TF binding reaction is chosen by the SSA, it is performed by randomly choosing an empty site in $\mathbf{B}$ and by placing the value b_{ij}^{m+} or b_{ij}^{m+} (1 or −1) according to the type of the binding. A TF unbinding reaction is instead performed by randomly selecting a bound TF value, anywhere in $\mathbf{B}_m$ and placing the empty value 0 on the chosen site.

The algorithm therefore performs the time evolution of all the binding site matrices $\mathbf{B}_m$. We call this algorithm the *binding sites evolution SSA* (bseSSA).

Algorithm 4.1

The binding sites evolution stochastic simulation algorithm for performing the fast reactions of TF-DNA binding and unbinding.

Input:
- a stochastic system X^f from Eq. (4.20b) for the reactions of Eq. (4.13),
- the set of matrices $\mathscr{B}$ with cardinality M_p.

Output:
- the evolution of the system X^f.

procedure BSESSA($X^f(0)$, $\mathscr{B}$)

Set the initial time value.

$t_f = 0$

Quantify the number of activated, repressed and empty promoters as per Eq. (4.21)

for all $m = 1, \dots, M_p$ *do*

$\mathbf{B}_m = \mathscr{B}(m)$

$X^f_{pr^f_m}(0) = \left|\{b_{ij}^m\}\right|$

$X^f_{pr^{f+}_m}(0) = \left|\{b_{ij}^{m+}\}\right|$

$X^f_{pr^{f-}_m}(0) = \left|\{b_{ij}^{m-}\}\right|$

end for

Perform the time evolution over all the matrices $\mathbf{B}_m$

while $t_f < T^f_{\text{MAX}}$ *do*

Perform one SSA step:

1) select the j-th reaction to be fired in the system $X^f(t), j = 1, \dots, M_f$

2) compute τ_f

3) get the index m of the promoter for which the j-th reaction will be fired

$[j, \tau_f]$ = SSA_step($X^f(t_f)$)

Identify which promoter is affected, i.e. retrieve the index m of the promoter involved in the j-th reaction.

m = getPromoterIndex(j)

Evolve the binding site matrices $\mathbf{B}_m(t_f)$
if (j = "an index of an activator binding reaction") *then*
Select a random empty site b^m in $\mathbf{B}_m(t_f)$
and change its value to 1 (as per Eq. (4.6))
$b^m = 1$
else if (j = "an index of an activator unbinding reaction") *then*
Select a random site b^{m+} occupied by an activator in $\mathbf{B}_m(t_f)$
and change its value to 0
$b^{m+} = 0$
else if (j = "an index of a repressor binding reaction") *then*
Select a random empty site b^m in $\mathbf{B}_m(t_f)$ and change its value to -1
$b^m = -1$
else if (j = "an index of a repressor unbinding reaction") *then*
Select a random site b^{m-} occupied by a repressor in $\mathbf{B}_m(t_f)$
and change its value to 0
$b^{m-} = 0$
else
No time evolution of $\mathbf{B}_m(t_f)$ required for the reaction j.
end if

Perform the state change on $X^f(t_f)$ according to the relative ν_j^f
$X^f(t_f + \tau_f) = X^f(t_f) + \nu_j^f$
Increase the time step
$t_f = t_f + \tau_f$
end while
end procedure

The main question, that arises here is, how long the time evolution of the binding site matrices should be simulated, i.e. how big should T^f_{MAX} be. This question is tightly related to the slow-scale approximation in Def. A.1. However, in contrast with the virtual fast process $\hat{X}^f$ of the ssSSA (see section A.1.1), the process X^f in the bseSSA is strictly defined for the TF-DNA binding reactions only, and hence there could be much more manoeuvring space for the definition of T^f_{MAX}. A possible value for T^f_{MAX} can be defined heuristically or as per condition used for the nSSA [192]

$$T^f_{\text{MAX}} \approx \Delta_s \ll \bar{t}_s, \tag{4.22}$$

where $\bar{t}_s$ is the average expected time to the next slow reaction to occur and Δ_s is the time increment, for which the product $\overline{a_j^s}(x^s; x^f)\Delta_s$ is the probability, that a slow scale reaction R_j^s will occur in the next time interval $[t, t + \Delta_s)$.

4.2.2 *Combining slow and fast scale simulations*

The algorithm, that performs

- the SSA simulation over X^s and
- the bseSSA over the fast system X^f,

is called the *dynamic multi-scale stochastic simulation algorithm* (DMSSA) and it is depicted in Alg. 4.2. The DMSSA simulates the slow system X^s, where at each time step τ_s, a bseSSA procedure is called in order to retrieve the states of every promoter in the GRN.

Algorithm 4.2

The dynamic multi-scale stochastic simulation algorithm.

Input:
- Ω (a model from 3.10) with the arrangements from Def. 4.2,
- $\mathscr{H}$ (the set of all the promoters' activation rules).

Output:
- the system state evolution X.

procedure DynamicMultiscaleSSA(Ω, $\mathscr{H}$)

Given the values c_m and n_m, construct the set $\mathscr{B}$ containing all the matrices $\mathbf{B}_m$, for each $m = 1, \ldots, M_p$

$\mathscr{B} = \left\{\mathbf{B}_1, \mathbf{B}_2, \ldots \mathbf{B}_{M_p}\right\}$

Initialize the system

$X^f = \left\{X_{\mathscr{P}^f}, X_{\mathscr{T}^f}\right\}$

$X^s = \left\{X_{\mathscr{P}^s}, X_{\mathscr{T}^s}, X_{\mathscr{Y}^s}\right\}$

$X = \left\{X^f, X^s\right\}$

Set the initial time

$t_s = 0$

while $t_s < T^s_{\text{MAX}}$ *do*

Update the TFs species used in the slow-scale to the fast process X^f

$X^f_{\mathscr{T}_{s\cap f}} = X^s_{\mathscr{T}_{s\cap f}}$

Time evolve the fast process $X^f(t)$

$X^f(t_s + T^f_{\text{MAX}}) = bseSSA\big(X^f(t_s), \mathscr{B}\big)$
Compute the activation matrices $\mathbf{A}_m$ by applying the activation rule on the matrices $\mathbf{B}_m$, and update the slow promoter species with the according frequencies, as per Eq. (4.19)
for all $(m = 1, \dots, M_p)$ *do*
$\mathbf{A}_m(t_s) = h_m\big(\mathbf{B}_m(t_s)\big)$
$pr^{s+}_m = f_1\big(\mathbf{A}_m(t_s)\big)$
end for
Update the TFs species used in the fast-scale to the slow process X^s
$X^s_{\mathscr{T}_{s\cap f}} = X^f_{\mathscr{T}_{s\cap f}}$
Perform one step of the SSA for the slow reactions,
i.e. compute Eq. (3.47) and Eq. (3.48).
$\big[j, \tau_s\big] = SSA\big(X^s(t_s)\big)$
Apply the change and increase the time step t_s
$X^s(t_s + \tau_s) = X^s(t_s) + \nu^s_j$
$t_s = t_s + \tau_s$
end while
end procedure

If $n_m = 1 (\forall m = 1, \dots, M_p)$, all the matrices $\mathbf{B}_m$ reduce to a column vector, in which case the exponential number of binding reactions reduces to two single reactions, as per Eq. (4.13). In such scenario, the Alg. 4.2 reduces to a classic SSA. In this case it would be computationally more convenient to adopt one of the multi-scale approaches, such as the ssSSA, MSSA or the nSSA. The algorithm 4.2 can manage the explosion of the number of promoters' states by simply constructing the binding sites matrices, one for each promoter type of the GRN. Moreover, the algorithm from 4.2 allows the application of custom activation rules for the evaluation of the promoter functional states.

4.2.3 Computational validation

To illustrate the performance enhancement of the DMSSA, as well as to validate its correctness, we propose a simple model of a SGRN, containing a promoter with multiple non-cooperative TFBSs. The cis-regulatory module of this promoter, namely *pr*, contains 2 TFBSs ($n = 2$) on which either a repressor R or an activator A can bind. For the purpose of the validation, we will assume, that both the activator and the repressor species are present in the reacting volume in constant amounts. The cis-regulatory

module of the promoter pr is driving the regulation of a gene, for which the product is a protein P. The slow reactions of the system are defined as

$$pr^{+} \xrightarrow{k_{\text{trsc}}} mRNA + pr^{+}, \tag{4.23a}$$

$$mRNA \xrightarrow{k_{\text{trsl}}} mRNA + P, \tag{4.23b}$$

$$mRNA \xrightarrow{k_{\text{deg-mRNA}}} \varnothing, \tag{4.23c}$$

$$P \xrightarrow{k_{\text{deg}}} \varnothing, \tag{4.23d}$$

while the fast reactions are represented with the two bidirectional reactions

$$pr + A \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr^{+}, \tag{4.24a}$$

$$pr + R \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr^{-}. \tag{4.24b}$$

The TF binding to the promoter's pr cis-regulatory module is of competitive nature. Both A and R compete to bind over the same binding sites. We assume that both TFs have equivalent binding and unbinding kinetic constants. Let's assume also that the promoter is governed by a majority rule, but with the difference, that a bound repressor has a stronger influence than the activator, hence such rule can be defined as

$$h\left(\mathbf{b}_m\right) = \begin{cases} 1; & \text{if } |b^{m+}| > |b^{m-}|, \\ -1; & \text{if } |b^{m+}| \leq |b^{m-}|. \end{cases} \tag{4.25}$$

We compare the response of the system obtained with Alg. 4.2, with the response obtained with a classic SSA from Alg. 3.1.

The classic SSA should perform the following binding and unbinding reactions

$$pr_{00} + A \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{0A}, \qquad pr_{00} + A \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{A0},$$

$$pr_{00} + R \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{0R}, \qquad pr_{00} + R \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{R0},$$

$$pr_{0A} + A \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{AA}, \qquad pr_{A0} + A \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{AA},$$

$$pr_{0R} + A \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{AR}, \qquad pr_{R0} + A \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{RA},$$

$$pr_{0A} + R \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{RA}, \qquad pr_{A0} + R \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{AR},$$

$$pr_{0R} + R \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{RR}, \qquad pr_{R0} + R \underset{k_{\text{off}}}{\overset{k_{\text{on}}}{\rightleftharpoons}} pr_{RR}.$$

With the notation $pr_{b_1 b_2}$ we denote each of the $3^2 = 9$ possible state of the promoter, where only those in which no repressor is bound should be consider active (according to the rule h).

Whether the DMSSA creates one binding matrix $\mathbf{B}$ of size $c \cdot 2$, where c is the copy number of the promoter, for representing the dynamics of the two binding sites, the classic SSA has to deal with 12 binding and unbinding reactions and with three more slow reactions, involving the transcription reaction from all the possible active states of the promoter, according to the majority rule h

$$pr_{00} \xrightarrow{k_{\text{trsc}}} mRNA + pr_{00},$$

$$pr_{0A} \xrightarrow{k_{\text{trsc}}} mRNA + pr_{0A},$$

$$pr_{A0} \xrightarrow{k_{\text{trsc}}} mRNA + pr_{A0},$$

$$pr_{AA} \xrightarrow{k_{\text{trsc}}} mRNA + pr_{AA}.$$

Table 4.1

The kinetic parameters of the model described with reaction from Eq. (4.23) and Eq. (4.24).

constant	*value*	*description*
k_{on}	$10^8\ M^{-1}s^{-1}$	TF binding
k_{off}	$10^2\ s^{-1}$	TF unbinding
k_{trsc}	$0.018\ s^{-1}$	transcription
k_{trsl}	$0.043\ s^{-1}$	transcription
k_{deg}	$0.002\ s^{-1}$	protein degradation
$k_{\text{deg-mRNA}}$	$0.001\ s^{-1}$	mRNA degradation

For practicality, we have not considered promoter leakage[2]. However, we can consider the promoter with two empty binding sites as active.

The initial concentrations for the activator A and repressor R are set to 2000 molecules (which is approximately 1.66 μM in a HEK293 cell with a volume of $\approx 2 \cdot 10^{-12} dm^3$ [193]) and maintain constant thorough the simulation. The plasmid copy number, i.e. the initial concentration for the promoter of the gene P is set to 100 copies. The kinetic parameters of the model are listed in Tab. 4.1. We assume the dissociation constant K_d being in the micro molar region, and with a high value of k_{on}, the unbinding constant k_{off} can be therefore calculated as per $k_{\text{off}} = K_d \cdot k_{\text{on}}$. All the other parameters are reference values, that can be obtain from Eq. (3.15), Eq. (3.19) and Eq. (3.26) by choosing sampled values from Tab. 3.1, Tab. 3.2 and assuming a translation-elongation rate of 40 *a.a./s*. Also we assume that transcription is driven by the RNAPII. Note the difference in magnitude between the binding/unbinding kinetic constants and all the other parameters.

The simulation responses for the protein P from the DMSSA and the SSA are depicted in Fig. 4.1. The mean percentage error of the DMSSA's mean trajectory from the mean trajectory of the SSA is roughly 6%.

In order to statistically verify the validity of the DMSSA, we compare the simulation runs of both the SSA and the DMSSA for the output response of the protein P. We use a strategy that was adopted already in [95], where the virtual fast system $\hat{X}^f(t)$,

[2] This is a scenario where also an inhibited promoter may allow (occasionally) the RNAP to initiate the transcription, allowing the mRNA to be transcribed.

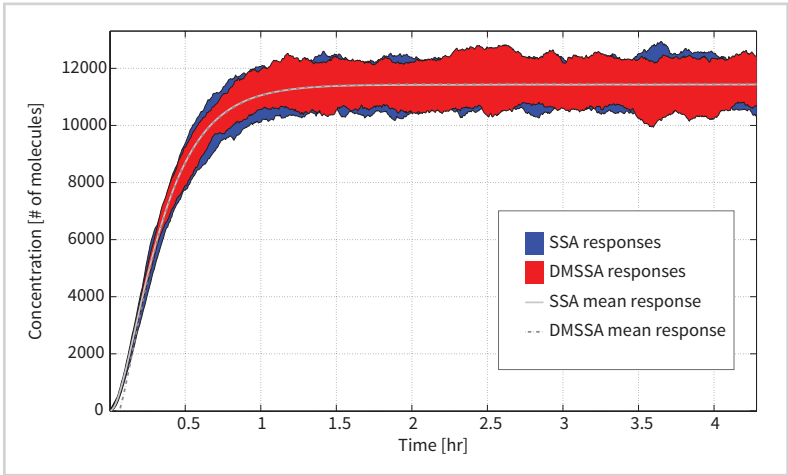


Figure 4.1

The first 100 of the 10,000 SSA and DMSSA responses obtained from the simulations for the protein P. The plot in red shows the range of the DMSSA trajectories, while the plot in blue shows the trajectories obtained from simple SSA runs. The continuous and the dotted lines in grey represent the mean values of all the 10,000 trajectories for the SSA and the DMSSA respectively.

described in section A.1.1, was compared to the fast system $X(t)$ at partial equilibrium. Here authors used the frequency distributions of multiple SSA simulation runs, sampled at different times, to show that at partial equilibrium, the two processes $\hat{X}^f(t)$ and $X(t)$ are equivalent.

We performed 10,000 runs for both the SSA and DMSSA, and we obtained the frequency distributions of the output response protein P, from each method at times $1h$, $2h$ and $4h$. Frequencies are computed over 100 bins[3] on the interval ranging from the minimum to the maximum sampled value at the chosen time over all the simulation responses. The obtained frequency distributions are plotted in Fig. 4.2. From Fig. 4.2 it is already clear, that both SSA and DMSSA return the same response. Note, that the 10,000 runs of both SSA and DMSSA were performed in parallel with the aid of 48 concurrent threads, for which the SSA took more than three days, while the DMSSA needed roughly eleven hours to complete the simulations (almost seven times faster than the SSA).

To statistically validate the equivalence of both the methods, we performed a two-sample *Kolmogorov-Smirnov test* (KS-test) over all the three frequency distributions.

[3] We divided the range of sampled values in 100 subintervals. The frequency distributions (or *histogram*) are then computed based on how many samples fall in each subinterval.

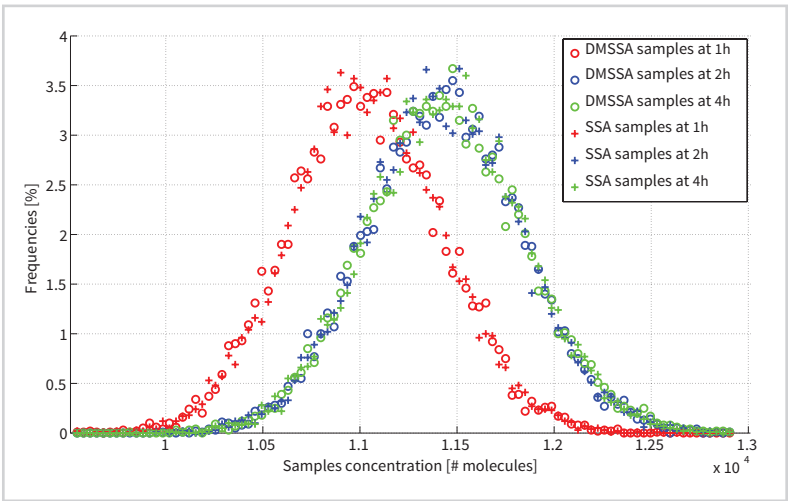


Figure 4.2

A comparison of the two frequency distributions of the SSA and DMSSA responses obtained from 10,000 simulation runs. The samples where captured at three different times: $1h$, $2h$ and $4h$ and classified in 100 bins with equal width. At time $t = 1h$, all the simulation responses show that the protein P is still increasing in concentration towards the steady state. At times $2h$ and $4h$ the simulations responses reached already the steady state, hence the two frequency distributions coincide.

The test statistic used in the KS-test is defined as

$$D_{n_1,n_2} = \sup_x \left\| \hat{F}(x_1) - \hat{F}(x_2) \right\|, \tag{4.27}$$

where $\hat{F}(x_1)$ and $\hat{F}(x_2)$ denote the two empirical distributions functions, of the two frequencies vectors x_1 and x_2 that we want to compare, n_1 and n_2 denote the size of the two samples, i.e. the vectors length, and sup stands for the supremum function. In our case $n_1 = n_2$, which equals the number of bins (100).

The KS-test presupposes as a null hypothesis H_0 that the two frequencies vectors x_1 and x_2 are sampled from the same distribution. The test rejects the null hypothesis, if

$$D_{n_1,n_2} > c(\alpha)\sqrt{N}, \tag{4.28}$$

where $N = \frac{n_1+n_2}{n_1 n_2}$, α is the significance level and $c(\alpha) = \sqrt{-\frac{1}{2}\ln\left(\frac{\alpha}{2}\right)}$. We set $\alpha = 0.01$, and hence $c(\alpha)\sqrt{N} = 0.23$ and the results of the KS-tests are the following:

	D	$D > 0.23$	$p - value$	$p \leq \alpha$	reject H_0?
$t = 1h$	0.04	no	1.0	no	no
$t = 2h$	0.03	no	1.0	no	no
$t = 4h$	0.06	no	0.99	no	no

The p-value in all the three tests is always greater than α, which confirms the null hypothesis to be non-rejectable. Since the test statistic D is lower than the critical value $c(\alpha)\sqrt{N} = 0.23$ at each time point, we can convene that the frequency distributions of the trajectories computed from the SSA and DMSSA are likely drawn from the same distribution and hence statistically equivalent. The test was performed with the MATLAB® tool *kstest2* [194].

5

Sensitivity analysis in systems and synthetic biology

Lt. Cmdr. Matt T. Sherman: My responsibility is this boat, and to get her out of here I'd even make a pact with the devil.
Lt. J.G. Nicholas Holden That's where I come in.
Lt. Cmdr. Matt T. Sherman: That's right.

Operation Petticoat, 1959

Mathematical and computational models are usually affected by some levels of uncertainty, that can be due to a variety of reasons: unknown parameter values and uncertain model inputs, unpredicted influence of a parameter in the model outputs, but also poor confidence of the modeller in the model design. The sensitivity analysis is the study of the uncertainty of the model outputs, based on the uncertainty of the model inputs [195]. We say that a model is "sensitive" to a particular input (or parameter) when a small change in its value deeply affects the value of the output (or outputs). In order to evaluate and estimate such effects, sensitivity analysis has developed a wide range of methods and techniques.

Sensitivity analysis provides powerful methods to uncover the insights of dynamical models when the response of the system cannot be easily described as a simple function of the inputs. Moreover, sensitivity analysis can be tied to model validations in order to provide an estimation of the model uncertainty. Ultimately, the sensitivity analysis can guide experimental design and execution, in the sense that, the most sensitive model inputs are potential candidates for further experimental investigation [84].

Sensitivity analysis can also be used to assess the robustness of a biological system. Robust biological systems are insensitive to input and environmental perturbations,

hence they may enhance their response effectiveness no matter how noisy conditions are present during the experimental application (or in the natural environment).

Hereafter we will briefly review the methods and techniques that are prevalently used in the field of system and synthetic biology for performing the sensitivity analysis [84].

5.1 Sensitivity analysis methods

Sensitivity analysis can be divided in two subcategories: *local sensitivity analysis* and *global sensitivity analysis*. Local sensitivity analysis refers to analytical approaches that are capable to measure and assess the variations of the model output, regarding small variations of one model input parameter at a time. Generally these variations are referred to a localized region around a specific base point. Global sensitivity analysis instead refers to numerical and statistical approaches, used to assess the variations of the model output, in light of large perturbations of multiple model inputs.

5.2 Local sensitivity analysis

A common definition of the local sensitivity is based on the sensitivity coefficients, which are the first order partial derivatives of the model output function with respect to specific inputs[1]:

Definition 5.1: The sensitivity coefficient S of a model output function y of the input parameter x_i is defined as the partial derivative of the function over the parameter:

$$\begin{aligned} S(x_i) &= \frac{\partial y(x_1, x_2, \dots, x_i, \dots, x_n)}{\partial x_i} \\ &= \lim_{\Delta x_i \to 0} \frac{y(x_1, x_2, \dots, x_i + \Delta x_i, \dots, x_n) - y(x_1, x_2, \dots, x_i, \dots, x_n)}{\Delta x_i}. \end{aligned} \tag{5.1}$$

By definition then local sensitivity provides an estimated value of how much an input parameter affects the model response output around a local point in the parameter

[1] The inputs of the model that are of interest in the sensitivity analysis are often referred as input parameters.

space [84]. Two of the most used local sensitivity analysis approaches that are based on Def. 5.1 are the following:

- *Finite difference approximation*: The most simple approximation of the partial derivative from Def. 5.1 is by choosing a small value of Δx_i and compute the sensitivity coefficients as

$$S(x_i) \approx \frac{o(x_1, x_2, \dots, x_i + \Delta x_i, \dots, x_n) - o(x_1, x_2, \dots, x_i, \dots, x_n)}{\Delta x_i}.$$

  Here the quality of estimation depends solely on the choice of value Δx_i.

- *Metabolic control analysis*: Metabolic control analysis (MCA) is a type of sensitivity analysis specifically developed to study the influence of input parameters in large metabolic pathways. Essentially it quantifies this influence with a sensitivity coefficient C defined as

$$C = \frac{\partial \ln(y)}{\partial \ln(x)}, \tag{5.2}$$

  where x and y are an input and an output of the model, respectively. The method has been prevalently used on different models of metabolic pathways and networks.

We refer to [84] for a complete review of local sensitivity analysis techniques. Local sensitivity analysis approaches are able to capture and estimate only the uncertainty of the outputs with respect to the model inputs, in a localized region of the parameter space. Often, however, it is required to address an estimation of this uncertainty for a larger region in the parameter space. Global sensitivity analysis approaches are able to cope with such a challenge and they provide a plethora of techniques to compute the sensitivity of model parameters globally.

5.3 Global sensitivity analysis

In the last two decades, several computational techniques were developed for performing global sensitivity analysis on dynamical models. The majority of these approaches has strong statistical foundations, meaning that they try to ensure a statistical independence between the design of the experiment and the model itself. Roughly, the global

sensitivity analysis needs to investigate and to assess how large variations (or perturbations) in the input parameters affect the output of the model. This could be translated as "try to change the inputs by any possible value of the parameter space, and observe the output response"[2], which literally means to sample indiscriminately the entire parameter space (which tends to be huge also for smaller models). This is a common practice of the so called *one-at-a-time* (OAT) approaches, which the Morris method is a typical representative. In this context, the study of perturbation of all possible values of the model parameters becomes crucial. Therefore the majority of approaches has to face the problem of parameter sampling. A review of all the different techniques and methods used for global sensitivity analysis is beyond the scope of this dissertation, hence we refer to [84, 195, 196] for a complete overview of the most popular approaches.

Hereafter we briefly review the Morris screening experiment, which is, due to its simplicity, one of the most largely used approach for estimating the global parameter sensitivities in systems biology models.

5.3.1 The Morris method

As a screening design technique, the experiment proposed by Morris [197], allows to rank the input parameters in order of importance [196], according to their sensitivity values. The Morris method, also known as the *Morris screening experiment* (MSE), is an OAT approach, that allows to determine which input parameter of the model has one of the following types of effects on the model outputs:

- negligible,
- additive or linear,
- non-linear or having interaction with other parameters.

The method is defined as follows. Let $\mathbf{x}$ be the set (vector) of all the k input parameters of the model, i.e.

$$\mathbf{x} = (x_1, x_2, \dots, x_k).$$

[2] Not all global sensitivity analysis techniques are based on this presumption. See [195] for more information.

Each input parameter is defined as a value (also called level) sampled from its reference interval with p possible discrete values

$$\left[0, \frac{1}{p-1}, \frac{2}{p-1}, \dots, \frac{p-2}{p-1}, 1\right], \tag{5.3}$$

where the value 0 and 1 refer to the normalized minimum and maximum values that the input parameter can take. Therefore an interval with p levels is a linearly divided region of the input parameter space with p discrete values. If the interval spans over multiple orders of magnitude, then it should be divided in p logarithmically spaced points. For such intervals, the definition adopted in Eq. (5.3) is instead rewritten as

$$\left[10^0, 10^{\frac{1}{p-1}}, 10^{\frac{2}{p-1}}, \dots, 10^{\frac{p-2}{p-1}}, 10^1\right], \tag{5.4}$$

which can be normalized, if necessary to the range $[0, 1]$. In practice the sampled parameters value are then rescaled to their "real" values (non-normalized).

In order to assess how much the i-th input parameter affects the system output, Morris suggests to compute the so-called *elementary effects*. By denoting with $\mathbf{x} + \Delta_i$ the change in the vector

$$[x_1, \dots, x_{i-1}, x_i + \Delta, x_{i+1}, \dots, x_k], \tag{5.5}$$

then an elementary effect is the ratio

$$d_i(\mathbf{x}) = \frac{y(\mathbf{x} + \Delta_i) - y(\mathbf{x})}{\Delta}, \tag{5.6}$$

where Δ is a predefined multiple of $\frac{1}{p-1}$ and the function $y(\mathbf{x})$ is the system output function, i.e. the system response function. The value $x_i + \Delta$ must lie inside the interval of Eq. (5.3) (or Eq. (5.4)). If the value $x_i + \Delta$ overflows the interval, then $x_i - \Delta$ is computed. Saltelli et al. [196] suggest to always set p as an even integer.

Note that the distribution of each elementary effect F_i from a random sample of k input parameters, in which values are randomly chosen from the interval in Eq. (5.3) (or Eq. (5.4)) is a simple permutation with repetition, but with the limitation of $x_i \pm \Delta$

still being inside the interval. By setting

$$\Delta = \frac{a}{p-1} \tag{5.7}$$

and by drawing a table of all the values from the interval in Eq. (5.3) (as columns) and all the values of the offset a (ranging from 1 to $p-1$ as rows), it is straightforward to compute the total number of elements for each F_i as $p^{k-1}\left[p - \Delta(p-1)\right]$.

The method requires to randomly sample the parameters space r-times, obtaining thus r-different samples of $\mathbf{x}$, upon which r- elementary effects are computed for each input parameter. This can be done by creating the *random trajectory generator matrix* (RTGM) $\mathbf{R}^*$, which is a $r \cdot k$ matrix containing r randomly sampled vectors $\mathbf{x}$.

The method of Morris provides two sensitivity measures for assessing the influence of the input parameters to the outputs, namely μ_i and σ_i. The first is the mean of elementary effects of the i-th input parameter over all the r samples, while the second is its standard deviation. Formally

$$\mu_i = \frac{\sum_{j=1}^{r} d_{ij}}{r}, \tag{5.8}$$

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{r}(d_{ij} - \mu_i)^2}{r}}. \tag{5.9}$$

The first measure μ_i allows to rank the input parameter by importance (i.e. which inputs affects the output most), while σ_i provides insights of which input parameter has

- a non-linear effect over the model or
- any sort of interaction among other parameters.

For instance, a low σ_i indicates a similar value of the elementary effects, meaning that the effect of the i-th input parameter is independent of the values taken by other parameters. On the other hand a high value of σ_i may indicate, that the elementary effects are strongly dependent on the choice of the values of the other parameters [196].

The measure μ_i has been later enhanced by Campolongo et al. [196, 198], which proposed the measure μ_i^* as the mean of absolute values of elementary effects. This

prevents the occurrence of opposite signs in the values of d_i and hence it allows to order the input parameters by an absolute overall influence on the output. This measure can be formalized as

$$\mu_i^* = \frac{\sum_{j=1}^{r} |d_{ij}|}{r}. \tag{5.10}$$

Both measures μ_i and μ_i^* (and therefore also σ_i) are highly dependent on the value of r. This parameter should be large enough to ensure an equal probability of sampling. However, it has been demonstrated in practice, that r does not need to be too large (authors in [196] demonstrated good results with $r = 10$). The value r defines the row-size of the matrix $\mathbf{R}^*$, which is used as a source for input parameter samples. $\mathbf{R}^*$ can be constructed by a Latin Hypercube Sampling or by an orthogonal sampling strategy (see section 5.4).

Each row of $\mathbf{R}^*$ provides one sample of $\mathbf{x}$. From each row of $\mathbf{R}^*$ a matrix $\mathbf{B}^*$ of size $(k + 1) \cdot k$, called the *orientation matrix*, can be constructed as follows:

1. Take a row from $\mathbf{R}^*$ and place it as the first row in the matrix $\mathbf{B}^*$.
2. The l-th row of $\mathbf{B}^*$ is created by copying the $(l - 1)$-th row and by augmenting (perturbing) its $(l - 1)$-th element randomly by $+\Delta$ or $-\Delta$.
3. Repeat the step 2 for the row $l + 1$ until the $(k + 1)$-th row is formed. This is the row in which the last element of the initial row taken from $\mathbf{R}^*$ is modified.

The main property of $\mathbf{B}^*$ is that any two adjacent rows differs exactly by one element, and this difference is exactly Δ. Therefore $\mathbf{B}^*$ can provide the samples for computing one elementary effect per input parameter, by taking any two adjacent rows from $\mathbf{B}^*$. The orientation matrix $\mathbf{B}^*$ defines a *trajectory* in the parameters space (see [196] for schematics and for additional examples). The matrix $\mathbf{R}^*$ hence provides r different orientation matrices $\mathbf{B}^*$ and therefore r different trajectories in the parameters space.

Usually a set of matrices $\mathbf{D}^*$, called the *signs matrices*, are constructed together with the orientation matrices. The matrix $\mathbf{D}^*$ is a k-dimensional diagonal matrix that defines the sign whether the Δ value should be added or subtracted to the input parameter value, so the parameter's change will still be inside the domain interval, i.e. the diagonal values of $\mathbf{D}^*$ are either $+1$ or -1, see Ex. 5.1 for more details. The use of matrices $\mathbf{R}^*$, $\mathbf{B}^*$ and $\mathbf{D}^*$ simplify the design and ensure a high economy value of the overall experiment, as will be discussed later.

Example 5.1: Let be $\mathbf{x}$ the input vector of size $k = 3$, and let be $\Delta = 0.5$. Let all the input parameter be defined in the interval $[0, 1]$, and let be $r = 2$. Given $\mathbf{R}^*$:

$$\mathbf{R}^* = \begin{bmatrix} 0.2 & 0.3 & 0.6 \\ 0.1 & 0.7 & 0.4 \end{bmatrix},$$

$$\mathbf{B}_1^* = \begin{bmatrix} 0.2 & 0.3 & 0.6 \\ 0.7 & 0.3 & 0.6 \\ 0.7 & 0.8 & 0.6 \\ 0.7 & 0.8 & 0.1 \end{bmatrix}, \quad \mathbf{B}_2^* = \begin{bmatrix} 0.1 & 0.7 & 0.4 \\ 0.6 & 0.7 & 0.4 \\ 0.6 & 0.2 & 0.4 \\ 0.6 & 0.2 & 0.9 \end{bmatrix},$$

$$\mathbf{D}_1^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad \mathbf{D}_2^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The sensitivity measures can be then computed by constructing r different orientation matrices and computing $r \cdot k$ elementary effects d_i.

One drawback of the definition of Eq. (5.6) is that a small value for Δ may cause an unexpected large elementary effect for the input parameter under scrutiny, and hence an unexpected high sensitivity apportioned to it. Such shortcoming can be overcome by normalizing the elementary effects as suggested in the work of Sin and Gernaey [199], in which the value d_i is multiplied by a ratio of standard deviations of the inputs and outputs. In short

$$d_i(\mathbf{x}) = \frac{y(x_1, \dots, x_{i-1}, x_i + \Delta, x_{i+1}, \dots, x_k) - y(\mathbf{x})}{\Delta} \frac{\sigma_{xi}}{\sigma_{yi}}, \tag{5.11}$$

where σ_{xi} and σ_{yi} are the standard deviation of the inputs x_i and the outputs y_i, respectively. This strategy is useful especially for comparing the elementary effects on different outputs [199, 200].

Another important aspect concerning all the screening techniques is the *economy* of the designed experiments. The economy of a screening experiment is defined as the number of elementary effects that need to be computed by the experiment divided by the number of necessary experimental runs of the model that are required by the

definition (e.g. Eq. (5.6)) to produce them [196]. This means that in a general case a total of $r \cdot k$ elementary effects need to be calculated, from $2rk$ experimental runs, which gives an economy of 0.5. But in the MSE the economy of the experiment, due to the creation of the orientation matrices $\mathbf{B}^*$, can increase to $\frac{k}{k+1}$.

5.3.2 *Other global sensitivity analysis approaches*

There are several techniques developed to extract global sensitivities coefficients from systems and synthetic biology mathematical models. They can be categorized in two groups as listed in Tab. 5.1.

Table 5.1

Some of the variance and non-variance approaches for performing global sensitivity analysis in systems and synthetic biology models [84].

non-variance based	▪ Morris method ▪ multi-parametric sensitivity analysis ▪ partial rank correlation coefficient analysis ▪ weighted average of local sensitivities
variance based	▪ Sobol sensitivity analysis method ▪ Fourier amplitude sensitivity test (commonly known as FAST) ▪ random sampling high-dimensional model representation (RS-HDMR)

Although there is no "general" method for performing the sensitivity analysis for any custom model in systems and synthetic biology, authors in [195, 196, 201] suggest, that the choice of "which method to use and when", should be based on the following criteria [195]:

1. the computational cost of running the model,
2. the number of input parameters and
3. features of the model (e.g. linearity).

The first constraint is probably the most influential regarding the choice of technique. For global sensitivity analysis, variance-based method are known to be computationally expensive, because of the need to perform a simulation run of the model multiple times. For a large number of input parameters non-variance methods are therefore preferred.

5.4 Parameter sampling

Global sensitivity analysis approaches are highly dependent on how well the parameters' space is sampled. In the original work of Morris a random sampling suffices for creating r samples of the input parameter vector $\mathbf{x}$ [196, 197]. Authors in [198] have provided an enhanced way to deal with the efficacy of sampling, by choosing only the samples that provide the maximum dispersion in the parameter space. In this context two sampling techniques are largely adopted:

- the latin hypercube sampling and
- the orthogonal sampling.

Latin hypercube sampling (LHS) is a simple method for sampling multidimensional variables. Given a vector of k parameters, each of which is defined inside an interval of m discrete values (e.g. as in Eq. (5.3)), the task is to randomly choose a set of k-values, one from each parameters' interval, by assuring a high uniformity across all the parameters dimensions. This can be achieved by choosing the samples from the parameters hyperplane, so neither two or more samples would lie on the same hyperplane axis. Such hyperplane is called Latin hypercube. For instance the hyperplane generated by solely two parameters is a two dimensional grid, where the samples are chosen in way so there would be only one sample on each row and column (see Fig. 5.1(a)). Such two dimensional grid is known as a Latin square, from which the generalized term Latin hypercube derives.

With the Latin hypercube one is able to reduce the correlation between samples over the entire parameter space, that may occur from a random sampling. LHS ensures a sufficient level of sparsity between samples. The shortcoming of this strategy is that it could sometimes produce high correlated samples within local (smaller) subspaces, e.g. in the two dimensional space, the samples could be picked by simply choosing all the diagonal values on the grid. Since most designed experiments demand uncorrelated

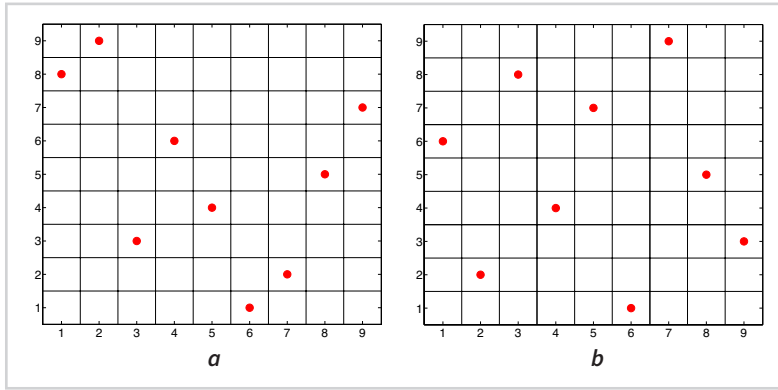


Figure 5.1

An illustrative example of two dimensional grids and the sampling of two parameters (red dots) [84]. (a) A Latin hypercube constructed for two parameters, having both the same range of values. (b) Orthogonal sampling, with three subspaces per dimension.

samples, not only over the entire parameter space, but also within smaller subspaces, it is often preferable to use other strategies, such as the orthogonal sampling.

Orthogonal sampling is a sampling improvement, that additionally decreases the correlation between samples, by ensuring the Latin square property also for smaller closed subspaces. For a two dimensional grid, such subspaces are smaller squares, which contain exactly one sample (see Fig. 5.1(b)).

The orthogonal sampling is a preferable choice when large number of parameters and larger intervals need to be sampled. The main drawback of the orthogonal sampling against the Latin hypercube sampling is its computational complexity, which is exponential with the number of subspaces. We refer to [202, 203] for more insights and technicalities for constructing orthogonal and Latin hypercubes.

The Latin hypercube and the orthogonal sampling are not the only approaches used in systems biology models for sampling the parameters spaces. The Gaussian sampling and the Monte Carlo search can also be adopted for such task [204].

6

Sensitivity analysis of GRNs with complex cis-regulatory modules

Chief Mechanic's Mate Sam Tostin: I'm a religious man, Captain, and I believe we'll get through if the Good Lord puts His mind to it. Of course, He'll have to give us His undivided attention.

Operation Petticoat, 1959

In order to understand the role of multiple TFBSs in the cis-regulatory modules, we propose to perform the global sensitivity analysis on the results of stochastic multi-scale simulations executed by DMSSA. Here we would like to understand and assess how much the variation in the number of TFBSs affects the system response, by using different metrics.

6.1 Sensitivity analysis of stochastic models

To perform a global sensitivity analysis on a stochastic model, one should take into consideration its computational definition and its complexity. In the work presented in [86, 205–209], authors have shown that sensitivity analysis approaches can be successfully applied to known stochastic models in systems biology. A short review of such approaches can be found in [208, 210]). Several software packages, that perform stochastic sensitivity analysis for chemical reaction networks have been also developed (see for example [87, 211]). The sensitivity analysis of stochastic models can provide good estimates for grading the overall robustness of stochastic models. Our methodology aims to analyse stochastic models that exhibit the *multi-scale condition*. The lack of methodologies for performing sensitivity analysis for such systems and the increas-

ing need for such techniques in systems and synthetic biology, has triggered a notable effort in recent years to fill this gap.

6.2 Sensitivity analysis of multi-scale stochastic models

The basic approach for the sensitivity analysis of multi-scale stochastic models is to evaluate the parameter sensitivities for the slow-scale system, while the fast-scale system is considered to be in a steady state. The parameter sensitivities can be then computed in the same way as for single scale systems.

This is the most largely adopted technique (see [212, 213] for more details). Gupta and Khammash formally prove that in certain conditions the parameter sensitivities evaluated for the reduced (single scale) model (based on the technique presented in [187]) equal the sensitivities obtained for original (multi-scale) model [44].

However, our main concern in the approaches using the reduction of the fast system, similarly as for the ssSSA previously in chapter 4, is that it would be difficult, if not impossible, to apply such reduction to a large stochastic multi-scale reaction network containing as many binding reactions as in Eq. (4.5).

Our aim here is to show that the parameter sensitivities of gene regulatory networks with multiple non-cooperative TFBSs can be evaluated by relying on extensive simulations via the DMSSA. In order to achieve that, we propose to use the Morris method from chapter 5 with few minor expedients.

6.3 The response function of the Morris method

The output of DMSSA is the discrete-time evolution of the system state

$$X = \left[y_1, y_2, \dots, y_j, \dots, y_N\right].$$

Let's observe only one discrete-time output of the DMSSA, i.e. the j-th species concentration, which we will denote as $y_j[n]$ or simply as $y[n]$ and let's denote with $y^{\mathbf{x}}[n]$ the same output obtained from the DMSSA with the input factors set $\mathbf{x}$. Let's also denote with $y^{\mathbf{x}+\Delta_i}[n]$ the same output, obtained from the DMSSA with the input factors set $\mathbf{x}$, having the i-th parameter changed by Δ, as per Eq. (5.5). The Morris method is based on the computation of elementary effects via Eq. (5.6), in which the model response function $y(\mathbf{x})$ provides a single scalar time-independent value. The same elementary effects cannot be computed, if the output sequence $y[n]$ is used in Eq. (5.6) instead of

$y(\mathbf{x})$. In this scenario the vector $y^{\mathbf{x}}[n']$ cannot be subtracted from $y^{\mathbf{x}+\Delta_i}[n'']$, unless the two discrete-time vectors are defined over the same discrete times, i.e. $n' = n''$. To allow such subtraction, we have to re-compute, i.e. interpolate the two vectors sequences of points over the same (and possible equidistant) discrete time points. This is analogous to the problem of approximating a discrete time series of unequally spaced observations. The Euler numerical integration approach with a constant time step δt is one of the easiest way to implement such interpolation. As such the elementary effects d_i, for each of the model's parameters, become a discrete-time sequences $d_i[n]$.

In order to avoid the time dependency completely, a representative value of the discrete-time output $y[n]$ has to be chosen. A possible candidate is the expected value

$$\mathbb{E}\left(\, y[n] \,\right) = \frac{1}{N} \sum_{n=0}^{N-1} y[n] = \overline{y}[n] = \overline{y}, \tag{6.1}$$

where N is the length of the output response $y[n]$. The value $\overline{y}$ is simply the time series average of the model output. The utilization of such average value is suggested in the works of Rathinam [86] and Sheppard [208]. Previous works [200] suggested instead to avoid time dependency by performing the *principal component analyses* (PCA) over all the model output responses. This is a good strategy, if one has to deal with a large number of different model outputs with multiple dimensions (21 in the case of the model of glucose homoeostasis analysed in [200]). However, with the DMSSA, we usually deal with just few model outputs responses, i.e. the output reporter proteins of GRNs. The PCA gives no additional benefits than the simple expected value, if it is performed over just few single-dimensioned output responses. Nevertheless a PCA implementation over all the time points needed by the Morris experiment, would be computationally expensive.

Instead, the average of the model output $\overline{y}$ provides a robust approximation of the model response in a scenario where the output is stationary through time. However, for oscillatory behaviours, the value of $\overline{y}$ does not adequately represent the response function, as we will later explain in section 6.3.3. For such scenario, we propose to use the period of the principal frequency of the model output y as a replacement for $\overline{y}$. Other alternatives rely on the Fourier coefficients of the Fourier series approximation, that fits the time points of the model output response y, as suggested in [214]. The details about this latter approach are presented in section A.2.1.

Although the model responses of particular output proteins are commonly used for determining the parameter sensitivities, the information regarding the importance and the effect that multiple binding sites can have over the model outputs, can be biased by several intermediate uncertainties that are present between the beginning TF regulation and the final product of translation – the output protein. In order to decrease such bias and to provide additional insights in the sensitivity of the gene regulation controlled by cis-regulatory modules containing multiple non-cooperative TFBSs, we propose a new measure to be used as a replacement of $y(\mathbf{x})$ in Eq. (5.6) and Eq. (5.11), i.e. a *TF binding gradient*.

6.3.1 The TF binding gradient

The TF binding gradient of the i-th copy of the m-th promoter, denoted by $g_m(t)$, is defined as

$$g_{m,i}(t) = \sum_{j=1}^{n_m} b_{i,j}^m(t), \tag{6.2}$$

where $b_{i,j}^m(t)$ is the binding site value from Eq. (4.6) and n_m is the number of TFBSs on the m-th promoter of the GRN. Summing up all the gradients for all the promoter copies, we obtain

$$g_m(t) = \sum_{i=1}^{c_m} g_{m,i}(t) = \sum_{i=1}^{c_m} \sum_{j=1}^{n_m} b_{i,j}^m(t). \tag{6.3}$$

A positive value of $g_m(t)$ tells us, that on the promoters of m-th type in the GRN, there are currently bound more activators than repressors. This can indicate a global active state of the m-th promoter. A negative value of $g_m(t)$ instead tells us, that promoters are inhibited. A value of 0 can indicate two things: either all the binding sites are empty or the total number of bound activators equals the total number of bound repressors.

The value $g_m(t)$ is limited by the structural parameters c_m and n_m. The maximal positive value of $g_m(t)$ is $c_m \times n_m$, which means that all the binding sites in all the copies of the m-th promoter in the GRN are occupied by the activators, while the maximal negative value $-c_m \times n_m$ means that all the binding sites in all the copies of the m-th promoter in the GRN are occupied by the repressors. The value $g_m(t)$ can be thus normalized in percentages inside the range [-100%, +100%].

As for the output response $y[n]$, the time evolution of the binding sites dynamics

can be represented by the discrete-time sequence

$$g_m[n] = \left[g_m(t_1), g_m(t_2), \dots \right]. \tag{6.4}$$

The time evolution of the binding gradient can provide excellent representation of the TFs binding dynamics inside the GRN, from which is possible to establish a good metric for apportion the uncertainty to the input parameters, regarding the influence of TF binding activity on the m-th promoter. We propose to use the binding gradient as a replacement of the model response $y[n]$ for calculation of the elementary effects. We can adopt the expected value of the gradient $g_m[n]$ to cancel the time dependency as

$$\mathbb{E}\left(g_m[n]\right) = \frac{1}{N} \sum_{n=0}^{N-1} g_m[n] = \overline{g}_m[n] = \overline{g}_m. \tag{6.5}$$

The average binding gradient $\overline{g}_m$ tells us the mean percentage of activation (or repression) of the m-th promoter through the entire simulation time.

In order to simplify the notation we will assume that only one type of promoter (i.e. $m = 1$) with multiple TFBSs is observed as a gradient output of DMSSA. We can therefore write $g_m[n]$ as $g[n]$ and $\overline{g}_m[n]$ as $\overline{g}_m[n]$ or $\overline{g}$.

6.3.2 *Adapted Morris elementary effects*

A single output sequence $y_j[n]$ is not statistically representative in stochastic simulations. Hence, the computation of the elementary effects in the Morris method via a single trajectory of the DMSSA will lead to unreliable results for parameters sensitivities. In order to overcome this shortcoming, multiple runs of the DMSSA must be performed to obtain a statistically valid representation of the model behaviour.

By repeating the simulation for obtaining $y[n]$ M-times we obtain the matrix

$$\mathbf{Y} = \begin{bmatrix} y_1[n_1] \\ y_2[n_2] \\ \vdots \\ y_M[n_M] \end{bmatrix} = \left[y_i[n_i] \right]_{M \times N}, \tag{6.6}$$

where N is the number of response samples (points) and $n_i, i = 1 \dots M$ refers to the M different discrete-time series of unequally spaced points on which the model outputs

are defined. If the number of runs M is large enough, then the average response of these multiple runs can be instead considered as a statistically representative measure for the model response. Similarly we can obtain the matrix $\mathbf{G}$, which provides M instances of the binding gradient g as

$$\mathbf{G} = \begin{bmatrix} g_1[n_1] \\ g_2[n_2] \\ \vdots \\ g_M[n_M] \end{bmatrix}. \tag{6.7}$$

The average mean response

We propose to include into the computation of the elementary effects, multiple simulation runs, obtained with the same input parameters. We can construct a vector of expected values

$$\mathbb{E}(\mathbf{Y}) = \begin{bmatrix} \overline{y}_1[n_1] \\ \overline{y}_2[n_2] \\ \vdots \\ \overline{y}_M[n_M] \end{bmatrix} = \overline{\mathbf{Y}}, \tag{6.8}$$

i.e. a vector containing the mean value of each output response of the matrix $\mathbf{Y}$. The average mean of $\overline{\mathbf{Y}}$ is a scalar

$$s_y(\mathbf{x}) = \mathbb{E}\left(\overline{\mathbf{Y}}\right) = \frac{1}{M}\sum_{i=1}^{M} \overline{y}_i[n_i], \tag{6.9}$$

where the averages $\overline{y}[n_i]$ are computed over the outputs obtained from the simulations of DMSSA with the input factors $\mathbf{x}$. The value $s_y(\mathbf{x})$ is a time independent statistic which tells us what is the expected average of the output response over multiple simulation runs. Such statistic can be directly applied to the computation of the elementary effects as

$$d_i^{s_y}(\mathbf{x}) = \frac{s_y(\mathbf{x} + \Delta_i) - s_y(\mathbf{x})}{\Delta}. \tag{6.10}$$

The elementary effects provided in such form, provide a robust and time independent value for computing the overall parameters sensitivities over the r trajectories required

by the Morris approach. The two sensitivities metrics μ_i^* and σ_i are therefore

$$\begin{aligned} \mu_i^*\left(s_y\right) &= \frac{1}{r}\sum_{l=1}^{r}\left|d_i^{s_y}(\mathbf{x}_l)\right|, \\ \sigma_i\left(s_y\right) &= \sqrt{\frac{\sum_{l=1}^{r}\left(d_i^{s_y}(\mathbf{x}_l)-\mu_i\right)^2}{r}}, \end{aligned} \tag{6.11}$$

where

$$\mu_i\left(s_y\right) = \frac{1}{r}\sum_{l=1}^{r} d_i^{s_y}(\mathbf{x}_l).$$

Similarly we can compute the elementary effects and the parameter sensitivities, through the binding gradient $g[n]$. By removing its time-dependency with the expected value of the matrix $\mathbf{G}$, we obtain

$$\mathbb{E}\left(\mathbf{G}\right) = \begin{bmatrix} \overline{g}_1[n_1] \\ \overline{g}_2[n_2] \\ \vdots \\ \overline{g}_M[n_M] \end{bmatrix} = \overline{\mathbf{G}}, \tag{6.12}$$

which is a matrix of binding gradients averages, obtained from multiple simulation runs with the same input parameters $\mathbf{x}$. The average mean of $\overline{\mathbf{G}}$ is

$$s_g(\mathbf{x}) = \mathbb{E}\left(\overline{\mathbf{G}}\right) = \frac{1}{M}\sum_{i=1}^{M}\overline{g}_i[n_i]. \tag{6.13}$$

As s_y, also s_g is a time independent statistic and therefore it can be directly used to compute the elementary effects as

$$d_i^{s_g}(\mathbf{x}) = \frac{s_g(x_1,\dots,x_{i-1},x_i+\Delta,x_{i+1},\dots,x_k) - s_g(\mathbf{x})}{\Delta}, \tag{6.14}$$

by which we can express the parameter sensitivities μ_i^* and σ_i as

$$\begin{aligned} \mu_i^*\left(s_g\right) &= \frac{1}{r}\sum_{l=1}^{r}\left|d_i^{s_g}(\mathbf{x}_l)\right|, \\ \sigma_i\left(s_g\right) &= \sqrt{\frac{\sum_{l=1}^{r}\left(d_i^{s_g}(\mathbf{x}_l)-\mu_i\right)^2}{r}}, \end{aligned} \tag{6.15}$$

where

$$\mu_i\left(s_g\right) = \frac{1}{r}\sum_{l=1}^{r} d_i^{s_g}(\mathbf{x}_l).$$

Both s_y and s_g accurately describe the model response and can be both used to compute the elementary effects in the Morris approach.

6.3.3 *Oscillatory response*

The main shortcoming of the parameter sensitivities computed with s_y and s_g is that they can be considered reliable only, if the model response y and the binding gradient g are exhibiting a non-oscillatory behaviour. This can be shown in the following example.

Example 6.1: Consider the case in which in Eq. (5.6) $y(\mathbf{x}+\Delta_i)$ oscillates similarly as a sinusoid function, while the signal $y(\mathbf{x})$ very quickly reaches a steady state. It can occur that both these signals can have the same mean value $\overline{y}$. In such scenario the elementary effect computed via Eq. (5.6) will equal to zero, determining a low sensitivity value for the parameter x_i and neglecting the fact that the two functions are qualitatively very different.

To address this issue we propose to compute the elementary effects with the aid of the *discrete Fourier transform* (DFT) by providing different representative values for the model response y and the binding gradient g:

1. the *average of the principal frequency* of the model responses in $\mathbf{Y}$,
2. the *average of the principal frequency* of the binding gradients in $\mathbf{G}$,
3. the *average of the principal period* of the model responses in $\mathbf{Y}$,

4. the *average of the principal period* of the binding gradients in $\mathbf{G}$,
5. the *average of the principal amplitude* of the model responses in $\mathbf{Y}$ and
6. the *average of the principal amplitude* of the binding gradients in $\mathbf{G}$.

The definitions of these representative values are presented in the next section.

The average of the principal frequency, period and amplitude

Let's consider an oscillatory output response signal $y[n]$ of N samples with the main period N_0. Hence we can write $y[n] = y[n + N_0]$, where the main frequency is defined as $1/N_0$. In practice however, the value N_0 is usually unknown and the main frequency often refers to the frequency with the highest amplitude in the spectrum. The simplest way to estimate the period N_0, and the main frequencies, is by computing the frequency spectrum of the signal $y[n]$. The DFT is the most common tool to compute the spectrum, as

$$\mathscr{F}(y) = \left\{ X_k = \sum_{n=0}^{N-1} y[n] \mathrm{e}^{-\frac{2\pi i k n}{N}}, \quad k = 0, \dots, N-1 \right\}, \tag{6.16}$$

where X_k is the k-th component (i.e. the k-th frequency) of the amplitude spectrum. The frequency with the highest amplitude in the signal is given by

$$F_{\max} = \arg\max_k \left| X_k \right| \frac{2\pi}{N}, \tag{6.17}$$

where

$$\left| X_k \right| = \sqrt{\Re(X_k)^2 + \Im(X_k)^2},$$

and the highest amplitude is given by

$$A_{max} = \max_k \left| X_k \right|. \tag{6.18}$$

The period $P_{\max}$ of $F_{\max}$ is defined as

$$P_{\max} = \frac{1}{F_{\max}}. \tag{6.19}$$

Before performing the DFT, which can be easily and efficiently implemented via the *fast Fourier transform* (FFT) algorithm, it is often beneficial to normalize the input signal $y[n]$ in order to cancel the DC term from the signal spectrum, i.e. $X_0 = \sum_{i=0}^{N-1} x_i$. This can be done simply by observing $y[n] - \overline{y}[n]$. Such pre-normalization (also known as detrending) is useful to avoid peak frequencies at zero Hz. Furthermore it is also beneficial to interpolate the discrete output $y[n]$ over equally spaced time points, namely $\mathbf{t}_{eq}$ before DFT. The spectral analysis described above can be performed for both the model output response $y[n]$ and the binding gradient $g[n]$. Let's denote with $\mathscr{F}(\mathbf{Y})$ the DFT performed on each row of the matrix $\mathbf{Y}$, i.e.

$$\mathscr{F}(\mathbf{Y}) = \begin{bmatrix} \mathscr{F}\left(y_1(\mathbf{x}, \mathbf{t}_1)\right) \\ \mathscr{F}\left(y_2(\mathbf{x}, \mathbf{t}_2)\right) \\ \vdots \\ \mathscr{F}\left(y_M(\mathbf{x}, \mathbf{t}_M)\right) \end{bmatrix}. \tag{6.20}$$

The frequencies with the highest amplitude and their periods are therefore computed as

$$F_{\max}(\mathbf{Y}) = \begin{bmatrix} F_{\max_1} \\ F_{\max_2} \\ \vdots \\ F_{\max_M} \end{bmatrix}, \quad P_{\max}(\mathbf{Y}) = \begin{bmatrix} P_{\max_1} \\ P_{\max_2} \\ \vdots \\ P_{\max_M} \end{bmatrix}, \quad A_{\max}(\mathbf{Y}) = \begin{bmatrix} A_{\max_1} \\ A_{\max_2} \\ \vdots \\ A_{\max_M} \end{bmatrix}. \tag{6.21}$$

The average maximum frequency, amplitude and period over all the M simulations, performed with the same input factors $\mathbf{x}$, are denoted with $f_y(\mathbf{x})$ and $p_y(\mathbf{x})$ respectively, and defined as

$$f_y(\mathbf{x}) = \mathbb{E}\Big(F_{\max}(\mathbf{Y})\Big) = \frac{1}{M} \sum_{i=1}^{M} F_{\max_i}, \tag{6.22}$$

$$p_y(\mathbf{x}) = \mathbb{E}\Big(P_{\max}(\mathbf{Y})\Big) = \frac{1}{M} \sum_{i=1}^{M} P_{\max_i}, \tag{6.23}$$

$$a_y(\mathbf{x}) = \mathbb{E}\Big(A_{\max}(\mathbf{Y})\Big) = \frac{1}{M} \sum_{i=1}^{M} A_{\max_i}. \tag{6.24}$$

$f_y(\mathbf{x})$, $p_y(\mathbf{x})$ and $a_y(\mathbf{x})$ provide a reliable alternative to the scalar s_y for performing the elementary effects d_i in a scenario, where the responses in $\mathbf{Y}$ have an oscillatory behaviour. E.g. using $f_y(\mathbf{x})$, the elementary effects can be computed as

$$d_i^{f_y}(\mathbf{x}) = \frac{f_y(x_1, \dots, x_{i-1}, x_i + \Delta, x_{i+1}, \dots, x_k) - f_y(\mathbf{x})}{\Delta}. \tag{6.25}$$

The related parameter sensitivities μ_i^* and σ_i are therefore

$$\begin{aligned} \mu_i^*\left(f_y\right) &= \frac{1}{r}\sum_{l=1}^{r}\left|d_i^{f_y}(\mathbf{x}_l)\right|, \\ \sigma_i\left(f_y\right) &= \sqrt{\frac{\sum_{l=1}^{r}\left(d_i^{f_y}(\mathbf{x}_l) - \mu_i\right)^2}{r}}, \end{aligned} \tag{6.26}$$

where

$$\mu_i\left(f_y\right) = \frac{1}{r}\sum_{l=1}^{r} d_i^{f_y}(\mathbf{x}_l).$$

Equivalently it is possible to express the parameter sensitivities $\mu_i^*\left(p_y\right)$ and $\sigma_i\left(p_y\right)$ from the measure $p_y(\mathbf{x})$, and $\mu_i^*\left(a_y\right)$ and $\sigma_i\left(a_y\right)$ from the measure $a_y(\mathbf{x})$.

The same equations, for the output response y, can be expressed for the binding gradient g. Shortly, by performing the DFT over the matrix $\mathbf{G}$, we obtain a similar matrix $\mathscr{F}(\mathbf{G})$, from which we can compute the maximum frequencies, with their relative maximum amplitudes, and the maximum periods for all the M simulations runs in $\mathbf{G}$. It follows, that we can express the averages f_g. p_g and a_g similarly as per Eq. (6.22) and Eq. (6.23), respectively. This allows to compute the parameter sensitivities $\mu_i^*\left(f_g\right)$, $\mu_i^*\left(p_g\right)$, $\mu_i^*\left(a_g\right)$, $\sigma_i\left(f_g\right)$, $\sigma_i\left(p_g\right)$ and $\sigma_i\left(a_g\right)$, similarly as per Eq. (6.26).

The use of the sensitivity measures $\mu_i^*\left(p_y\right)$ and $\sigma_i\left(p_y\right)$ is significant only, if the output signal exhibits periodicity. In fact they provide a time independent measure for computing robustly the elementary effects in the scenario where both the functions $y(\mathbf{x} + \Delta_i)$ and $y(\mathbf{x})$ from Eq. (5.6) oscillate. However, for the scenario described in Ex. 6.1, the computation of the elementary effects via f_y, p_y or a_y (and same for f_g, p_g or a_g) will not be reliable, because of the estimation of the main frequency over a non-oscillating signal.

The elementary effect computed in the scenario described in Ex. 6.1 should be there-

fore omitted from the computation of the sensitivities $\mu_i^*\left(s_y\right)$ and $\sigma_i\left(s_y\right)$ (or $\mu_i^*\left(p_y\right)$ and $\sigma_i\left(p_y\right)$). Hence, a system for periodicity detection has to be deployed to aid the Morris method to identify such possible scenarios.

Detection of periodicity in the model output

The detection of periodicity in the model outputs can be performed with the method of Wichert et al. [215]. Having a signal $y[n]$ with a power density spectrum $I(\omega)$ (i.e. periodogram), Wichert et al. suggest to identify the periodicity of the signal with the aid of the Fisher g-statistic and by applying the methodology presented in [216], in order to control the expected proportion of the occurring false positive detections.

Having a signal $y[n]$ being affected by stochastic noise, the periodogram $I(\omega)$ is given by

$$I(\omega) = \frac{1}{N}\left|Y_k\right|^2, \quad k = 0, 1, \dots, N-1, \tag{6.27}$$

where the Y_k is the DFT of $y[n]$ of length N. $I(\omega)$ can be rewritten as

$$I(\omega) = \frac{1}{N}\left|\sum_{n=0}^{N-1} y[n]\, \mathrm{e}^{-\frac{2\pi i k n}{N}}\right|^2. \tag{6.28}$$

The Fisher g-statistic is the value

$$g = \frac{\max_l I(\omega_l)}{\sum_{l=1}^{N_h} I(\omega_l)}, \tag{6.29}$$

which has the distribution

$$P(g > x) = \sum_{l=1}^{L} (-1)^{l-1} \binom{N_h}{l} (1 - x\, l)^{N_h - 1}, \tag{6.30}$$

where L is the largest integer less than $\frac{1}{x}$ and $N_h = \left\lfloor \frac{N-1}{2} \right\rfloor$. High value of g means that there is a significant periodicity present in the signal. Low value of g instead, means that the signal does not show periodicity [215].

The null hypothesis of the Fisher's test statistic assumes that the observed signal is not periodic (no distinctive peaks in the signal's periodogram). To test the periodicity, the

p-value for the observed value x (e.g. 0.5) has to be computed according to Eq. (6.30). If the p-value is lower than the significant level q (0.05, 0.01 or 0.001), then the null hypothesis can be rejected, and therefore we can say, that the signal $y[n]$ oscillates.

This Fisher's test can be applied for multiple testing. Having the matrix $\mathbf{Y}$ from Eq. (6.6), with M instances of the stochastic signal $y[n]$ generated from the same model output, we can compute M periodograms and thus M g-values. By sorting their p-values from Eq. (6.30) in ascending order, giving $p_1 < p_2 < \cdots < p_M$ and their corresponding signals $y_1, y_2, \ldots, y_M$, Wichert et al. suggest to use the following procedure, which minimize the occurrence of false positive detections (see [215] for more details):

1. find the largest index j for which holds $p_j \leq \frac{j}{M} q$,
2. reject the null hypothesis for all the y_i, $i = 1, 2, \ldots, j$.

With this procedure the detection of periodicity can be integrated in the Morris method, providing as such a robust mechanism that allows the computation of the elementary effects from the measures p_y, p_g, f_y, f_g, a_y and a_g when the output signal $y(\mathbf{x})$ and its perturbation $y(\mathbf{x} + \Delta_i)$ show periodicity.

The ratio

$$\rho = \frac{j}{M} \tag{6.31}$$

provides a valuable information about the oscillatory behaviour of response functions in $\mathbf{Y}$. A response $y[n]$ can be thought as a periodic signal, if the value of ρ, performed over M repetitions of $y[n]$, is greater than a threshold (usually set to 50%). This information can be integrated into the computation of the elementary effects. Given the two matrices $\mathbf{Y}$ and $\mathbf{Y}_{\Delta_i}$, where $\mathbf{Y}$ contains M responses of $y(\mathbf{x})$ and $\mathbf{Y}_{\Delta_i}$ contains M responses of $y(\mathbf{x} + \Delta_i)$, the ratios ρ for both the matrices are denoted as $\rho(\mathbf{Y})$ and $\rho(\mathbf{Y}_{\Delta_i})$. The elementary effects based on p_y (or p_g, f_y, f_g, a_y and a_g) can be thus computed as

$$d_i^{p_y}(\mathbf{x}) = \frac{\frac{p_y(\mathbf{x}+\Delta_i)}{\rho(\mathbf{Y}_{\Delta_i})} - \frac{p_y(\mathbf{x})}{\rho(\mathbf{Y})}}{\Delta}. \tag{6.32}$$

In a scenario of Ex. 6.1, the value of one of the two responses $\rho(\mathbf{Y})$ or $\rho(\mathbf{Y}_{\Delta_i})$ can be zero, therefore Eq. (6.32) will provide an infinite elementary effect. An infinite absolute value of $d_i^{p_y}$ means that at least one of the two responses $p_y(\mathbf{x} + \Delta_i)$ and $p_y(\mathbf{x})$ does

not exhibit periodicity (which means that there is high qualitative difference between the two responses and hence a high sensitivity to the parameter x_i perturbed by Δ_i). A zero absolute value of $d_i^{p_y}(\mathbf{x})$ instead means two things: both responses $p_y(\mathbf{x} + \Delta_i)$ and $p_y(\mathbf{x})$ converge to the same value (which means that there is no influence to the output of the system when the parameter x_i is perturbed by Δ_i) or both responses do not oscillate (both $\frac{p_y(\mathbf{x}+\Delta_i)}{\rho(\mathbf{Y}_{\Delta_i})}$ and $\frac{p_y(\mathbf{x})}{\rho(\mathbf{Y})}$ tend to infinite). The latter scenario must be therefore omitted from the computation of μ^*. In such occasion the elementary effect d^{s_y} can be used instead.

Case studies

7

Lt. Watson: Sir, Mr Holden is on his way back.
Lt. Cmdr. Matt T. Sherman: War is hell, Mr Watson!

Operation Petticoat, 1959

To illustrate the sensitivity analysis methodology introduced in chapter 6, we investigate two different models of GRNs containing multiple non-cooperative TFBSs in their promoters cis-regulatory regions: the *Epstein-Barr virus* (EBV) genetic switch and a synthetic oscillator based on SGRNs. The latter is considered an essential device for building logic computational structures in living cells. We apply the sensitivity analysis to the EBV model to determine the most influential input parameters of the model, while in the oscillator model, we exploited the results of the sensitivity analysis to investigate further the influence that these parameters, and especially the number of binding sites in the cis-regulatory modules, have on the model robustness.

7.1 The Epstein-Barr virus genetic switch

The Epstein-Barr virus (EBV) is widely spread in the entire adult population [217]. This virus infection has been linked to an increased risk of cancer development and recently, a notable effort to prevent this infection has been achieved [218]. The EBV is one of the most studied viruses and the majority of its related diseases have already been discovered [219]. The genome of this virus comprises 172000 base pairs and it can be found inside the nucleus of infected B-lymphocytes. Like other viruses the EBV exhibits two different states, a *proliferating* (active) and a *latent* (resting) state. Roughly

speaking, when the virus is in latent state, the cell – the B-lymphocyte – behaves normally and the virus genome duplicates together with the cell in the lysogenic cycle. When the virus is in the proliferating state (or lytic state), the cell membrane decays, the virus spreads outside the cell, it begins to duplicate in a very large number and it becomes highly infective for neighbour cells. The switch between these two states depends on the activity of two promoters, *Qp* and *Cp*, which drive the transcription of two coding regions in the virus genome, the *ebna1* and the *ebna1-6* respectively. A high Cp promoter activity indicates a high expression of the *ebna1* coding sequence, which translates into the change towards the lytic state, while a high Qp promoter activity indicates a high expression of the *ebna1-6* gene, which instead implies a change to the lysogenic (resting) state. It is known that the activity of these two promoters is mutually exclusive, which implies a switching behaviour. However, both Cp and Qp promoters are responsible of a more complex gene regulation [220]. The key component in this mutually exclusive mechanism is the Cp promoter's cis-regulatory module, which is composed of 20 clustered binding sites, known as the *Family of Repeats* (FR).

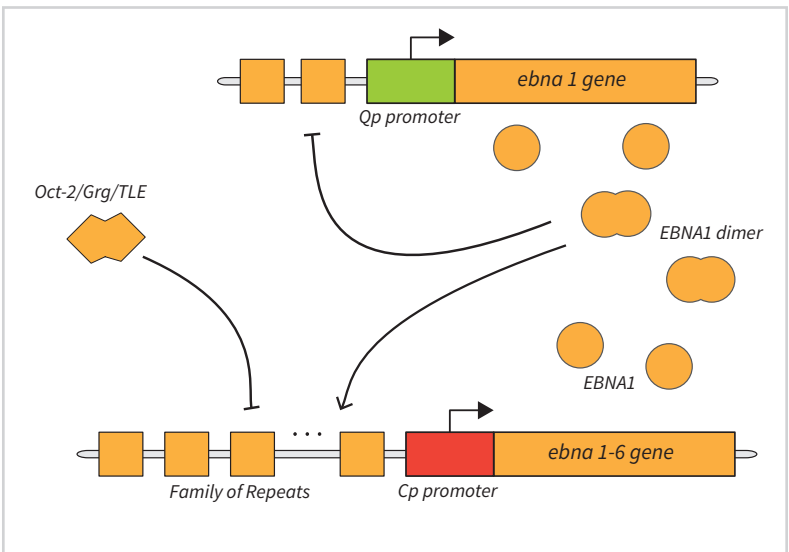


Figure 7.1

A simple scheme of the gene regulatory network responsible for the regulation of the switching mechanism in the Epstein-Barr virus infected B-lymphocytes [191]. Physically, both the coding sequences of *ebna1-6* and *ebna1* lies one behind the other inside the virus genome, i.e. the virus' DNA genome vector.

Two different TFs compete to bind over this cis-regulatory module: a dimer TF,

composed of two Epstein-Barr Nuclear Antigen-1 (EBNA1) molecules (one of the gene products of the genes regulated by the Cp and Qp promoters), which acts as powerful activator on the FR region, and the Oct2+Grg/TLE TF, an octamer powerful repressor of the Cp promoter. A simplified scheme of this regulatory mechanism is shown in Fig. 7.1. We refer to [191, 217, 220] for a more detailed overview of the complex gene regulation of these two promoters, although the dynamics of transcription regulation, i.e. the switching mechanism depicted in Fig. 7.1 that is responsible for the proliferating and resting state of the virus, can be represented (although oversimplified) as follows. The TF Oct2+Grg/TLE binds to its octamer binding sites inside the FR region [103] and acts as an inhibitor, which causes the repression of the Cp promoter. The EBNA1 can on the other hand activate the Cp promoter by binding on the same binding sites in the FR region. The EBNA1 acts as an activator for the CP promoter, but also as a repressor, when is bound on the two TFBSs in the Qp promoter region. EBNA1 inhibits the transcription of the gene product that is being regulated by the Qp promoter, i.e. the EBNA1 protein itself. But also it increases the expression of itself, by activating the Cp promoter. The EBNA1 hence has a negative feedback control via the Qp promoter and a positive feedback via the Cp promoter.

During the Cp promoter activation, the *ebna1-6* gene is transcribed into a long mRNA strand, which, among other nuclear antigenes, encodes the 619 amino acid long EBNA1 protein [104, 221]. This EBNA1 protein is involved in a dimerization process, since the DNA-TF binding over the FR requires a dimeric TF structure [222, 223]. The EBNA1 dimer can highly accelerate the transcription of the *ebna1-6* gene. This gene regulation results in a positive feedback, which occur during the lytic state. The Qp promoter instead provides a more tenuous expression of the *ebna1* gene, since its activity cannot be enhanced by activators. This results in a low EBNA1 protein concentrations, which in turn provides a weak negative feedback. To trigger the switch between the activities of the Cp and Qp promoter, it is necessary to provide a high concentration of an external signal, which is in this case the Oct2+Grg/TLE protein complex. Our task is to construct a stochastic model for describing as precisely as possible the dynamics of the EBV switch. It is obvious, that the use of the ordinary SSA approach would fail to describe the system according to Eq. (4.4), since the FR region only, gives us more than 46 billion different binding and unbinding reactions to simulate using the Alg. 3.1[1]. The main parameters of the model are shown in Tab. 7.1.

[1] More precisely 46.490.458.680 different binding and unbinding reactions.

Table 7.1

Parameters used in the Epstein-Barr virus genetic switch model.

parameter	*value*	*reference*
Nuclear volume (V)	$1.796 * 10^{-13}$ lt.	[191]
mRNA half-life $\tau_{1/2 mRNA}$	10 hr	[224]
EBNA1 amino acid length	619 – 641	[104, 221, 225]
EBNA1 – 6 coding sequence length (strain B95-8 HHV-4 from Human herpesvirus 4)	13.524[a] kb	[226]
EBNA1 - FR binding dissociation constant K_{dEFR}	$15 * 10^{-12}$ M	[191, 227]
EBNA1 transcription rate	12 bp/s	[228]
EBNA1 translation rate	20 aa/s	[164]
EBNA1 half-life $\tau_{1/2 \text{EBNA1}}$	20 – 48 hr	[229, 230]
EBNA1 dimerization rate K_{dE}	$10^{-9} M$	[191]
EBNA1 - Qp dissociation constant K_{dQ}	0.21 μM	[191, 227]
Oct2 + Grg/TLE - FR dissociation constant K_{dOFR}	2.5 – 12.5nM	[191, 231]

[a] This length is calculated from the sum of amino acids lengths of all the coded proteins of the *ebna1-6* gene, i.e. 641 (EBNA-1) + 487 (EBNA-2) + 944 (EBNA-3) + 938 (EBNA-4) + 506 (EBNA-5) + 992 (EBNA-6).

We exploited the DMSSA, described in chapter 4 for simulating the behaviour of the genetic switch depicted in Fig. 7.1. The model of the EBV can be described with the fast reactions listed in Tab. 7.2 and with the slow reactions listed in Tab. 7.3. These reactions represent a minimum-reaction model of the EBV switch, based on reactions from Def. 3.3.

We performed several simulation runs with *ParMSSA* (a multi-scale stochastic simulation engine described thoroughly in appendix A.3) by varying the initial conditions such as the number of promoters per cell (promoters copy number) and the main parameters given in Tab. 7.1. The reference values of all the parameters were set to the same same values used in [191]. The results are depicted in Fig. 7.2.

7.1.1 The system response to the external signal Oct-2+Grg/TLE

We modelled the FR region with the aid of the binding matrix $\mathbf{B}$ with a reference size of 2 promoters × 20 binding sites (a copy number of 2 Cp promoters per cell). The

Table 7.2

Fast reactions in the EBV switch model. The kinetic constants of each reactions were obtained by fixing the unbinding rate constants k_2, k_4 and k_6 (i.e. the k_{off} constants) to a reference value ($1.0\ s^{-1}$) and then computing the binding rates k_1, k_3 and k_5 (i.e. the k_{on} constants) by applying the dissociation constant K_d to the equation $k_{on} = k_{off}/K_d$ as shown in section 3.3.1. The dissociation constants for the EBNA1 and Oct2+Grg/TLE are shown in Tab. 7.1. Here dimEBNA1 states for the EBNA1 dimer and Oct2 for the Oct2+Grg/TLE inhibitor complex. Cp and Qp represent the two EBV promoters, Cp^- and Qp^- their inhibited version, and Cp^+ the activated state of the Cp promoter.

fast reactions	*type*	k_i
$Cp + dimEBNA1 \xrightarrow{k_1} Cp^+$	Cp promoter activation	$6.6667 * 10^{10}\ M^{-1}s^{-1}$
$Cp^+ \xrightarrow{k_2} Cp + EBNA1$	Cp - EBNA1 dissociation	$1.0\ s^{-1}$
$Cp + Oct2 \xrightarrow{k_3} Cp^-$	Cp promoter inhibition	$4.0 * 10^{8}\ M^{-1}s^{-1}$
$Cp^- \xrightarrow{k_4} Cp + Oct2$	Cp - Oct2 dissociation	$1.0\ s^{-1}$
$Qp + EBNA1 \xrightarrow{k_5} Qp^-$	Qp promoter inhibition	$4.7619 * 10^{6}\ M^{-1}s^{-1}$
$Qp^- \xrightarrow{k_6} Qp + EBNA1$	Qp - EBNA1 dissociation	$1.0\ s^{-1}$

maximum simulation time was set to 25 days, in order to perform a direct comparison with the model presented in [191]. We set the reaction rates of both fast and slow reactions, to the same value as found in the reviewed literature (see Tab. 7.1). The activation rule used for the Cp promoter was defined as follows: the promoter is being activated when at least eight EBNA1dimers TFs are bound to the FR region [191]. We performed several simulations and the results are presented in Fig. 7.2.

After we set the reference model with the reference parameters values from Tab. 7.1, we perturb the Epstein-Barr genetic switch model, by altering the level of the external signal Oct2+Grg/TLE, in accordance with the previous modelling strategy proposed by Werner et al. in [191]. We observed a poor genetic switch behaviour, when the model was altered with an insufficient amount of Oct2+Grg/TLE. A successful on-off switch between the two viral states in the EBV was observed with a level of Oct2+Grg/TLE of at least 500000 molecules induced after a simulation time of 12 days (see Fig. 7.2 (d)).

In Fig. 7.2 (b), the promoter binding gradient shows that the bounded repressors Oct2+Grg/TLE in the FR region do not have sufficient influence to inhibit the Cp promoter for a longer period of time. Yet, by increasing the amount of Oct2+Grg-

Table 7.3

Slow reactions in the EBV switch model. The transcription rates k_{trsc16} and k_{trsc1} were obtained via Eq. (3.15), assuming a rate limiting step for transcription elongation only. For the *ebna1-6* gene we applied the following values: a gene length of 4508 amino acids and a transcription rate of 12 bp/s, as per Tab. 7.1. For the *ebna1* gene we assume a length of 641 amino acids and the same transcription rate of 12 bp/s. Similarly for translation we adopt a rate limiting step from Eq. (3.19), with a translation rate of 20 aa/s as per Tab. 7.1 for both the transcripts: mRNA16 of the *ebna1-6* and mRNA1 of the *ebna1*. We applied the same strategy for calculating the binding and unbinding kinetic constants in Tab. 7.2, for computing the EBNA1 dimerization constant, by using the value K_{dE} from Tab. 7.1. All the degradation rates constants, $k_{degmRNA}$, k_{degE} and k_{degO}, were computed by using the inverse equation of Eq. (3.26) with the half-life values provided from Tab. 7.1.

slow reactions	*type*	k_i
$Cp^+ \xrightarrow{k_{trsc16}} Cp^+ + mRNA16$	Cp transcription	$0.002069\ s^{-1}$
$Qp \xrightarrow{k_{trsc1}} Qp + mRNA1$	Qp transcription	$0.006240\ s^{-1}$
$mRNA16 \xrightarrow{k_{trsl16}} mRNA16 + EBNA1$	mRNA16 translation	$0.005172\ s^{-1}$
$mRNA1 \xrightarrow{k_{trsl1}} mRNA1 + EBNA1$	mRNA1 translation	$0.0156\ s^{-1}$
$EBNA1 + EBNA1 \xrightarrow{k_{dim}} dimEBNA1$	EBNA1 dimerization	$1.8492 * 10^{-10}\ M^{-1}\ s^{-1}$
$dimEBNA1 \xrightarrow{k_{dis}} EBNA1 + EBNA1$	EBNA1 dimer dissociation	$10^{-8}\ s^{-1}$
$mRNA16 \xrightarrow{k_{degmRNA}} \varnothing$	mRNA16 decay	$1.9254 * 10^{-5}\ s^{-1}$
$mRNA1 \xrightarrow{k_{degmRNA}} \varnothing$	mRNA1 decay	$1.9254 * 10^{-5}\ s^{-1}$
$EBNA1 \xrightarrow{k_{degE}} \varnothing$	EBNA1 decay	$9.627 * 10^{-6}\ s^{-1}$
$dimEBNA1 \xrightarrow{k_{degdE}} \varnothing$	EBNA1 dimer decay	$9.627 * 10^{-6}\ s^{-1}$
$Oct2 \xrightarrow{k_{degO}} \varnothing$	Oct2 decay	$9.627 * 10^{-6}\ s^{-1}$

/TLE in the cell, and therefore augmenting the transition step of the Oct2+Grg/TLE, the repression of the Cp promoter become increasingly stronger, as shown in (d) (f) and (h). A full inhibition, with a gradient value reaching a negative value of 100% for a longer period (see (g) and (h)), is achieved with a high level of Oct2+Grg/TLE in the range of 500,000 molecules, which equals approximately $4.6\mu M$ concentration in the infected cell.

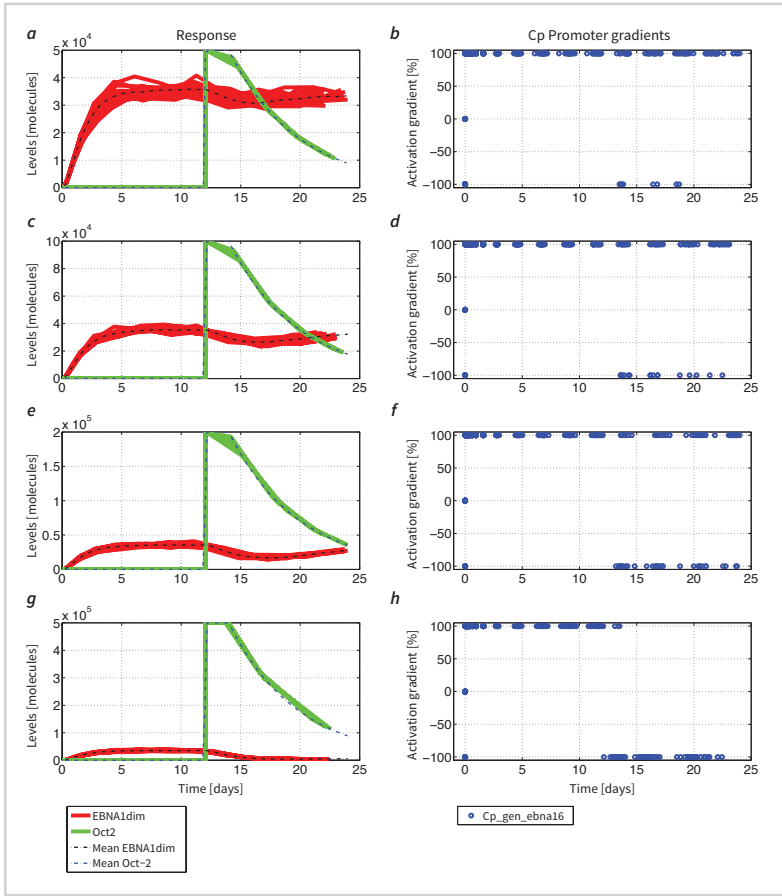


Figure 7.2

The Epstein-Barr virus genetic switch triggered by different Oct2+Grg-/TLE levels: (a) 50000 molecules, (c) 100000 molecules, (e) 200000 molecules and (g) 500000 molecules. All responses were obtained from 100 parallel simulation runs for each Oct2+Grg/TLE level. The dotted lines show the average levels of the EBNA-1 dimer and the external signal Oct2-+Grg/TLE. (b), (d), (f) and (h) show instead the Cp promoter' activation gradients g, previously described in chapter 6, over the FR region, for the same simulations. Responses and activation gradients were sampled every 5000 DMSSA points, i.e. with a sampling frequency of $2.4 \cdot 10^{-3}$ Hz.

7.1.2 *Sensitivity analysis*

In order to assess the effects of parameters' perturbations on the switch, we performed the sensitivity analysis based on the Morris method, with the enhancements proposed in chapter 6. By computing the elementary effects over the model responses and over the binding gradient of the Cp promoter we were able to identify the most important parameters of the EBV genetic switch model.

We decided to compute the parameter sensitivities of the following fundamental parameters:

1. the dimerization dissociation constant K_{dE},
2. the EBNA1 half-life $\tau_{1/2\text{EBNA1}}$,
3. the EBNA1 dimer half-life $\tau_{1/2\text{EBNA1dim}}$,
4. the Oct2+Grg/TLE half-life $\tau_{1/2\text{Oct2}}$ and
5. the Cp promoter copy number c_{Cp}.[2]

The first four parameters directly affect the kinetic constants of the degradation and dimerization reactions in the slow scale of the model, while the last parameter affects the size of the binding site matrix $\mathbf{B}$, representing the FR regions in the virus genome.

Given the number of levels $p = 10$, we defined the values' range for each of the above parameters. These intervals are listed in Tab. 7.4.

Table 7.4

The values range of each parameter in the EBV model used to perform the sensitivity analysis.

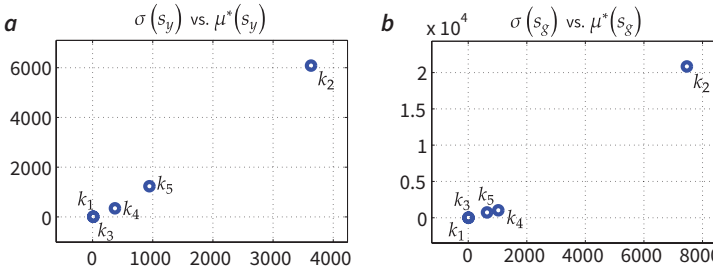
parameter	*min*	*max*	*interval type*
K_{dE}	10^{-12}	10^{-6}	logarithmic
$\tau_{1/2\text{EBNA1}}$	0.5	72	logarithmic
$\tau_{1/2\text{EBNA1dim}}$	0.5	72	logarithmic
$\tau_{1/2\text{Oct2}}$	1	48	logarithmic
c_{Cp}	1	20	linear

The interval value were defined with orthogonal sampling, i.e. one of the first step of the MSE implemented in *ParMSSA* (see section A.3.2). The results of the sampling are the matrix $\mathbf{R}^*$ (see section 5.3.1) and r different orientation matrices $\mathbf{B}^*$. We carried out the MSE, by using the DMSSA as the core simulation engine. With *ParMSSA* we executed M parallel simulations with the same set of sampled parameters for each

[2] This equals the number of total virus genome vectors present inside the infected B-lymphocyte.

Figure 7.3

The parameter sensitivities obtained from the output response using the average mean s_y of the EBNA1 (a) and the average mean s_g of the binding gradient of the Cp promoter (b). k1 is the dissociation constant K_{dE} for the EBNA1 dimerization, k2 and k3 are the half-life of the EBNA1 dimer and EBNA1 respectively, k4 is the Oct2-+Grg/TLE half-life and k5 is the Cp promoter copy number.



row of the orientation matrices $\mathbf{B}^*$. Here the number M stands for the number of OpenCL CPU-based computing units that are available on the computing platform that is running *ParMSSA*. In our case $M = 48$. Hence, being N_p the number of parameters analysed by the MSE, a total of $(N_p + 1) \cdot r \cdot M = 2304$ simulations were required for performing the Morris sensitivity analysis.

The MSE was carried out by *ParMSSA*, which set the DMSSA to record two types of responses: the response levels of the EBNA1 dimer, denoted with y, and the response of the binding gradient of the Cp promoter (i.e. the overall binding gradient of the FR region), denoted with g. By performing M parallel simulations, we obtained M responses for y, thus creating the matrix $\mathbf{Y}$, as per Eq. (6.8), and M responses for g, thus creating the matrix $\mathbf{G}$, as per Eq. (6.12), for each row of the orientation matrices $\mathbf{B}^*$. Over these two outputs we computed the elementary effects as per Eq. (5.11), by using the expected value of the EBNA1 average response, namely s_y and the average binding gradient s_g of the Cp promoter, defined as per Eq. (6.9) and Eq. (6.13) respectively. The results of the sensitivity analysis of the EBV switch model are depicted in Fig. 7.3.

Since no oscillatory behaviours were detected in the model output responses, no elementary effects were computed with the aid of the representative values f_y, p_y and a_y from Eq. (6.22), Eq. (6.23) and Eq. (6.24), respectively (or similarly with the values f_g, p_g and a_g).

The absolute mean and the standard deviation of the elementary effects were obtained over r=8 different trajectories in the parameter space. The number of levels p was set to 10 and the offset a from Eq. (5.7) was set to 3, given thus a value of $\Delta = 1/3$.

Such values for r, p and Δ are commonly suggested in literature [196]. The obtained elementary effects d_i were normalized according to Eq. (5.11).

Fig. 7.3(a) shows which parameters affect the most the average value of the model output EBNA1. Fig. 7.3 (b) shows instead which parameter affects the most the average of the Cp promoter binding gradient.

Fig. 7.3 shows that the main parameters, that affect the output concentration of EBNA1 the most, are the half-life of the EBNA1 dimer (k_2), the half-life of the Oct2-+Grg/TLE repressor (k_4) and the Cp promoter copy number (k_5). These three parameters show relatively large mean μ^* and large standard deviation σ in comparison to the other parameters. A non-zero value of μ^* means that the parameter has a non-neglecting overall influence on the output, while a high value of σ can be generally interpreted as a non-linear effect on the model output. A large σ also tells us that the influence of the parameter on the output is dependent also on the interactions with other parameters. To put in other words, the influence of the parameter on the output, with respect to other parameters cannot be considered as independent.

The EBNA1 dimer and the Oct2+Grg/TLE repressor half-life can be thought as the two main values that mainly control the behaviour of the output of the EBV switch. The degradation reactions already intuitively play a key role in the regulation of the output concentrations. Sensitivity analysis performed in Fig. 7.3 shows that the model is sensitive to the change occurring in the speed of degradation. Another important conclusion that we can extrapolate from the same results is that the initial concentration, i.e. the initial amount of the genetic material of the virus, does affect the response of the model at non-neglecting level. Which is a also a confirmation of the fact that the level of virus activity in the B-lymphocyte during the lytic state is proportional to the initial size of the virus infection.

The Epstein-Barr virus genetic switch shows a very robust behaviour, since it is able to achieve remarkable resistance to the external signal Oct2+Grg/TLE, thanks to both a negative and positive feedback motif. Our stochastic simulations performed with *ParMSSA* and the sensitivity analysis based on the Morris method confirmed these statements. Although our reaction model is quite generic for describing the complex dynamics that occur in the EBV, our simulations are still in accordance with previously reported work by Werner et al. [191].

7.2 A synthetic genetic oscillator

Synthetic genetic oscillators are among the main achievements of synthetic biology [34]. By assembling a negative feedback circuit, such as one from Fig. 2.7, it is possible to obtain oscillations in the concentration of each TF of the circuit. Here we are interested to model a similar SGRN, where each element is composed of a cis-regulatory module containing multiple non-cooperative TFBSs. An example of such circuit is depicted in Fig. 7.4. This circuit can be constructed by using TFs based on TALE [7][3]. The scheme from Fig. 7.4 also shows a possible snapshot of the system in a particular state, where the $gene_B$ is inhibited by the numerous TAL_C TFs bound on its multiple non-cooperative TFBSs. Such inhibition results in a poor presence of TAL_B TFs, which triggers the expression of the $gene_A$. The products of $gene_A$, the TAL_A TFs, eventually bind the cis-regulatory module of the $gene_C$, which is currently expressing the TAL_C TFs. The TALE TFs are assumed to have attached a repressor functional domain, such as KRAB (see section 2.6.1), i.e. a macromolecule that has a powerful inhibition effect over the gene's promoter[4]. The binding dynamics of such repressor is

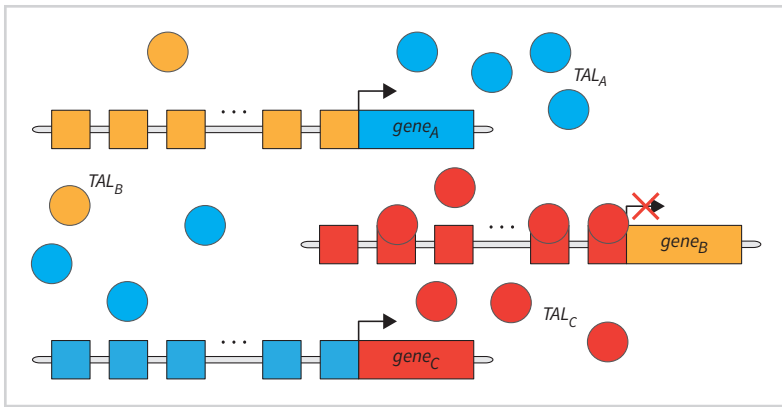


Figure 7.4

A synthetic genetic oscillator scheme base on the repressilator circuit from [34]. The circuit is composed of three synthetic genes encoding three different TALE-based TFs: TAL_A, TAL_B and TAL_C. Each of them is transcriptionally regulated by multiple TALE-TFBSs, over which, the same TFs can bind non-cooperatively.

[3] Here it is important to emphasize that such design is only suitable for eukaryotic hosts.

[4] The gene inhibition through TF repressors based on the KRAB functional domain is not that straightforward. In fact, the KRAB repression affects the chromatin structure, thus disabling the RNAP ability to bind over the constitutive promoter. Usually such inhibition is very strong even, if there is a singular KRAB based TF bound to one of the available TFBS. This property reminds the singular rule, that was introduced in chapter 2. Instead of a KRAB functional domain, we investigated an hypothetical TALE-TF that can bind

depicted in Fig. 2.3 (c). We refer to [7] for more details about the specifics of TALE TFs. Here we denote with TAL the complex repressor formed by the TALE binding domain and a strong repressor's functional domain.

Each TALE repressor protein (TAL_A, TAL_B and TAL_C), is the product of its respective gene, which is regulated by a cluster of non-cooperative TFBSs controlling a constitutive promoter. Each promoter of the circuit can be found in one of the two possible states: *active* or *repressed*. On each of TFBS the remaining types of TF can bind non-cooperatively and inhibit the expression of the gene. E.g. the product of the $gene_B$, i.e. the TAL_B, can bind on the cis-regulatory module of the gene $gene_A$ and inhibit its expression. If no products of the $gene_A$ are present, i.e. the TAL_A TFs, then no inhibition will occur on the cis-regulatory module of $gene_C$, which in turn means that its expression will results in a high presence of the TAL_C TFs. TAL_C proteins eventually bind on the multiple non-cooperative TFBSs of the $gene_B$ and inhibit the expression of the $gene_B$, which results in a poor presence of TAL_B TFs. In such occasion the $gene_A$ will eventually express itself, and the entire switching process will be repeated.

Table 7.5

The parameters of the repressilator model.

symbol	*parameter*	*value*	*reference*
K_d	TALE-DNA dissociation constant	0.1 – 10 nM	assumed range [164]
v_{trsc}	transcription elongation speed	6 – 100 bp/s	[164]
v_{trsl}	transcription elongation speed	30 – 60 bp/s	[164]
L_{TALE}	TALE genes average length	≈ 2.7 kb (110 kDa)	[7]
$t_{\frac{1}{2}_{TAL}}$	TALE half-life	4 – 24 hr	[232]
$t_{\frac{1}{2}_{mRNA}}$	TALE's mRNA half-life	2 – 20 hr	assumed value from [164]

We construct a model of the GRN depicted in Fig. 7.4, based on the multi-scale nature of such system, i.e. the presence of multiple non-cooperative TFBSs. We used 4 binding sites for each targeted TF to each of the cis-regulatory module of the TALE genes.

non-cooperatively to its TFBS and act as a "roadblock"-type repressor.

Table 7.6

Fast reactions in the repressilator model. The binding and unbinding rate k_{on} and k_{off} were obtained by simply set the k_{off} constant to a reference value of 10 s^{-1} (a typical value for TF-DNA unbinding [164]), and then compute k_{on} as $^{k_{off}}/_{K_d}$.

fast reactions	*type*	k_i
$gene_A + TAL_B \xrightarrow{k_{on}} gene_A^-$	DNA-TF repressor binding	$1.0 \cdot 10^{10}\ M^{-1}s^{-1}$
$gene_A^- \xrightarrow{k_{off}} gene_A + TAL_B$	DNA-TF repressor unbinding	$1.0 \cdot 10^{1}\ s^{-1}$
$gene_B + TAL_C \xrightarrow{k_{on}} gene_B^-$	DNA-TF repressor binding	$1.0 \cdot 10^{10}\ M^{-1}s^{-1}$
$gene_B^- \xrightarrow{k_{off}} gene_B + TAL_C$	DNA-TF repressor unbinding	$1.0 \cdot 10^{1}\ s^{-1}$
$gene_C + TAL_A \xrightarrow{k_{on}} gene_C^-$	DNA-TF repressor binding	$1.0 \cdot 10^{10}\ M^{-1}s^{-1}$
$gene_C^- \xrightarrow{k_{off}} gene_C + TAL_A$	DNA-TF repressor unbinding	$1.0 \cdot 10^{1}\ s^{-1}$

We therefore defined a similar two-scaled reaction system as for the EBV model in the previous section, based on Def. 4.1. The main parameters of the model are listed in Tab. 7.5. The fast reactions of the system are defined in Tab. 7.6, while the slow reactions are shown in Tab. 7.7.

All the three TALE TFs, are roughly of the same length (2.7 kb). Hence they may exhibit the same slow kinetics for transcription, translation and degradation, which means that they may share the same kinetic constants. This is of course a simplification, however this presumption allows to make the model as simple as possible for performing the stochastic simulations faster.

The fast reactions are similarly sharing the same kinetics, due to the similar DNA-binding dynamics of all the three TALE TFs. In other words they all share the same reaction kinetics for the TF-DNA binding and unbinding, as per Def. 3.3.

An example of the response of this SGRN, i.e. the oscillation in the concentration of the three TFs, is shown in Fig. 7.5 (a). The concentration of each TF oscillates with a period that can span from 60 to 140 hours. The main frequencies of these oscillations are the same for all the three components of the repressilator, as shown in the frequency spectrum of the responses, see Fig. 7.5 (c).

Fig. 7.5 (b) shows the binding gradient plot for all the three genes of the model. As predicted, we observed an oscillatory change in the value of each binding gradient. The

Table 7.7

Slow reactions in the repressilator model. The transcription rate constant k_{trsc} can be computed from Eq. (3.15) by including only the elongation factor, i.e. $k_{\text{trsc}} = {}^{v_{trsc}}/_{L_{TALE}}$. The translation rate k_{trsl} can be similarly computed from Eq. (3.19) as $k_{\text{trsl}} = {}^{v_{trsl}}/_{L_{TALE}}$. Here we implied that the length of the mRNA transcripts is of the same length of the TALE gene. The degradation rates $k_{\text{deg-mRNA}}$ and k_{deg} were derived from Eq. (3.26).

slow reactions	*type*	k_i
$\text{gene}_A \xrightarrow{k_{\text{trsc}}} \text{gene}_A + \text{mRNA}_{\text{TAL}_A}$	TAL_A gene transcription	$1.7857 \cdot 10^{-2}\ s^{-1}$
$\text{gene}_B \xrightarrow{k_{\text{trsc}}} \text{gene}_B + \text{mRNA}_{\text{TAL}_B}$	TAL_B gene transcription	$1.7857 \cdot 10^{-2}\ s^{-1}$
$\text{gene}_C \xrightarrow{k_{\text{trsc}}} \text{gene}_C + \text{mRNA}_{\text{TAL}_C}$	TAL_C gene transcription	$1.7857 \cdot 10^{-2}\ s^{-1}$
$\text{mRNA}_{\text{TAL}_A} \xrightarrow{k_{\text{deg-mRNA}}} \varnothing$	mRNA degradation	$4.8135 \cdot 10^{-5}\ s^{-1}$
$\text{mRNA}_{\text{TAL}_B} \xrightarrow{k_{\text{deg-mRNA}}} \varnothing$	mRNA degradation	$4.8135 \cdot 10^{-5}\ s^{-1}$
$\text{mRNA}_{\text{TAL}_C} \xrightarrow{k_{\text{deg-mRNA}}} \varnothing$	mRNA degradation	$4.8135 \cdot 10^{-5}\ s^{-1}$
$\text{mRNA}_{\text{TAL}_A} \xrightarrow{k_{\text{trsl}}} \text{TAL}_A + \text{mRNA}_{\text{TAL}_A}$	TAL_A mRNA translation	$4.2857 \cdot 10^{-2}\ s^{-1}$
$\text{mRNA}_{\text{TAL}_B} \xrightarrow{k_{\text{trsl}}} \text{TAL}_B + \text{mRNA}_{\text{TAL}_B}$	TAL_B mRNA translation	$4.2857 \cdot 10^{-2}\ s^{-1}$
$\text{mRNA}_{\text{TAL}_C} \xrightarrow{k_{\text{trsl}}} \text{TAL}_C + \text{mRNA}_{\text{TAL}_C}$	TAL_C mRNA translation	$4.2857 \cdot 10^{-2}\ s^{-1}$
$\text{TAL}_A \xrightarrow{k_{\text{deg-mRNA}}} \varnothing$	TAL_A degradation	$3.209 \cdot 10^{-5}\ s^{-1}$
$\text{TAL}_B \xrightarrow{k_{\text{deg-mRNA}}} \varnothing$	TAL_B degradation	$3.209 \cdot 10^{-5}\ s^{-1}$
$\text{TAL}_C \xrightarrow{k_{\text{deg-mRNA}}} \varnothing$	TAL_C degradation	$3.209 \cdot 10^{-5}\ s^{-1}$

free TFs are able to quickly bind on the multiple binding sites of each cis-regulatory module, due to the low value of the dissociation constant (in the *nM* region), thus providing a fast saturation of the binding site cluster. This resulted in a total (100%) repression of the TALE genes. Also, as predicted, we observed slower TF unbinding events, leaving a gradual but increasing number of free binding sites. As such the binding gradients reached the 100% limit, thus allowing the TALE genes to be expressed.

The response of the repressilator is remarkably robust, despite the fluctuation in the parameters values that can be apportioned by the designer. To understand how

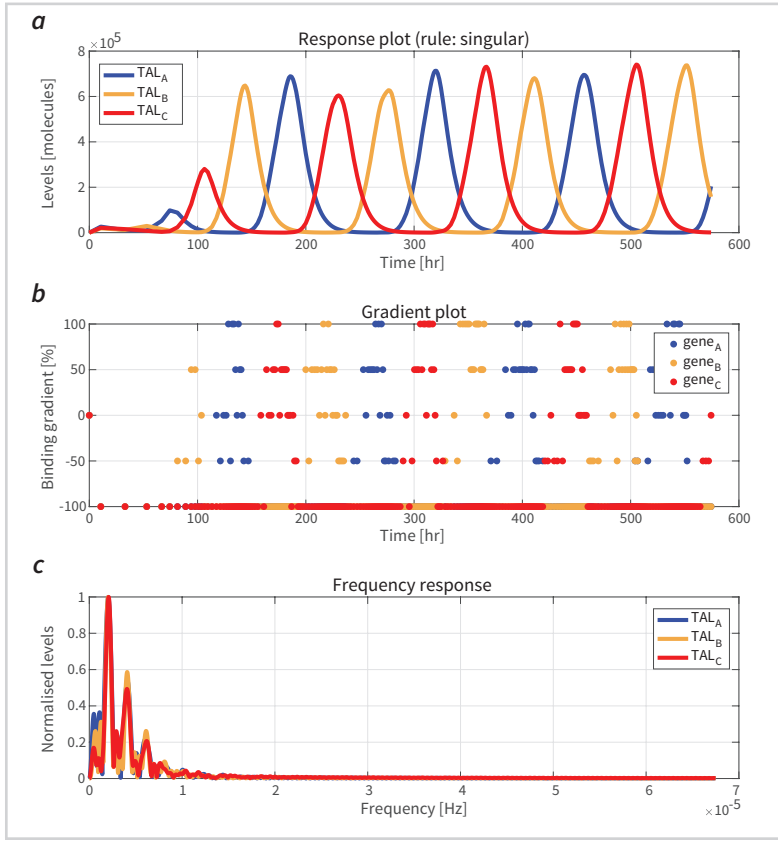


Figure 7.5

An example of repressilator response trajectory (a), the binding gradient plot for the cis-regulatory module of the $gene_A$ (b) and the response's frequency spectrum (c), of a simple simulation run of the multi-scale model defined with reactions from Tab. 7.7 and Tab. 7.6. Each TALE gene of the model contains 4 binding sites for the targeted TFs. The simulation run was set for testing the singular activation rule. The dissociation constant K_d was set to 1 nM, the transcription elongation speed v_{trsc} to 50 bp/s, the translation elongation speed v_{trsl} to 20 aa/s, the mRNA half-life to 4 hr and the TALE half-life to 6 hr.

such fluctuations can affect the model response, we performed the sensitivity analysis proposed in chapter 6 with the aid of *ParMSSA*.

7.2.1 *Sensitivity analysis*

By performing the Morris sensitivity analysis with the elementary effects described in chapter 6, we can provide different parameter sensitivity values, that can be used to assess the global robustness of the repressilator. Each parameter sensitivity can directly refer to the amount of influence that the parameter has on the model outputs on aver-

age. Larger the influence is, less robust is the model against the change in the parameter value. The parameters sensitivities, can help to facilitate further investigation, i.e. to identify which parameters drive the response of the repressilator to an unwanted behaviour. This can be invaluable in a scenario where only computational models can be used to represent the system.

Table 7.8

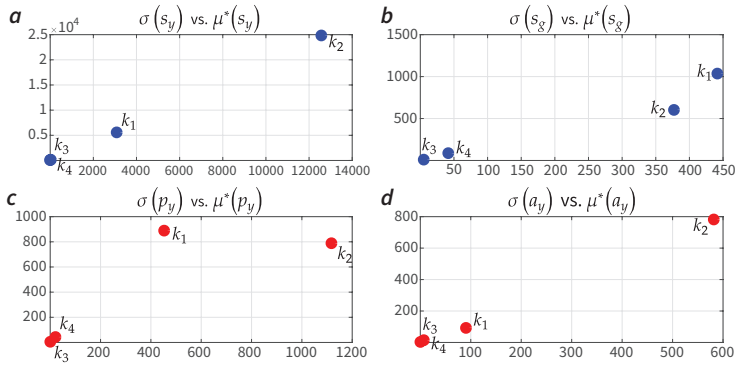
The values range of each parameter in the TALE repressilator model used to perform the sensitivity analysis.

parameter	*symbol*	*min*	*max*	*interval type*	*reference*
TALE half-life	$\tau_{1/2\text{TAL}}$	4 *min*	72 *hr*	logarithmic	[164, 232]
mRNA half-life	$\tau_{1/2\text{mRNA}}$	2 *min*	24 *hr*	logarithmic	[164]
$gene_{\text{A}}$ copy #	c_A	1	100	linear	[233, 234]
TAL_{A} # TFBS	n_A	1	32	linear	set by design

In order to perform the Morris sensitivity analysis of the repressilator circuit, we selected four basic parameters of the model, which could be also physically perturbed in a real environment. These parameters are the proteins' and mRNAs' half-lives, the genes copy numbers and the number of binding sites in the genes' cis-regulatory modules. The values ranges for each of these parameters are shown in Tab. 7.8.

As in the case study of the EBV genetic switch, we selected (observed) just two outputs of the model: the TAL_{A} concentration (y) and the binding gradient of the $gene_{\text{A}}$ (g), on which the parameter sensitivities were computed. Additionally to the measures used in the EBV model for computing the elementary effects, i.e. the average responses s_y and s_g, we used also the average principal period in the model response's frequency spectrum and the average of the maximum amplitude of the principal frequency of the response, computed over multiple repetitions. The obtained elementary effects d_i were normalized according to Eq. (5.11).

The sensitivity analysis of the model defined with reactions depicted in Tab. 7.6 and Tab. 7.7, was carried out with *ParMSSA*, by applying (by default) the singular activation rule to all the promoters of the repressilator. The results are shown in Fig. 7.6. The value r of the matrix $\mathbf{R}^*$ was set to 30. The parameter intervals were divided in $p = 10$ sub-spaces. The entire sensitivity analysis took approximately 2.5 days for computing all the $(N_p+1)\cdot r\cdot M = 5\cdot 30\cdot 48 = 7200$ required simulations needed to calculate the



The standard deviation versus the absolute mean of the elementary effects of: (a) the average mean of the TAL_A (s_y), (b) the average mean of the binding gradient of the $gene_\text{A}$(s_g), (c) the average mean of the period of the component with the highest amplitude in the frequency spectrum of the responses TAL_A (p_y), and (d) the average amplitude of the principal frequencies in the frequency spectrum of the responses TAL_A (a_y), for each of the parameters listed in Tab. 7.8. The parameter k_1 stands for the TFs' half-life, k_2 is the mRNA' half-life, k_3 is the copy number of the $gene_\text{A}$ (the initial amount), and k_4 is the number of non-cooperative TFBSs of $gene_\text{A}$.

Figure 7.6

The parameter sensitivities of the repressilator circuit.

parameter sensitivities, i.e. μ^* and σ, which are shown in Fig. 7.6. Fig. 7.6 (a) shows the $\sigma(s_y)$ vs $\mu^*(s_y)$ graph of the elementary effects computed with the average responses s_y. By observing the average of the absolute value of the elementary effects and their standard deviations, we can see that the input parameters k_1 and k_2, i.e. the TFs' and mRNAs' average half-life, have the largest influence on the model output responses, i.e. on the concentration of the output signal TAL_A as well as on the binding gradient of the $gene_\text{A}$. The two parameter sensitivities $\sigma(s_y)$ and $\mu^*(s_y)$ of the parameter k_1 tell us, that the value of the TFs' half-life strongly influence the output value of the model, i.e. the concentration of the TFs. This is in accordance with our intuitive interpretation of the dynamics of GRNs: TFs are produced by the gene expression process and they are consumed by the degradation process. Fast or slow degradation will strongly affect the concentration of the free TFs in the cell. The value of the half-life defines the kinetic constant of the degradation reaction, as per Eq. (3.26). The

Morris sensitivity analysis shows that the repressilator model response is affected by the perturbations in the degradation speed, i.e. by a change in the value of the TFs half-life. The parameter sensitivity $\sigma(s_y)$ however, tells us, that the TFs half-life has also a large "spread", i.e. the effect of this input parameter on the model output is non-linear, which means, that the parameter k_1 can affect profoundly the model output TAL_A, even with a small perturbation in its value. The same conclusion can be derived by observing the sensitivity values of the elementary effects obtained with the values $\sigma(p_y)$ and $\mu^*(p_y)$, i.e. the standard deviation and the mean of the elementary effects obtained with the period of the main component in the frequency spectrum of the model response TAL_A.

A similar conclusion can be derived from the parameter sensitivities obtained by the binding gradient of the $gene_\mathrm{A}$. The parameters with the highest influence are the TFs and mRNA half-lives involved in the degradation process.

Fig. 7.6 was obtained by considering only the singular inhibition rule, since, as we will show in the next section, no oscillations were detected in the model response of the repressilator, if is driven by the all-or-none or the additive inhibition rule. The computation of the elementary effects using the average principal period p_y and the average principal amplitude a_y were carried out solely for those scenarios in which both $p_y(\mathbf{x})$ and $p_y(\mathbf{x} + \Delta_i)$ (or $a_y(\mathbf{x})$ and $a_y(\mathbf{x} + \Delta_i)$) exhibit oscillations (non oscillatory responses were discarded).

Equipped with this findings, we began investigating further the influence of multiple TFBSs to the repressilator response. We then analysed the applicability of the parameter sensitivity of the TALE half-life as a metric for estimating the robustness of the model response' oscillations.

7.2.2 Multiple transcription factors binding sites implications

We proposed an experiment in which two groups of simulations were performed. In the first configuration the TFs half-life was set to a default referenced value of 6 hr (in accordance with Tab. 7.5). In the second configuration, the same half-life was set to a much lower value, i.e. 3,75 min, which oversteps the boundaries, albeit slightly, of the interval used in the sensitivity analysis (see 7.8)[5]. We performed three simulations over

[5] Lowering the half-life means to increase the protein degradation rate. Increasing (or decreasing) the speed of the protein degradation physically, is not a trivial task. However, there are few techniques in molecular biology capable of changing the rate of degradation through the use of specific proteases, as described in [235].

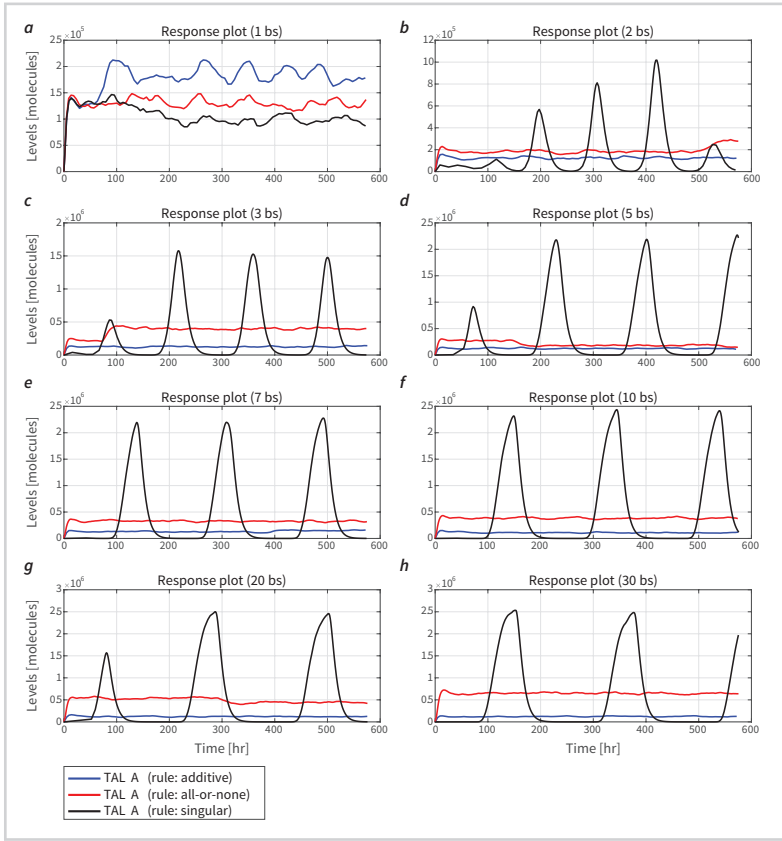


Figure 7.7

Simulations responses of three different activation rules, i.e. additive, all-or-none and singular. The simulations were carried on the model from Fig. 7.4 with a different number of TFBSs in the cis-regulatory module of all the three TALE genes. The half-life of the model output proteins is set to 6 hr.

both configurations, each time by applying a different activation rule and each time by applying a new value for the number of TFBS to place in each cis-regulatory module of the circuit. Such experiment provides a broad overview and a direct comparison between the oscillatory scenarios of the repressilator. We applied the following three inhibition rules:

1. additive,

2. all-or-none and

3. singular.

Let's denote with $f_U(0,1)$ the random function f, which returns an equally distributed random number in the interval . And let $\mathbf{b}$ be a row (the binding vector) of the system binding site matrix. The rules listed above have been previously described in chapter 2 as activation rules and in the case of the repressilator can be formalized as follows:

$$h_{add}(\mathbf{b}) = \begin{cases} 1; & \text{if } |b^{m+}| = 0 \ \wedge \ |b^{m-}| = 0 \\ a; & \text{otherwise} \end{cases}, \tag{7.1}$$

$$h_{aon}(\mathbf{b}) = \begin{cases} -1; & \text{if } |b^{m-}| = n \\ 1; & \text{otherwise} \end{cases}, \tag{7.2}$$

$$h_{sin}(\mathbf{b}) = \begin{cases} -1; & \text{if } |b^{m-}| > 0 \\ 1; & \text{otherwise} \end{cases}, \tag{7.3}$$

where

$$a = \begin{cases} -1; & \text{if } f_U(0,1) < \frac{|b^{m-}|}{n} \\ 1; & \text{otherwise} \end{cases}.$$

We did not observed any periodicity in the model responses under the additive and all-or-none rule. The oscillations were detected only when the TAL repression was driven with the singular rule.

The results of the first group of simulations are shown in Fig. 7.7. Fig. 7.7(a-h) show the outputs TAL_A responses obtained with different numbers of TFBSs in each TALE's gene of the repressilator model. One can observe that, with the exception of Fig. 7.7(a), on every configuration the singular rule exhibits oscillations robustly. The repressilator model that comprises only one TFBS does not exhibit any oscillations in the outputs. This was also confirmed by the frequency spectrum of Fig. 7.7(a) (not shown). We suppose that the main reason for this discrepancy is the lack of non-linearity in the reaction system of Def. 3.4 as also described in [236]. By augmenting the non-linearity, or by increasing the time delay between the reactions of transcription (or translation) of each repressor and the binding of the respective repressor to its binding sites (see [237]), we speculate that the model would eventually oscillate.

We noticed that the frequency of the oscillations decreases with the increase of the number of available TFBS. The frequency spectrum (not shown) confirmed this trend. The periods of the frequency components with the higher amplitudes are shown in

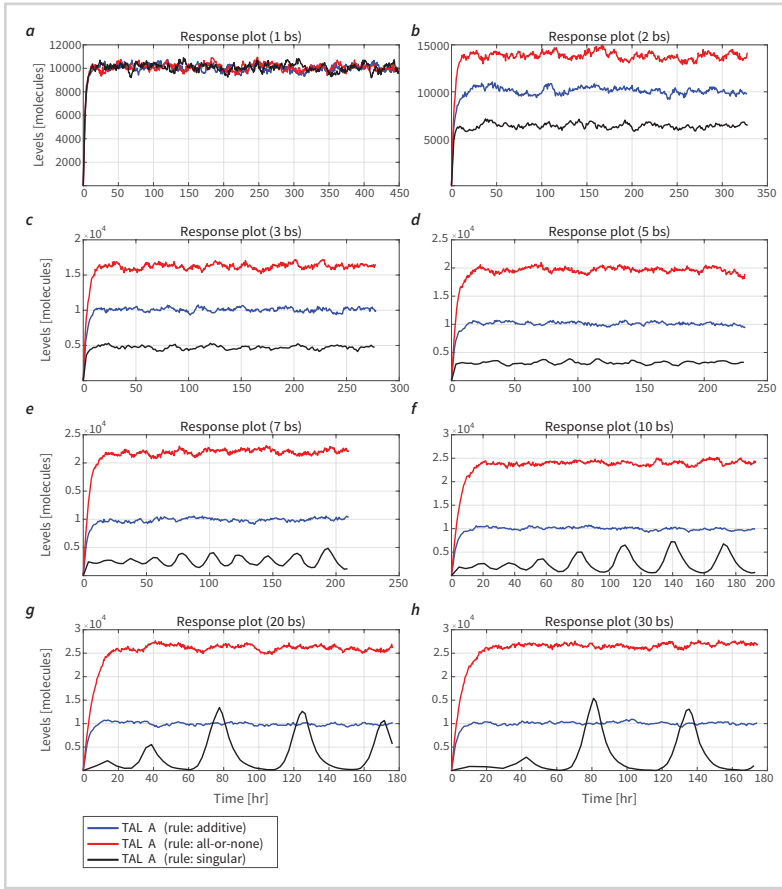


Figure 7.8

Simulations responses of three different activation rules, i.e. additive, all-or-none and singular. The simulations were carried on the model from Fig. 7.4 with a different number of TFBSs in the cis-regulatory module of all the three TALE genes. The procedure of Wichert for detecting oscillations described in section 6.3.3, detected periodicity in the responses obtained with more (or equal) than 2 TFBSs.

Fig. 7.9.

In the second group of simulations we ran the same parameter configurations, as in the previous group, with the only difference that, this time we perturbed the system with its most sensitive parameter $\tau_{1/2\text{TAL}}$. The results are shown in Fig. 7.8. The perturbation has been carried out by simply decreasing the value of $\tau_{1/2\text{TAL}}$ to 3,75 min, which translates to a faster degradation by two orders of magnitude. We observed that

Table 7.9

A comparison of the oscillation main periods (in hours) of the repressilator response function, obtained from the simulations performed with the reference parameters values (i.e. slow degradation) and with a fast degradation rate (i.e. fast degradation) under the singular repression rule. Very low frequencies are achieved by the response function, if higher numbers of non-cooperative binding sites are used in the cis-regulatory module of promoters. Shortest periods are achieved by the response function in the case of having a fast degradation rate, especially if higher numbers of non-cooperative TFBSs are placed in the cis-regulatory module.

b. sites	periodicity (slow degrad.)	periodicity (fast degrad.)
1	no	no
2	107.89	partial
3	135.66	partial
5	165.17	24.58
7	171.19	25.32
10	198.63	32.35
20	203.07	46.41
30	221.16	53.25

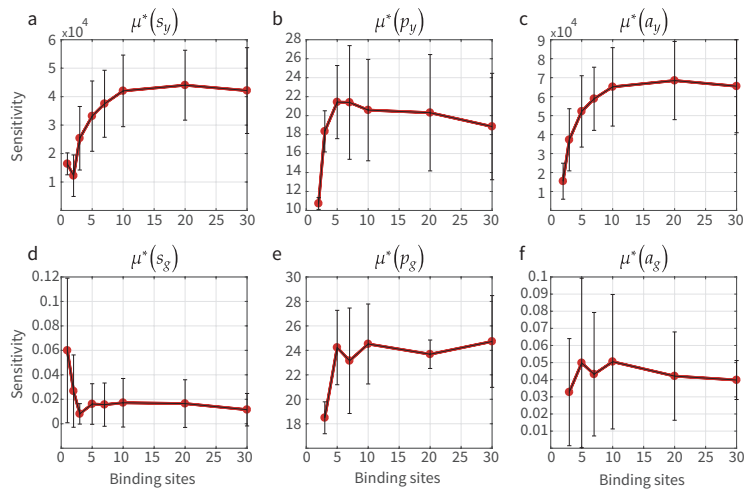
also in this scenario the system exhibits oscillations.

Fig. 7.8(a) shows that also in this case, a single TFBS is insufficient to regulate the gene inhibition properly, as the concentration of TALE proteins do not oscillate. A partial oscillation was detected in the simulation with two and three available TFBSs, see Fig. 7.7(b) and (c). In all other cases, the oscillations resulted to be up to 30 fold faster than the oscillations obtained with the reference value of $\tau_{1/2\mathrm{TAL}}$ (see Tab. 7.9). Also in this scenario, the frequency of oscillation decreases with the increase of available TFBS.

We suspect that this decreasing frequency effect (or increasing periodicity), is due to an intense activity of TFs in proximity of the cis-regulatory module of each gene in the circuit. Having a greater number of bound TFs (because of an increased number of TFBSs), means that there are less free floating TFs. Multiple TFBSs protect bound TFs from degradation and hence their concentration is less susceptible to drop quickly, thus increasing the period of oscillations. We did not investigate the possibility of allowing the degradation of bound TFs, since it would require a modification of Def. 4.2.

7.2.3 Robustness estimation

After performing these two groups of simulations, we tackled the challenge of estimating the sensitivity of the oscillations as a function of the number of TFBS. Such sensitivity estimation can be used as a valid metric for validating the robustness of the oscillations. We designed a Morris screening experiment in which the elementary effects of the reference model of the repressilator are computed solely for the parameter $\tau_{1/2\text{TAL}}$. This experiment was performed over eight different numbers of TFBSs: 1, 2,



The sensitivity analysis of the repressilator circuit based on the parameter $\tau_{1/2\text{TAL}}$ for eight different numbers of TFBSs: 1, 2, 3, 5, 7, 10, 20 and 30 in the cis-regulatory module of $gene_A$, $gene_B$ and $gene_C$, according to the singular inhibition rule of Eq. (7.3). (a), (b) and (c) show respectively the parameter sensitivity $\mu^*(s_y)$, obtained from the average response, (b) the parameter sensitivity $\mu^*(p_y)$, obtained from the principal period and (c) the parameter sensitivity $\mu^*(a_y)$, obtained from the average amplitude of the principal frequency of the TAL_A protein response. (d), (e) and (f) show respectively the parameter sensitivity $\mu^*(s_g)$, obtained from the mean value of the binding gradient, the parameter sensitivity $\mu^*(p_g)$, obtained from the principal period and the parameter sensitivity $\mu^*(a_g)$, obtained from the average amplitude of the principal frequency of the binding gradient response of $gene_A$. All sensitivities values are depicted with their 95% confidence interval.

Figure 7.9

Robustness analysis of the repressilator circuit.

3, 5, 7, 10, 20 and 30. We began by setting the same configuration of the model as used to obtain the response depicted in Fig. 7.7(a). Then we computed eight elementary effects from the Morris experiment by perturbing only the $\tau_{1/2\text{TAL}}$ parameter (i.e. a matrix $\mathbf{R}^*$ of eight different orthogonal samples of $\tau_{1/2\text{TAL}}$ is constructed for such task). We computed next $\mu^*(s_y)$, $\sigma(s_y)$, $\mu^*(s_g)$ and $\sigma(s_g)$. We repeat this procedure only for the singular inhibition rule and over all the eight models, differing from each other by 1, 2, 3, 5, 7, 10, 20 and 30 binding sites[6]. The results are shown in Fig. 7.9. In this

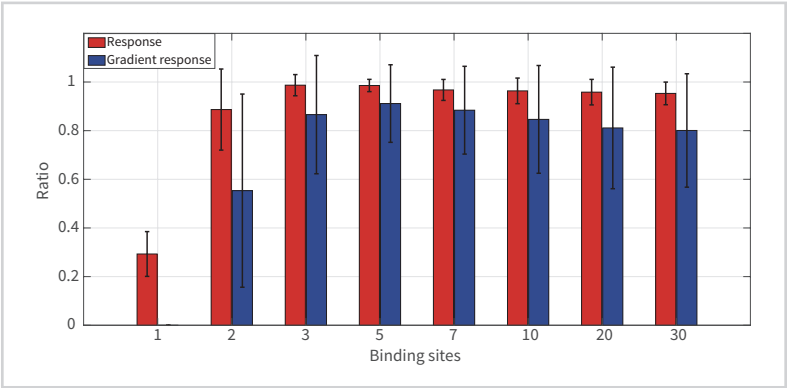


Figure 7.10

Ratio of oscillatory patterns in the robustness analysis experiment.

experiment the obtained elementary effects d_i were not normalized according to Eq. (5.11), since the standard deviation of the input factors σ_{xi}, due to the analysis of only one parameter, is always zero, which hence produces null elementary effects. However, they were computed via Eq. (6.32), so the oscillatory ratio was included (the values of ratio are shown in Fig. 7.10).

Fig. 7.9 shows the results of the Morris sensitivity analysis for the $\tau_{1/2\text{TAL}}$ parameter. We performed six different computations of the sensitivity based on the mean response, the principal period and the mean amplitude of the principal frequency, from both the response of the TAL_A protein and the binding gradient of $gene_\text{A}$. The sensitivity based on the half-life, as a value of the model's sensitivity appears to be non-linear and monotone with the number of TFBSs. This can be alternatively interpreted as the robustness of the model decreasing with the number of TFBSs. Although the sensitivities based

[6] The *ParMSSA* engine is limited to 32 TFBSs per promoter, due to implementation constraints.

on the binding gradient show that no prominent influence is provided by the increment of TFBSs (i.e. the sensitivity is almost linear), a high number of TFBSs has still a profound effect in the response of the TAL_A protein (same for TAL_B and TAL_C). Despite that these sensitivities do not decrease with the number of binding site, the periodicity and the stability of the oscillations do, as shown in Tab. 7.9. This can be interpreted as the *tunability* of the parameters increases with the number of TFBSs.

Since the sensitivities were computed solely for the responses that exhibit periodicity and since the matrix $\mathbf{R}^*$ was set to produce only 8 different trajectories, many elementary effects have relatively high standard deviation. However, the results shown in Fig. 7.9 can help the SGRNs designer to determine the optimal design and structure of the SGRNs, based on the requirements for a particular frequency selection.

Conclusions

8

Lt. Cmdr. Matt T. Sherman: To paraphrase Mr. Churchill: 'Never have so few stolen so much from so many.'

Operation Petticoat, 1959

In this dissertation we developed an approach for performing stochastic simulations of GRNs containing promoters with multiple non-cooperative TFBSs. Such GRNs are difficult to modelled stochastically with current approaches, because of the high number of reactions that need to be accounted. We also developed a methodology for performing the sensitivity analysis of such systems efficiently, despite the challenging computational complexity.

We applied our approach to the Epstein-Barr genetic switch model and to the genetic oscillator, known as *repressilator*. These two models contain a relatively high number of non-cooperative TFBSs, making their stochastic modelling particularly difficult to perform with the classical techniques. We showed that the application of our approach to such models is straightforward and, due to the parallel implementation of the algorithm DMSSA, the results can be obtained quickly.

We showed that the number of non-cooperative TFBSs inside the cis-regulatory modules of promoters plays an important role in the tuning of the main frequency of oscillations in the repressilator circuit.

The necessity of providing accurate modelling of biological systems, has stimulated the development of the stochastic modelling also for the most complex and challenging systems.

8.1 The main contributions of the dissertation

We achieved the following two contributions:

- We developed a new multi-scale stochastic simulation algorithm called DMSSA capable to cope with the stiffness and with the large combinatorial complexity of chemically reacting systems of GRNs that contain multiple non-cooperative transcription factors binding sites. We exploit the simplicity of the stochastic simulation algorithm for performing accurate simulations of such complex GRNs, achieving unprecedented success. We included the algorithm inside a parallel simulation engine called *ParMSSA*, built for simulating complex systems and synthetic biology models of GRNs with any feasible number of binding sites. The validity of the method was proved on a simple multi-scale reaction model and compare with the results obtained by a less effective SSA.

- We established a methodology for tackling the sensitivity analysis of stochastic multi-scale models, based on the Morris method. The *state-of-the-art* sensitivity analysis approaches lack the capability to investigate stochastic multi-scale models. By establishing our methodology we filled this gap, and we showed how this methodology can be applied to both systems and synthetic biology models. By providing a parallel engine for performing multiple instances of the stochastic simulations with DMSSA we encourage the utilization of the Morris screening experiment also for the computationally more demanding stochastic simulations.

8.2 Future research directions

The results obtained with the DMSSA on the repressilator circuit should be properly assessed and compared also with experimental results. A physical implementation of the circuit depicted in Fig. 7.4 is within the reach of any synthetic biology laboratory, see the similar constructs implemented by Lebar and Jerala in [7].

We think that our DMSSA approach presented in this thesis, coupled with the sensitivity analysis of the enhanced Morris method, should find several interesting applications in systems and synthetic biology, since it allows manageable computational complexity of stochastic simulations of GRNs containing multiple non-cooperative

TFBSs. This approach should be used wherever the stiff property of a chemically reaction network would prevent the use of a classical simulation approach (as with SSA) for performing stochastic simulation of the system.

The results of our sensitivity analysis methodology can be exploited in future processes of the design of logic structures, based on SGRNs with multiple non-cooperative TFBSs. With the obtained parameters sensitivities it is possible not only to identify the most important parameters, but also to provide a rough estimation of the robustness of the multi-scale models of GRNs. Such information can be used to construct robust metrics to improve the quality of the design of biological systems with information processing capabilities.

The model definition of the reaction networks that the DMSSA uses, does not consider any intermediate step that occurs between the processes of gene regulation and expression. For instance in eukaryotes, once it is formed, the $mRNA$ must reach the ribosome in the cytosol. This can be viewed as a time-delay step that occurs between the reactions of transcription and translation. The SSA, and as well the DMSSA can cover this shortcoming by including an additional reaction between these two reactions. Such approach should be implemented when the delay heavily affects the rate-limiting step of both transcription and translation. In our models we assume that this delay is at least an order of magnitude smaller than the total transcription or translation time. We refer to [183] for additional information about an example of implementation of the delayed-SSA, that can cope with such delays, and that might affect the correctness of the conventional SSA.

The DMSSA is limited regarding the representation of the possible DNA binding dynamics. The DMSSA was developed to simulate mainly the dynamics of non-cooperative binding. An important improvement would be to allow the DMSSA to tackle also cooperative binding.

Computationally, the DMSSA can be further improved by using, instead of the classical SSA, a different variant of the SSA, or even better, one of the possible approximation technique (such as the τ-leaping) as the simulation engine for both the scales. The nested SSA in fact can theoretically be replaced by any of its variants (tau-leaping, etc).

8.3 Final remarks

It is now time to unveil what our trio of objects (the bacteria, the fish hook and the lonely tree in an empty field) has in common.

When we observe a lonely tree, which is totally devoid of any autonomous and coordinated movement, growing alone on an arid and brittle field, which is an environment that at first glance seems less inclined to accommodate the development of plants, we will probably get surprised and we would not probably stop asking the question: "How did it do?". The answer of course is adaptation, but as being pedantic scientists, we would probably dive into the research of the main cause that allows the tree to grow in such hostile place. Besides, we can all agree upon the fact, that the ability to adapt to an environment is inherent in the organisms not only of the kingdom Plantae but also for Animalia.

The evolution can be seen as a decrease in the values of the sensitivity of the organism, that favours the adaptation in an environment, which at first was less prone to the development of the observed living being. Smaller the sensitivity is, greater the robustness of the organism would be against a hostile environment. The environmental conditions define the limits of the sensitivity values of each living being. Sudden changes (or big perturbation) in the environment properties may lead to an increased sensitivity response of the biological system, causing possible serious damage to the organism itself. The tree for instance will only grow as much as the wind, that usually blow on this field, will allow, meaning that, if the tree would grow too high then, with a sudden change of the wind speed over the reference interval, the tree's branches or the main trunk will break, posing as such a critical survival risk to the organism. The evolution would therefore "adjust" the maximum tree high for the next generations of trees growing in that region of space according to the reference values of the wind speed. If such adjustment will not be possible, then the existence of the tree (and any of its future descendants) would be questioned. Otherwise, the tree would adapt to the adverse conditions by developing a higher robustness, i.e. decreasing its sensitivity, to the environmental perturbations.

In a broaden view we can say that both the bacterium and the lonely tree in the arid field are two striking examples of adaptation of living organisms to hostile environmental conditions. The tree's branches bend when it is hit by the wind, while the bacterium uses chemotaxis to escape or react to a strong change in the environmental

conditions. Both such behaviours of the organisms can be specified through parametric values that can be framed inside specific intervals. An off-the-shelf value may result in a poor adaptation, for example, it could lead to a fatal event inside bacterium for its metabolism.

The sensitivity of a tree may be represented as its tolerance to the bending according to the force of the wind. This value is hence imposed externally by the environment. In the bacterium, however, sensitivity can be attributed to multiple factors, both external and internal, which makes the bacterium a challenging model for performing the sensitivity analysis.

The fishing hook instead represents an example of inanimate object, which cannot adapt to environmental changes. Its ability to withstand strong forces depends entirely on its design and manufacture. Hence, the sensitivity of this object can be entirely attributed to the quality of its construction. The larger the force that the hook will be able to support, the lower its sensitivity will be, ergo, more robust the hook will be. The sensitivity is therefore inversely proportional to the robustness. It can be determined by the maximum force we think the hook will have to endure. This force clearly represents an external event and can be similarly measured as the force of the wind that bends the branches of the tree.

Hence yes, a high robustness towards the environment is one of the key elements that all the objects of our trio have in common.

Appendices

A

A.1 Multi-scale stochastic simulation techniques

Here we review the three simulation techniques mentioned in chapter 3 that currently represent the *state-of-the-art* approaches for stochastic multi-scale modelling and simulation of GRNs.

A.1.1 Slow-scale stochastic simulation algorithm

Authors in [94] developed the *slow-scale stochastic simulation algorithm* (ssSSA) in order to tackle the stiffness in chemically reacting systems of the type defined in Def. 3.10. The ssSSA is an improvement of the SSA from Alg. 3.1 and is closely related to the QSSA. In short, the ssSSA, given a system defined as in Def. 3.10, introduces a new *virtual fast process* $\hat{X}^f(t)$, which is composed of the same species of $X^f(t)$, but evolves solely from the fast reactions (i.e. with all the slow reactions turned off). Since the dynamics of $X^f(t)$ depends from both $X^f(t)$ and $X^s(t)$, we cannot represent $X^f(t)$ as a Markovian process, but sure we can for $\hat{X}^f(t)$, which means that its time evolution is governed by the master equation

$$\frac{\partial \hat{P}(x^f, t \mid x_0, t_0)}{\partial t} = \sum_{j=1}^{M_f} a_j^f(x^f - \nu_j^f, x_0^s)\hat{P}(x^f - \nu_j^f, t|x_0, t_0) - a_j^f(x^f, x_0^s)\hat{P}(x^f, t|x_0, t_0), \tag{A.1}$$

in which $\hat{P}(x^f, t|x_0, t_0) = Pr\left[\hat{X}^f(t) = x^f | X(t_0) = x_0\right]$. Moreover, being the stiffness defined as the multiple-time scale dynamics of the reacting system comprising both fast and slow reactions, in which the fast reactions are stable, the process $\hat{X}^f(t)$ has to meet the following requirements, called the *stiffness conditions*:

Condition A.1: In the stable process $\hat{X}^f(t)$

$$\exists\, \hat{P}(x^f, \infty|x_0) : \lim_{t\to\infty} \hat{P}(x^f, t|x_0, t_0) \equiv \hat{P}(x^f, \infty|x_0),$$

where $\hat{P}(x^f, \infty|x_0)$ is a time independent probability function of the process $\hat{X}^f(\infty)$ when it equals the state x^f, given the initial state x_0.

Condition A.2: The relaxation time $\hat{t}^f_{relax}$ of $\hat{X}^f(t)$ to its asymptotic form $\hat{X}^f(\infty)$ should be small in comparison to the average expected time $\bar{t}_s$ in which a slow reaction will occur, i.e.

$$\hat{t}^f_{relax} \ll \bar{t}_s.$$

The integration of the two time scale processes $X^f(t)$ and $X^s(t)$ rises the question of how propensity functions behave in a scenario, where $X^f(t)$ quickly converge to its asymptotic state in a time frame that is short compared to the time $\bar{t}_s$, in which a slow reaction will occur. In other words, we would like to calculate the propensity functions, if the process $X^f(t)$ is replaced with $\hat{X}^f(t)$. Authors in [94] proposed, that if the conditions Cond. A.1 and Cond. A.2 are met, then it is possible to define the *slow-scale approximation* [94]:

Definition A.1: Let be a system from 3.10 in the state (x^f, x^s) at a time t, and let denote with $\hat{t}^f_{relax}$ the relaxation time of the stable fast process $\hat{X}^f(t)$ and with $\bar{t}_s$ the average expected time to the next slow reaction. It is possible then to prove, that it exists such time increment Δ_s, for which it holds

$$\hat{t}^f_{relax} \ll \Delta_s \ll \bar{t}_s, \tag{A.2}$$

i.e. Δ_s is very large compared to $\hat{t}^f_{relax}$ and very small compared to $\bar{t}_s$, whereas the probability that one reaction R^s_j will occur in the next time interval $[t, t+\Delta_s)$ can be approximated with $\overline{a^s_j}(x^s; x^f)\Delta_s$, where

$$\overline{a^s_j}(x^s; x^f) = \sum_{x^{f'}} \hat{P}(x^{f'}, \infty | x^f, x^s)\, a^s_j(x^{f'}, x^s), \tag{A.3}$$

in which $\hat{P}(x^{f'}, \infty | x^f, x^s)$ is the probability that $\hat{X}^f(\infty) = x^{f'}$, given that $X(t) = (x^f, x^s)$.

The Eq. (A.3) is often referred as the *slow-scale propensity function* for the reaction channel R^s_j. Eq. (A.3) calculates the j-th slow propensity function by simply averaging all

the j-th slow propensities over all fast variables in $\hat{X}^f(t)$. The Def. A.1 is the basis for the ssSSA:

Algorithm A.1

The slow-scale stochastic simulation algorithm (ssSSA).

Input: ▪ a model from the Def. 3.10.

Output: ▪ the model responses $x = x^s$ over time t.

procedure ssSSA

1. Identify the virtual fast system $\hat{X}^f(t)$ and compute its stationary probability function $\hat{P}(x^{f'}, \infty | x^f, x^s)$. Initialize the initial state $t = t_0$, $x^f = x_0^f$ and $x^s = x_0^s$.
2. Determine all the propensities values $\overline{a_j^s}(x^s, x^f)$ according to Eq. (A.3).
3. Compute $\overline{a_0^s}(x^s; x^f) = \sum_{j-1}^{M_s} a_j^s(x^s; x^f)$.
4. Monte Carlo step: sample two random numbers r_1 and r_2 from the interval $(0, 1]$ and compute τ and the index j as:

$$\tau = \frac{1}{\overline{a_0^s}(x^s; x^f)} \ln\left(\frac{1}{r_1}\right),$$

$$\min_j \left\{ \sum_{k=1}^{j} \overline{a_k^s}(x^s; x^f) \geq r_2 \overline{a_0^s}(x^s; x^f) \right\}.$$

5. Increase the time step $t \leftarrow t + \tau$ and update the system states:

$$x_i^s \leftarrow x_i^s + \nu_{ij}^{ss}, \quad (i = 1, \dots N_s), \tag{A.4}$$

$$x_i^f \leftarrow x_i^f + \nu_{ij}^{fs}, \quad (i = 1, \dots N_f), \tag{A.5}$$

$$x^f \leftarrow \text{sample from } \hat{P}(x^{f'}, \infty | x^f, x^s). \tag{A.6}$$

6. Record the current pair $X(t) = (x^f, x^s)$ with t and go back to step 2 if $t < t_{max}$.

end procedure

It is clearly visible, that the Alg. A.1 is an extension of Alg. 3.1.

The main problem of Alg. A.1 is that the probability function $\hat{P}(x^f, \infty | x_0)$ is difficult to compute. One way to obtain an exact calculation is to estimate the asymptotic

behaviour of $\hat{P}(x^f, t|x_0, t_0)$, which equals to set

$$\frac{\partial \hat{P}(x^f, t \mid x_0, t_0)}{\partial t} = 0, \tag{A.7}$$

from Eq. (A.1), as we get

$$0 = \sum_{j=1}^{M_f} a_j^f(x^f - \nu_j^f, x_0^s)\hat{P}(x^f - \nu_j^f, \infty|x_0) - a_j^f(x^f, x_0^s)\hat{P}(x^f, \infty|x_0), \tag{A.8}$$

which gives a set of algebraic equations, that can be solved much easier than the set of differential equations generated by Eq. (A.1). If the solution of Eq. (A.8) cannot be easily computed because of the size of the system, then an approximation is required. The probability function $\hat{P}(x^f, \infty|x_0)$ is fundamental for the Alg. A.1, because it is needed to both calculate the propensities and to update the fast system in Eq. (A.6). Authors in [94] show that, it is possible to perform all these operations by knowing only the first moments of $\hat{P}(x^f, \infty|x_0)$.

Authors in [94] also show, that the calculation of the slow-scale propensities $\overline{a_j^s}(x^s; x^f)$ in step 2 can be done straightforwardly. For instance, consider a slow reaction in which only slow species occur. In such scenario $a_j^s(x)$ is obviously not dependent from x^f and hence $\overline{a_j^s}(x^s; x^f) = a_j^s(x^s)$. The remaining four reactions that can occur are the following:

$$S_i^f \xrightarrow{c_j^s} \dots \qquad a_j^s(x) = c_j^s x_i^f, \qquad \overline{a_j^s}(x^s; x^f) = c_j^s \langle \hat{X}_i^f(\infty) \rangle, \tag{A.9a}$$

$$S_i^f + S_{i'}^s \xrightarrow{c_j^s} \dots \qquad a_j^s(x) = c_j^s x_i^f x_{i'}^s, \qquad \overline{a_j^s}(x^s; x^f) = c_j^s x_{i'}^s \langle \hat{X}_i^f(\infty) \rangle \tag{A.9b}$$

$$S_i^f + S_i^f \xrightarrow{c_j^s} \dots \qquad a_j^s(x) = c_j^s x_i^f (x_i^f - 1) \qquad \overline{a_j^s}(x^s; x^f) = \frac{1}{2} c_j^s \langle \hat{X}_i^f(\infty) \left(\hat{X}_i^f(\infty) - 1 \right) \rangle \tag{A.9c}$$

$$S_i^f + S_{i'}^f \xrightarrow{c_j^s} \dots \qquad a_j^s(x) = c_j^s x_i^f x_{i'}^f \qquad \overline{a_j^s}(x^s; x^f) = c_j^s \langle \hat{X}_i^f(\infty) \hat{X}_{i'}^f(\infty) \rangle, \; i \neq i', \tag{A.9d}$$

where the average notation $\langle\rangle$ (see [94]) is defined as

$$\langle f(\hat{X}_i^f(\infty)) \rangle = \sum_{x^{f'}} \hat{P}(x^{f'}, \infty|x^f, x^s) f(x^{f'}).$$

We refer to the original paper for a more exhaustive description of the steps that lead to the Alg. A.1, together with few examples illustrating the steps involving the identification of $\hat{X}^f(t)$ and the estimation of the first moments of $\hat{P}(x^f, \infty|x_0)$.

It is important to remember, that if the system does not met either A.1 or A.2, then it means that the fast reactions affect the system in a way, that it cannot be approximated as a simple stationary Markov process. In this case the slow-scale approximation should rather be replaced by the common SSA, as authors in [94] strongly emphasize.

A.1.2 Multi-scale stochastic simulation algorithm

The MSSA has been developed along the ssSSA and it appears first in [95]. MSSA has a lot in common with the ssSSA, starting with the same definition of the stiff chemically reacting system given in Def. 3.10. The MSSA assumes that the *stochastic partial equilibrium* is quickly reached in the virtual fast process $\hat{X}^f(t)$. Such equilibrium is told to be partial because it is confined in the time scale of $\hat{X}^f(t)$. In this assumption, two aspects are considered:

1. $\hat{X}^f(t)$ eventually reaches the final equilibrium state $\hat{X}^f(\infty)$, with a distribution, that is not affected by the occurrence of fast reactions, rather than the slow reactions[1].

2. The transient or relaxing time τ_{relax} (or relaxation period), in which the process $\hat{X}^f(t)$ converges to $\hat{X}^f(\infty)$, is negligible, if compared to the time in which a slow reaction should occur. When the partial equilibrium is reached, then will hold

$$\hat{X}^f(t) \;\approx\; \hat{X}^f(t + \tau_{relax}) \;\approx\; \hat{X}^f(\infty), \tag{A.10}$$

   where both $\hat{X}^f(t)$ and $\hat{X}^f(t + \tau_{relax})$ can be thought as independent random variables.

These two aspects are in a way equivalent to the ssSSA conditions from Cond. A.1 and Cond. A.2. MSSA is conceptually similar to the ssSSA. It begins by defining the following:

[1] This statement can be easily proved by plotting the histogram of a simple process $X(t)$, from multiple runs of the ordinary SSA, at different values of t and then compare it to the histogram of the system at the equilibrium, i.e. $\hat{X}(\infty)$.

- $P_0^s(\tau|x^f, x^s, t)$, which is the probability that no slow reaction will occur in the time interval $[t, t+\tau]$, if $X(t) = (x^f, x^s)$, and
- $a_0^s(X) = a_0^s(x^f, x^s) = \sum_{j=1}^{M} a_j^s(x^f, x^s)$.

MSSA chooses such time τ so that no slow reactions will occur. This allows the fast process $X^f(t)$ to be replaced by the virtual $\hat{X}^f(t)$. If at this time the process $X^f(t)$ is at a specific state $x^{f'}$, i.e. $X^f(\tau) = x^{f'}$ then we can define the probability of at least one reaction occurring in the next infinitesimal time interval $[\tau, \tau + \mathrm{d}\tau]$ as $a_0^s(x^{f'}, x^s)\mathrm{d}\tau$. Its total probability can be defined as

$$E(a_0^s(X^f(\tau), x^s)|x^f, x^s)\mathrm{d}\tau = \sum_{x^{f'}} P(X^f(\tau) = x^{f'}|x^f, x^s) \cdot a_0^s(x^{f'}, x^s)\mathrm{d}\tau, \quad \text{(A.11)}$$

where $E(a_0^s(X^f, x^s)|x^f, x^s)$ denotes the conditional mean of the slow propensity functions over all the fast states $x^{f'}$ (or equivalently over all the fast reactions), if $X(t) = (x^f, x^s)$. Here, the notation of the total probability with the mean $E(a_0^s(X^f, x^s)|x^f, x^s)$ is intentional. $P(X^f(\tau) = x^{f'}|x^f, x^s)$ instead denotes the conditional probability of the fast process $X^f(t)$ being in the state $x^{f'}$, as well if $X(t) = (x^f, x^s)$.

Similarly as in the case of the original SSA, it is possible to define the probability $P_0^s(\tau|x^f, x^s, t)$ with the aim of an infinitesimal equation. Consider the scenario in which in the next infinitesimal time interval $[\tau, \tau + \mathrm{d}\tau]$ at most one slow reaction can occur. This means that the total probability that no slow reaction will occur, in the time frame $[t+\tau, t+\tau+\mathrm{d}\tau]$, i.e. $P_0^s(\tau + \mathrm{d}\tau|x^f, x^s, t)$, is given by the product of the probability that no slow reaction has occurred till the beginning of this time interval, and the probability that no slow reaction will occur in the time frame $[\tau, \tau + \mathrm{d}\tau]$:

$$P_0^s(\tau + \mathrm{d}\tau|x^f, x^s, t) = P_0^s(\tau|x^f, x^s, t) \cdot \left(1 - E(a_0^s(X^f(\tau), x^s)|x^f, x^s)\mathrm{d}\tau\right). \quad \text{(A.12)}$$

Eq. (A.12) can be solved similarly as in Eq. (3.40) as follows:

$$\begin{aligned}
\frac{P_0^s(\tau + \mathrm{d}\tau|x^f, x^s, t) - P_0^s(\tau|x^f, x^s, t)}{\mathrm{d}t} &= -E(a_0^s(X^f(\tau), x^s)|x^f, x^s)\, P_0^s(\tau|x^f, x^s, t) \\
\frac{\dot{P}_0^s(\tau|x^f, x^s, t)}{P_0^s(\tau|x^f, x^s, t)} &= -E(a_0^s(X^f(\tau), x^s)|x^f, x^s) \\
\ln\left(P_0^s(\tau|x^f, x^s, t)\right) &= -\int_t^{t+\tau} E(a_0^s(X^f(\mu), x^s)|x^f, x^s)\mathrm{d}\mu \\
P_0^s(\tau|x^f, x^s, t) &= \mathrm{e}^{-\int_t^{t+\tau} E(a_0^s(X^f(\mu), x^s)|x^f, x^s)\mathrm{d}\mu}.
\end{aligned} \tag{A.13}$$

By defining the probability density function $p(\tau, j|x^f, x^s, t)$, the expression

$$p(\tau, j|x^f, x^s, t)\mathrm{d}\tau$$

denotes the probability that a slow reaction R_j^s will occur in the next infinitesimal time interval $[t + \tau, t + \tau + \mathrm{d}\tau]$, given $X(t) = (x^f, x^s)$. Hence it is possible to write

$$p(\tau, j|x^f, x^s, t) = E(a_j^s(X^f(\tau), x^s)|x^f, x^s) \cdot P_0^s(\tau|x^f, x^s, t). \tag{A.14}$$

If the stochastic partial equilibrium assumption is applied to Eq. (A.14), i.e. $\tau << \tau_{relax}$ and $X^f(\tau) \approx \hat{X}^f(\infty)$, Eq. (A.14) leads to

$$p(\tau, j|x^f, x^s, t) = E(a_j^s(\hat{X}^f(\infty), x^s)|x^f, x^s) \cdot \mathrm{e}^{-\tau E(a_0^s(\hat{X}^f(\infty), x^s)|x^f, x^s)}. \tag{A.15}$$

The MSSA is based on Eq. (A.15), by applying the stochastic partial equilibrium to the fast process $\hat{X}^f(t)$. The notation above could be simplified as

$$\begin{aligned}
\overline{a}_j^s(x^s) &= E(a_j^s(\hat{X}^f(\infty), x^s)|x^f, x^s), \\
\overline{a}_0^s(x^s) &= \sum_{j=1}^{M} \overline{a}_j^s(x^s), \\
P_0(\tau|x^f, x^s, t) &= \mathrm{e}^{-\overline{a}_0^s(x^s)\tau}
\end{aligned} \tag{A.16}$$

The algorithm proposed by Cao et al. in [95] is depicted in Alg. A.2.

Algorithm A.2

The multi-scale stochastic simulation algorithm (MSSA) with the stochastic partial equilibrium assumption.

Input: ▪ a model from the Def. 3.10.

Output: ▪ the model responses $x = x^s$ over time t.

procedure MSSA

1. Apply the stochastic partial equilibrium assumption to $X^f(t)$ and compute $\hat{X}^f(\infty)$.
2. Determine all the propensities values $\overline{a_j^s}(x^s)$ and $\overline{a_0^s}(x^s)$ according to Eq. (A.16), for all $j = 1, \ldots, M_s$.
3. Monte Carlo step: sample two random numbers r_1 and r_2 from the interval $(0, 1]$ and compute τ and the index j of the next slow reaction to fire, as

$$\tau = \frac{1}{\overline{a_0^s}(x^s)} \ln\left(\frac{1}{r_1}\right),$$

$$\min_j \left\{ \sum_{k=1}^{j} \overline{a_k^s}(x^s) \geq r_2 \overline{a_0^s}(x^s) \right\}.$$

4. Record $X(t)$ and increase the time step $t \leftarrow t + \tau$. Update the system states $x = x + \nu_j$ and go back to step 1 if $t < t_{max}$.

end procedure

The most difficult task of Alg. A.2 is step 1. In order to compute the stochastic partial equilibrium of the virtual fast process $\hat{X}^f$, one has to isolate the fast reactions first and solve the equilibrium and conservation law equations for the fast reacting system. We leave the technicalities of derivation of these equations, as well as examples, to [95]. On the other hand the propensity functions can be calculated in the same way as in Eq. (A.9).

MSSA can fill the gap of hybrid methods, when the species population are small in both the reactions time scales, as thoroughly argumented in [95]. As well as the ssSSA, also MSSA notably improve the overall performance of the stochastic simulation for stiff chemically reacting systems.

A.1.3 Nested stochastic simulation algorithm

The third computational approach that deserved to be shortly described is the nSSA [192]. The nSSA has been developed precisely to resemble the multi-scale dynamics of chemically reacting systems by simulating, in each separating time scale, the reactions of the system, without any *a priori* assumptions about the analytic form of the rate functions.

The nSSA proposed by E et al. in [192] differs substantially from the two previous described algorithms, i.e. the ssSSA and MSSA proposed by Cao et al. , since the input model of nSSA does not require to differentiate the species as the slow and fast variables, but only the reactions, i.e. $R^s = (a^s, \nu^s)$ and $R^f = (a^f, \nu^f)$. The two groups of reactions are distinguished solely from the propensity functions $a_j(x)$ having different magnitudes, i.e.

$$\begin{aligned} a^s(x) &= \left(a_1^s(x), \dots, a_{M_s}^s\right) = \mathcal{O}(1), \\ a^f(x) &= \left(a_1^f(x), \dots, a_{M_f}^f\right) = \mathcal{O}(1/\epsilon), \end{aligned} \tag{A.17}$$

where $\epsilon \ll 1$ and the $\mathcal{O}$ notation denotes the size in magnitudes in dimensionless unit.

The nSSA, instead of approximate the process $X(t)$ with the virtual system $\hat{X}^f(t)$ as in ssSSA and MSSA, which basically allows to omit entirely the simulation of the fast reactions, it performs a normal SSA for the slow reactions, while running an inner SSA for a short time T_f to compute the fast variables for each step of the outer SSA. The inner SSA performs the fast reactions $R^f(a^f, \nu^f)$ only and it is ran in N-independent[2] replicas. These replicas are then used to average the slow propensities $\tilde{a}_j^s(x)$ as

$$\tilde{a}_j^s = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{T_f} \int_{T_0}^{T_0+T_f} a_j^s(x_k^f(\tau)) \mathrm{d}\tau, \tag{A.18}$$

in which x_k is the k-th auxiliary fast process, identical to the virtual fast process $\hat{X}^f$, as defined in the ssSSA and MSSA. The full algorithm is depicted in Alg. A.3.

Algorithm A.3

The nested stochastic simulation algorithm (nSSA).

[2] Here N does not denote the same N as the number of different chemical species in Def. 3.9.

Input: ▪ a model from the Def. 3.10.

Output: ▪ the model responses $x = x^s$ over time t.

procedure nSSA

1. Identify the fast and the slow propensity functions from $R^s = (a^s, \nu^s)$ and $R^f = (a^f, \nu^f)$ as in Eq. (A.17). Identify the auxiliary fast process x^f. And set the current state to the initial state x_0.
2. Create N different "inner" SSA instances of the fast process x^f from the current state.
3. Run the N-replicas of the "Inner" SSA for the fast process.
4. Determine all the propensities values $\tilde{a}_j^s$ according to Eq. (A.18), for all $j = 1, \dots, M_s$.
5. "Outer" SSA: run one step of the SSA for the slow reactions with the modified slow propensities $\tilde{a}_j^s$.
6. Record the state x and increase the time step $t \leftarrow t + \tau$. Update the system states $x = x + \nu_j$ and go back to step 2 if $t < t_{max}$.

end procedure

The main problem of the Alg. A.3 is the step 4. To determine the temporal averages, one must select such time T_f and such repetition number N in order to provide meaningful approximation of the quasi-equilibrium that needs to be reached in the fast auxiliary process x^f. Authors in [192] show that the propensities $\tilde{a}_j^s$, estimated from Eq. (A.18), can very well converge to the average rates, with respect to the quasi-equilibrium denote by $\mu_y(x)$ and computed as

$$\bar{a}_y = \sum_{x \in X} a_j(x) \mu_y(x)$$

where y denotes the slow observable (variable) and X is the space of all possible states x, if T_f and N are chosen based on the relation

$$N = \frac{T_f}{\epsilon} = \frac{1}{\lambda^2}, \tag{A.19}$$

for which λ denotes the error tolerance. We refer to the original paper [192] as well as in the arguments summarized in [238, 239] for more details regarding the nSSA

formulation, as well as the full estimation of the time complexity of the Alg. A.3.

The nSSA has many common foundations with the ssSSA, primarily sharing the two stiffness conditions. However, the similarity regarding the generality of the nSSA approach, has been a topic of debate in a correspondence between authors of [94, 95] and authors of [192], occurred as a sequence of comments over the same papers (see [238] and [239] for further details).

A.2 Additional notes

A.2.1 Alternative parameter sensitivities definitions

Here we give a more detail presentation of the alternative response value measures used in the Morris elementary effects for computing the parameter sensitivities that we introduced in section 6.3.2. More specifically, we present here an alternative measure, based on the Fourier coefficients of the Fourier series, which is used to approximate the output signal of the model, in the scenario where it shows an oscillatory behaviour.

A.2.2 The Fourier series approximation

With a Fourier series function approximation one can provide a time independent value for the response function $y[n]$ used to compute the elementary effects. One can use the estimates of the first components of the *discrete Fourier series* (DFS) approximation of $y(\mathbf{x})$, as suggested in [214]. Such methodology can be also integrated with other structural properties of the nonlinear or oscillatory response function, such as the main frequency, the local extrema [240] and the phase [241].

The Fourier series of $y(\mathbf{x})$ can be computed as

$$y(t) \approx \mathscr{F}_N(t) = \frac{a_0}{2} + \sum_{n=1}^{N} \left(a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)\right), \tag{A.20}$$

where N is the number of desired harmonics (a maximum of 8 are allowed in MATLAB®), $\omega_0 = \frac{2\pi}{T_0}$, T_0 is the base period of the function y, and where the coefficients a_n and b_n

are defined as

$$a_n = \frac{2}{T_0} \int_{t_0}^{t_0+T_0} y(t) \cos(n\omega_0 t)\, \mathrm{d}t$$
$$b_n = \frac{2}{T_0} \int_{t_0}^{t_0+T_0} y(t) \sin(n\omega_0 t)\, \mathrm{d}t \tag{A.21}$$

In a scenario where the function y is a discrete time signal $y[n]$, the Fourier series from Eq. (A.20) can be rewritten as

$$y[n] \approx x[n] = \sum_{k=0}^{N-1} a_k \mathrm{e}^{ik\omega_0 n}, \tag{A.22}$$

in which the coefficients a_k are defined as

$$a_k = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \mathrm{e}^{-ik\omega_0 n}. \tag{A.23}$$

where i is the *imaginary unit* $\sqrt{-1}$. By using a model response representation as per Eq. (A.20) or Eq. (A.22) it is possible to apply the Morris method even for non-monotonic responses, as we will show in the next sections.

A.2.3 The average and the median of Fourier coefficients

The average of the principal frequency $f_y(\mathbf{x})$ (and so the average amplitude $a_y(\mathbf{x})$ and the average period $p_y(\mathbf{x})$) of the response y provides a robust scalar value for assessing the elementary effects in a scenario where the function y behaves in an oscillatory manner. However, there might be some occasions in which the function y may not necessarily exhibit a full period of oscillation during the entire simulation time frame. In such scenario, the average frequency approach may fail to describe correctly the nature of the function y. In order to overcome this shortcoming we propose instead to perform a Fourier series approximation of the function y, and refer to the coefficients a_n and b_n as possible alternatives for the scalar value representing the response function y.

Let's considered to be $\mathscr{F}_{N_s}(y)$ the Fourier series approximation of the output re-

sponse $y[n]$, i.e.

$$y[n] \approx \mathscr{F}_{N_s}(y) = \frac{a_0}{2} + \sum_{l=1}^{N_s} (a_n \cos(l\omega n) + b_n \sin(l\omega n)), \tag{A.24}$$

similarly as in Eq. (A.20). By applying Eq. (A.24) over the matrix $\mathbf{Y}$ we obtain

$$\mathscr{F}_{N_s}(\mathbf{Y}) = \begin{bmatrix} \mathscr{F}_{N_s}(y_1) \\ \mathscr{F}_{N_s}(y_2) \\ \vdots \\ \mathscr{F}_{N_s}(y_M) \end{bmatrix}. \tag{A.25}$$

By computing the matrix $\mathscr{F}_{N_s}$, we can derive a matrix containing all the Fourier coefficients a_n, b_n and ω of all the M responses

$$\mathbf{C}_{\mathscr{F}(\mathbf{Y})} = \mathbf{C}_{\mathscr{F}} = \begin{bmatrix} a_{0,1} & a_{1,1} & \dots & a_{N_s,1} & b_{1,1} & \dots & b_{N_s,1} & \omega_1 \\ a_{0,2} & a_{1,2} & \dots & a_{N_s,2} & b_{1,2} & \dots & b_{N_s,2} & \omega_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{0,M} & a_{1,M} & \dots & a_{N_s,M} & b_{1,M} & \dots & b_{N_s,M} & \omega_M \end{bmatrix}. \tag{A.26}$$

The average values of $\mathbf{C}_{\mathscr{F}}$ are denoted by

$$\overline{\mathbf{C}}_{\mathscr{F}} = \left[\bar{a}_0, \bar{a}_1, \dots \bar{a}_{N_s}, \bar{b}_1, \dots \bar{b}_{N_s}, \overline{\omega}\right], \tag{A.27}$$

while the median value of $\mathbf{C}_{\mathscr{F}}$ are

$$\hat{\mathbf{C}}_{\mathscr{F}} = \left[\hat{a}_0, \hat{a}_1, \dots \hat{a}_{N_s}, \hat{b}_1, \dots \hat{b}_{N_s}, \hat{\omega}\right], \tag{A.28}$$

where

$$\bar{a}_j = \frac{1}{M} \sum_{k=1}^{M} a_{j,k}, \quad j = 0..N_s, \tag{A.29}$$

$$\bar{b}_j = \frac{1}{M}\sum_{k=1}^{M} b_{j,k}, \quad j = 1..N_s, \tag{A.30}$$

$$\overline{\omega} = \frac{1}{M}\sum_{k=1}^{M} \omega_k, \tag{A.31}$$

and

$$\hat{c}_j = \begin{cases} c_{j,k} \ \text{ where } k = \left\lfloor \frac{M-1}{2} \right\rfloor, & \text{if } M \text{ is odd} \\ \frac{1}{2}\left(c_{j,k'} + c_{j,k''}\right) \ \text{ where } k' = \left\lfloor \frac{M}{2} \right\rfloor,\ k'' = \left\lfloor \frac{M}{2} + 1 \right\rfloor, & \text{if } M \text{ is even} \end{cases} \tag{A.32}$$

where c_j is one of the following value: a_j, b_j or ω. We can use any of the values in $\overline{\mathbf{C}}_{\mathscr{F}}$ or in $\hat{\mathbf{C}}_{\mathscr{F}}$ as an alternative value to the average of the principal frequencies $f_y(\mathbf{x})$ for computing the elementary effects as per Eq. (5.6). Therefore, by computing the elementary effects

$$d_i^{\overline{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}))} = \frac{\overline{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}+\Delta_i)) - \overline{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}))}{\Delta} \tag{A.33}$$

and

$$d_i^{\hat{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}))} = \frac{\hat{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}+\Delta_i)) - \hat{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}))}{\Delta}, \tag{A.34}$$

we can obtain the parameter sensitivities for the i-th input of the model, for each of the Fourier series coefficient, as

$$\begin{aligned} \mu_i^*\left(\overline{\mathbf{C}}_{\mathscr{F}}\right) &= \frac{1}{r}\sum_{l=1}^{r}\left| d_i^{\overline{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}_l))}\right|, \\ \sigma_i\left(\overline{\mathbf{C}}_{\mathscr{F}}\right) &= \sqrt{\frac{\sum_{l=1}^{r}\left(d_i^{\overline{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}_l))} - \mu_i\right)^2}{r}}, \end{aligned} \tag{A.35}$$

where

$$\mu_i = \frac{1}{r}\sum_{l=1}^{r} d_i^{\overline{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}_l))}.$$

The same statistics can be obtained with the median values $\hat{\mathbf{C}}_{\mathscr{F}}$ as

$$\begin{aligned}\mu_i^*\left(\hat{\mathbf{C}}_{\mathscr{F}}\right) &= \frac{1}{r}\sum_{l=1}^{r}\left|d_i^{\hat{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}_l))}\right|,\\ \sigma_i\left(\hat{\mathbf{C}}_{\mathscr{F}}\right) &= \sqrt{\frac{\sum_{l=1}^{r}\left(d_i^{\hat{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}_l))} - \mu_i\right)^2}{r}},\end{aligned} \tag{A.36}$$

where

$$\mu_i = \frac{1}{r}\sum_{l=1}^{r} d_i^{\hat{\mathbf{C}}_{\mathscr{F}}(\mathbf{Y}(\mathbf{x}_l))}.$$

These statistics are vectors of the form:

$$\begin{aligned}\mu_i^*\left(\overline{\mathbf{C}}_{\mathscr{F}(\mathbf{Y})}\right) &= \left[\mu_i^*(\overline{a}_0), \mu_i^*(\overline{a}_1), \dots, \mu_i^*\left(\overline{a}_{N_s}\right), \mu_i^*\left(\overline{b}_1\right), \dots, \mu_i^*\left(\overline{b}_{N_s}\right), \mu_i^*(\overline{\omega})\right],\\ \sigma_i\left(\overline{\mathbf{C}}_{\mathscr{F}(\mathbf{Y})}\right) &= \left[\sigma_i(\overline{a}_0), \sigma_i(\overline{a}_1), \dots, \sigma_i\left(\overline{a}_{N_s}\right), \sigma_i\left(\overline{b}_1\right), \dots, \sigma_i\left(\overline{b}_{N_s}\right), \sigma_i(\overline{\omega})\right],\\ \mu_i^*\left(\hat{\mathbf{C}}_{\mathscr{F}(\mathbf{Y})}\right) &= \left[\mu_i^*(\hat{a}_0), \mu_i^*(\hat{a}_1), \dots, \mu_i^*\left(\hat{a}_{N_s}\right), \mu_i^*\left(\hat{b}_1\right), \dots, \mu_i^*\left(\hat{b}_{N_s}\right), \mu_i^*(\hat{\omega})\right],\\ \sigma_i\left(\hat{\mathbf{C}}_{\mathscr{F}(\mathbf{Y})}\right) &= \left[\sigma_i(\hat{a}_0), \sigma_i(\hat{a}_1), \dots, \sigma_i\left(\hat{a}_{N_s}\right), \sigma_i\left(\hat{b}_1\right), \dots, \sigma_i\left(\hat{b}_{N_s}\right), \sigma_i(\hat{\omega})\right].\end{aligned}$$

Being a_0 the leading coefficient of the Fourier series, then an obvious choice for parameter sensitivities, would be the statistics:

$$\mu_i^*(\overline{a}_0), \sigma_i(\overline{a}_0) \quad \text{and} \quad \mu_i^*(\hat{a}_0), \sigma_i(\hat{a}_0). \tag{A.37}$$

The main benefit of such approach is that we can efficiently deal also with model output responses that exhibits only a partial oscillatory behaviour. In such scenarios the main component of the frequency spectrum, will not be a reliable representative scalar value of the response function. The Fourier series approximation however, will be

able to fit accurately the discrete-time points of the output response function. Having available both the mean and median statistics, as representative scalars of the model response function, allows to avoid the possible bias that can be generated from the computation of the means in Eq. (A.27). It can occur a scenario where a Fourier series coefficient differs from the others $M-1$ coefficients in the same column, by several orders of magnitude. Such large deviation is known as a skewed value. The mean of these coefficients would be therefore heavily biased by this value. The median value instead helps us to avoid the influence of such skewed value by providing instead another value that represents more solidly the most typical occurring coefficient in the Fourier series approximations of the matrix $\mathbf{Y}$. Furthermore, the same Fourier series approximations that we derived for the matrix $\mathbf{Y}$, can be also applied for the binding gradients responses defined within the matrix $\mathbf{G}$ from Eq. (6.7).

A.3 ParMSSA: a simulator engine for parallel multi-scale stochastic simulations and sensitivity analysis

The sensitivity analysis based on the Morris experiment usually requires several evaluations of the model response for different sets of input parameters. When a deterministic simulations are applied to obtain the response of the model [196], the computational expensiveness of the Morris sensitivity analysis is bounded by:

- the size of the parameters space, that has to be sampled in order to evaluate the elementary effects, and by
- the expensiveness of the model simulation.

Stochastic analysis requires to perform multiple simulation runs of the same configurations, in order to construct a probability distribution of the chemical master equation (CME). It is vital to ensure that the stochastic algorithm returns the model's response quickly, so:

- one can perform multiple simulations runs faster, making the results statistically significant,
- the Morris method can perform the sensitivity analysis efficiently and in a reasonable amount of time.

Since both the SSA and the DMSSA are notoriously hard to parallelise, we explore the possibility to perform multiple independent simulations at once, by exploiting *coarse-grained* parallelism in modern computer hardware[3].

A.3.1 Accelerating stochastic simulations

The use of parallel programming for performing large number of stochastic simulations in systems biology is not new and several successful attempts can be found in the literature [242–247]. In fact, parallelisation has become a standard practice as a way to speed-up the simulations in systems biology. In the last decade three different computer platforms were exploited for this task: the *computer cluster*, the *multi-core CPU*, and the many-core *general purpose graphic processing unit* (GPGPU).

For each of them, several *application programming interfaces* (APIs) were developed, mostly for the C and C++ languages, in order to make the integration with already existing code easier. These APIs usually offer a high level of abstraction, so the developers do not need to know the basic machine operations that lie behind the API. However in some parallel platforms, a certain user control is often required, such as in the case of the computer cluster, where the communication between the machines has to be properly configured. This and other problems have lead to the development of protocols for data communication in distributed memory systems.

Parallelisation for computer clusters

In a computer cluster, *Message Passing Interface* (MPI) is commonly used to exploit the processing power of multiple computer systems at the same time [248, 249]. The MPI is a *de-facto* standard used for communication between parallel jobs running on a distributed machine or cluster. It is based on the message passing technique for asynchronous data communication. The C/C++ implementation of this standard provides an API which enables in-program communication between multiple simulation instances scattered throughout the cluster. With MPI one can target each computer in the cluster to perform one or more parallel instances of the task. With the MPI, multiple simulations can be performed on a distributed system, which may contain

[3] However we may accelerate the DMSSA by accelerating the SSA core by using faster methods than DM. For instance, significant improvements can be achieved by using the logarithmic direct method (LDM) or the tau-leaping method instead.

from a dozen to hundred of machines [245–247]. We refer to [249] for a more detail overview and examples of the MPI programming.

Parallelisation for multi-core systems

In the multi-core system one can exploit the ability to simultaneously perform as many simulations as is the number of cores that are available in the CPU. The API that is suited for this task is called *Open Multi-Processing* (OpenMP), which is today one of the most largely used API for exploiting parallel computation in multi-core systems. OpenMP provides an efficient way of utilization of multiple cores of modern processors. More specifically it helps to parallelise loops and blocks of code which may share specific memory contents. More details about OpenMP specifications and programming can be found in [250, 251]. In systems biology one may find OpenMP useful in the context of accelerating those parts of the simulation that can be parallelised [252, 253]. Tian et al. show that the OpenMP can be exploited to perform multiple SSA steps at once and hence reduce the computation expensiveness of running multiple stochastic simulations [254].

Nowadays it is common to combine both OpenMP and MPI together in a hybrid fashion [255, 256]. This provides a powerful technique to improve the computation on a cluster, since each run can further exploit multi-core architecture via OpenMP. Such optimizations can be indeed applied to improve performances in stochastic simulations [246] as well as in a variety of systems biology models [257, 258].

Parallelisation for GPGPU and many-core accelerators

The third computer system that can be exploited for parallelisation tasks is the *general purpose graphic processing unit* (GPGPU). A modern GPGPU is usually composed of several multi-processors. Each one contains tens (or hundreds) of small processing units, optimized for *single instruction multiple data* (SIMD) processing (see the schematic examples in [259]). The first examples of this architecture were developed in the early 2000', mainly for the need to accelerate the graphical pipeline used for rendering three dimensional (3D) objects in the context of computer graphics.

By using specific APIs, this computing resource can be used for general purpose computing. The two most widely used APIs for GPGPU programming are: the Nvidia's[4] *Compute Unified Device Architecture* (CUDA™) [260] and *Open Computing Language*

[4] Nvidia© is one of the major GPU manufacturer.

(OpenCL™) [261]. Both APIs contain programming libraries that help developers to build parallel applications capable to exploit both *fine* and *coarse-grain* parallelisation on GPGPUs. Parallel applications built with CUDA or OpenCL can make use of several hundred of parallel threads that can perform an identical task, usually called *kernels*, over a great amount of data (as in the SIMD processing). Therefore, algorithms suited for a SIMD architecture can benefit from a GPGPU implementation. Even though performance benchmarks commonly reward CUDA as a slightly faster API [262], both APIs are regarded as equivalent standards for parallel programming. However programming with GPGPU usually requires deep knowledge of the hardware specifics, especially the memory hierarchy. In fact, one of the main things that the developers must be aware of, is that they should minimize the memory access when using the data stored in the device's main memory[5]. A single access to the main memory usually results in big latencies. This creates a lag, which can diminish the performance benefits of the GPGPU parallelism [263]. Moreover, mastering the GPGPU programming usually requires time, since there are several differences in the way algorithms are developed for GPGPU in respect to the common CPUs (see examples in [264]). Anyway, the effort made for programming GPGPU usually pays off and today this has become a *de facto* standard for accelerating computation in a wide range of scientific fields [265]. In systems biology the GPGPUs provide a suitable platform for accelerating simulations [266].

A similar computer system that is largely used for parallel computing, with similar SIMD computation properties, is the many-core accelerator, such as the Intel© Xeon Phi™ [267–269] which has become a powerful computing platform used for a wide range of massive scientific computations [267]. Here we refer to [265] for an overview of the algorithms suited for SIMD architectures and to [270] for a recent comparisons of scientific applications running on the many-core and GPGPU systems.

The GPGPUs and the many-core accelerators can be extensively used to perform massive stochastic simulations. Li and Petzold [242], Komarov and D'Souza [244] and Nobile et al. [243] show that multiple runs of stochastic simulations can be efficiently implemented on a GPGPU, either using SSA or τ-leaping algorithm. However all of the current SSA and τ-leaping implementations for the GPGPU are made for single scale simulations only, and since the memory on the GPGPU usually can accommodate limited model sizes, the cost for performing large size SSA simulations

[5] The device's main memory is usually referred as the global memory.

on a GPGPU can be very high. In the case of DMSSA this cost increases even more. DMSSA requires nested SSA simulations, in order to calculate the concentration of chemical species involved in the fast reactions. These concentrations are then applied to the species engaging in the slow reactions of the outermost SSA. Implementing such nested procedure in a single kernel, may result in a poorly optimized SIMD program, and hence a low exploitation of the GPGPU parallelism is expected. Therefore, such implementation is much more suitable for a multi-core CPU platform.

For the purpose of multiple multi-scale stochastic simulations evaluation, we develop a simulator engine named *ParMSSA* in which the DMSSA algorithm is implemented on an OpenCL platform, which performs multiple simulations in parallel. The *ParMSSA* was built to target the multi-core CPU platforms in order to exploit the benefits of today fast multi-core processors.

A.3.2 The ParMSSA engine

We developed *ParMSSA*, short for *parallel multi-scale stochastic simulation algorithm*, a command line OpenCL engine written in C++, aimed for performing parallel stochastic simulations of gene regulatory networks that contain multiple non-cooperative TFBSs.

We decided to opt for OpenCL because of its portability to all types of parallel architectures: multi-core CPUs, GPGPUs and computing accelerators. However, writing an optimal OpenCL code, that will optimally run and exploit all the beneficial properties in all three parallel systems, is a hard task. Each of these parallel systems has a different memory organization that requires different type of optimizations.

ParMSSA contains a simulation engine which implements the DMSSA on a single OpenCL kernel. This kernel can be executed in parallel over multiple processing cores. Moreover, *ParMSSA* can also perform the Morris sensitivity analysis described in chapter 6, which requires to perform a large amount of simulations in order to calculate the elementary effects.

Implementation

ParMSSA is an object oriented application, built with the aim to be the foundation of a modular simulation framework for multi-scale systems and synthetic biology models. The engine implements the following functionalities:

- a *model descriptor*,

- the *DMSSA engine* and
- the *Morris screening experiment* (MSE).

The first functionality is responsible to parse the input data and to provide the necessary data structures to the DMSSA engine. The DMSSA engine provides a framework for performing large multi-scale simulations in parallel. The MSE instead consists of a set of functionalities aimed to ease the execution of the Morris sensitivity analysis.

Model descriptor

The model descriptor roughly consists of two mark-up language parsers that enable external user definitions of the model input data needed by *ParMSSA* to perform the stochastic simulations. A parser is available for importing and exporting the model data formatted with SBML (*Systems Biology Markup Language*) or YAML (*Yet-Another Markup Language*) format. Making the engine compliant with the SBML standard by using its API[6] [271, 272], guarantees portability of model definition and description between already existing tools, such as MATLAB® Simbiology [273] and Copasi [274][7]. The input data formatting and model definitions, as well as a quick starting guide of the *ParMSSA* are fully documented in section A.4.

DMSSA engine

The second functionality and the core of the *ParMSSA* engine is the DMSSA, which is implemented as an OpenCL kernel, and therefore able to be executed on any OpenCL platform. DMSSA engine is used both for independent model simulations and for performing model's evaluations in the Morris screening experiment.

The OpenCL kernel is optimized for running on multi-core CPUs or on many-core accelerators. The kernel itself is an example of coarse-grain parallelisation, because of the intrinsic independence of multiple DMSSA simulations. However, the optimization for GPGPU resulted to be impractical, due to the DMSSA data structures' requirements for large space of global memory with fast random access. Some of the main data structures, such as the binding-site matrix, were successfully optimized in

[6] http://sbml.org/Software/libSBML/docs/cpp-api/

[7] A full list of SBML compliant softwares is available at: http://sbml.org/SBML_Software_Guide/SBML_Software_Summary

order to minimize the data structure size in the device main memory. Also, the binding-site operations were optimized and reduced as fast bitwise shift logical operations, thus increasing the speed of the inner SSA, i.e. the bseSSA. The OpenCL implementation of DMSSA can highly benefit from caching and automatic vectorization — the first property is already available in all modern general purpose CPUs, whereas the automatic vectorization is a powerful functionality available in high-end compilers. Some suitable platforms for *ParMSSA* are therefore multi-core CPUs[8] and many-core accelerators, such as the Intel© Xeon™ Phi.

The OpenCL kernel relies on a variety of data structures that have to be defined and initialized in the main memory before the kernel can be fired. For this task the C++ framework, built around the OpenCL kernel, contains a data interface adapter, called MssaEngineDataAdapter, which provides an efficient translation of the model data into a class of ordered data structures that fit all the requirements of the OpenCL kernel arguments. When these engine data structures are defined also the initial values of all the species and parameter vectors are assigned. At the same time the engine also retrieves the parameters and constraints, such as the maximum number of parallel threads that the user defined and provided in the configuration file.

When the kernel is fired for multiple simulation instances, each thread accesses its own part (or a copy) of the data structures. This is a typical property of the OpenCL programming[9]. The memory concurrent access provided by the OpenCL runtime ensures that these accesses can be executed independently. This means that, when an argument in the Alg. 4.2 is a vector, in the OpenCL kernel implementation the same argument should be a vector of vectors, i.e. a matrix. Each thread is therefore designed to access one vector (column) per matrix, independently of other threads. This of course is necessary when a thread needs to change the content of its vector. For data structures that are read only, there is no need to enlarge the vector arguments to matrices. The DMSSA's slow and fast parameters vectors $\mathbf{k}^s$ and $\mathbf{k}^f$ are typical

[8] One of the advantages of running OpenCL on multi-core CPUs is the availability of the in-host-memory buffer allocation via the OpenCL directive `CL_MEM_USE_HOST_PTR`, which allows besides all, direct data structures sharing between the kernel and the host's code in the same physical memory.

[9] The same statement also holds for CUDA programming.

examples:

$$\mathbf{k}^s = \begin{bmatrix} k_1^s \\ k_2^s \\ \vdots \\ k_{m_s}^s \end{bmatrix}^\top, \quad \mathbf{k}^f = \begin{bmatrix} k_1^f \\ k_2^f \\ \vdots \\ k_{m_f}^f \end{bmatrix}^\top. \tag{A.38}$$

Both parameter vectors values are shared between all the OpenCL threads, due to the concurrent access that OpenCL runtime provides for constant data structures in the OpenCL device memory. The thing is quite different however in the case of vectors of slow and fast species. These vectors represent the system's *state* and contain values that will be often changed by the DMSSA. Thus these vectors should not be stored in the constant section of the OpenCL device memory. Given N threads, the vector of slow species $\mathbf{x}_i^s$ (and similarly fast species vector $\mathbf{x}_i^f$) used in thread i is stored in the matrix of species $\mathbf{X}^s$ in the following way:

$$\mathbf{X}^s = \left[\mathbf{x}_1^s, \mathbf{x}_2^s, \dots, \mathbf{x}_i^s, \dots, \mathbf{x}_N^s\right], \tag{A.39}$$

where $\mathbf{x}_1^s$, $\mathbf{x}_2^s$,... , $\mathbf{x}_N^s$ are column vectors of size m_s of the form:

$$\mathbf{x}_i^s = \begin{bmatrix} x_{1i}^s \\ x_{2i}^s \\ \vdots \\ x_{m_s i}^s \end{bmatrix}. \tag{A.40}$$

Hence, a collection of all supporting data structures needed by the DMSSA algorithm should be provided, together with the number of threads. Here we can find the binding-site matrices, the propensity functions supporting vectors and several other DMSSA related data structures. We refer directly to the *ParMSSA*'s source code documentation for more details regarding these arguments. One will probably argue why this data adaptation task cannot be achieved already in the descriptor procedure, since after the parsers load the data from the input files, all the required information is available for constructing the data structures needed by the DMSSA. We decided to keep the raw data available for any tools of the engine, so that each simulation engine can adapt this data for its own use through its own data adapter. This choice ensures modularity and independence between engines that may be developed in the future. We will see later

on, that also the Morris experiment needs such an adapter. The data adapter therefore provides all the properly formatted data necessary for the execution of the engine's OpenCL kernel.

The OpenCL kernel accepts also different types of activation rules for establishing the promoters activation or inhibition state when the bseSSA procedure of the DMSSA is simulating the transcription factors binding over the binding-site matrix (see section 4.2 for more details). We described in the previous chapters that in the case of promoters with multiple non-cooperative TFBSs the dynamics of their activation can profoundly affect the final response of the system. Since these rules cannot be generalized over all known models containing multiple non-cooperative TFBSs, the application of these rules has to be customized outside the OpenCL kernel, with the help of simple external headers directives.

DMSSA engine accepts also custom increase or decrease of a species' concentration at specific simulation time, in order to simulate perturbations of a specific species concentration in the simulated environment. As in the case of activation rules, such changes are also restricted to simple custom directives in external headers used by the OpenCL kernel during the preprocessing time.

Morris screening experiment

The third part of the *ParMSSA* code implements the Morris's screening experiment (MSE), which integrates both of the previous two parts. In order to calculate a simple elementary effect, at least two evaluations of the model response have to be performed: one with unmodified parameters and another with only one parameter modified by the factor Δ, see Eq. (5.6). For this reason, MSE is designed to run two (or more) parallel threads of DMSSA with a different value of a simple model's parameter. In short, given n model parameters, MSE configures the DMSSA engine to set n consecutive threads, each of which differs from previous by only one parameter, Each thread is then cloned u-times and executed in parallel. Here we would like to remind that running u identical parallel threads (or repeating it u times) with the same parameter configuration is necessary, if one wants to obtain statistical valid simulations.

Given n threads, or n computing units, we would like to perform on each thread a DMSSA simulation with different values of the input parameters. We can thus store the vector of reactions' parameters $\mathbf{k}_i^s$, such as the one defined in Eq. (A.38) used in

thread i, in the matrix of parameters $\mathbf{K}^s$ in the following way:

$$\mathbf{K}^s = \left[\mathbf{k}_1^s, \mathbf{k}_2^s, \dots, \mathbf{k}_i^s, \dots, \mathbf{k}_n^s\right], \tag{A.41}$$

where $\mathbf{k}_1^s$, $\mathbf{k}_2^s$,... , $\mathbf{k}_n^s$ are column vectors of size m_s of the form:

$$\mathbf{k}_i^s = \begin{bmatrix} k_{1i}^s \\ k_{2i}^s \\ \vdots \\ k_{m_s i}^s \end{bmatrix}. \tag{A.42}$$

Each element k_{ji}^s, $j = 1, 2, \dots, m_s$, represents the kinetic rate constant of the j-th slow reaction of the simulation running on the i-th thread. Similarly this definition holds also for the vector of fast reactions' parameters $\mathbf{k}_i^f$ and for the matrix of fast reactions' parameters

$$\mathbf{K}^f = \left[\mathbf{k}_1^f, \mathbf{k}_2^f, \dots, \mathbf{k}_i^f, \dots, \mathbf{k}_n^f\right], \tag{A.43}$$

where $\mathbf{k}_1^f$, $\mathbf{k}_2^f$,... , $\mathbf{k}_n^f$ are column vectors of size m_f of the form:

$$\mathbf{k}_i^f = \begin{bmatrix} k_{1i}^f \\ k_{2i}^f \\ \vdots \\ k_{m_f i}^f \end{bmatrix}. \tag{A.44}$$

In the same way we can also define the slow and fast species concentrations vectors $\mathbf{X}^s$ and $\mathbf{X}^f$ (see Eq. (A.39)). When running the Morris experiment however, as we mentioned above, u simulations with identical set of parameters are required to be performed. This means that the matrix $\mathbf{K}^s$ (or similarly $\mathbf{K}^f$) has to be stretched out as follow:

$$\mathbf{K}^s = \Big[\underbrace{\mathbf{k}_1^s, \mathbf{k}_1^s, \dots, \mathbf{k}_1^s}_{\substack{\text{the first } u \text{ parallel} \\ \text{threads run the same} \\ \text{set of slow reaction's} \\ \text{parameters } \mathbf{k}_1^s}}, \underbrace{\mathbf{k}_2^s, \mathbf{k}_2^s, \dots, \mathbf{k}_2^s}_{\substack{\text{the next } u \text{ parallel} \\ \text{threads run the same} \\ \text{set of slow reaction's} \\ \text{parameters } \mathbf{k}_2^s}}, \dots, \underbrace{\mathbf{k}_n^s, \mathbf{k}_n^s, \dots, \mathbf{k}_n^s}_{\substack{\text{the last } u \text{ parallel} \\ \text{threads run the} \\ \text{set of slow reaction's} \\ \text{parameters } \mathbf{k}_n^s}} \Big]. \tag{A.45}$$

The matrix of parameters $\mathbf{K}^s$ with size $m \cdot n$ from Eq. (A.41) is in Eq. (A.45) stretched to the size $m \cdot N$, where $N = n \cdot u$, and each $\mathbf{k}_i^s$ from Eq. (A.41) is cloned u-times. Hence the overall number of parallel threads that should be executed at once totals N.

Generally all the elements of $\mathbf{K}^s$, indexed with $i = 1, 2, \ldots, N$ can be defined as

$$\mathbf{k}_{i+1}^s = \mathbf{k}_i^s + \delta_i^s, \tag{A.46}$$

where

$$\mathbf{k}_i^s = \begin{bmatrix} k_{1i}^s \\ k_{2i}^s \\ \vdots \\ k_{ji}^s \\ \vdots \\ k_{mi}^s \end{bmatrix}, \quad \delta_i^s = \begin{cases} [0, 0, \ldots, \Delta_j^s, \ldots, 0]^\top, & \text{if} \quad (i+1) \bmod u = 1 \\ [0, 0, \ldots, 0, \ \ldots, 0]^\top, & \text{otherwise} \end{cases}.$$

A similar definition can be also derived for the parameters $\mathbf{K}^f$ as we can write

$$\mathbf{k}_{i+1}^f = \mathbf{k}_i^f + \delta_i^f, \tag{A.47}$$

where

$$\mathbf{k}_i^f = \begin{bmatrix} k_{1i}^f \\ k_{2i}^f \\ \vdots \\ k_{ji}^f \\ \vdots \\ k_{mi}^f \end{bmatrix}, \quad \delta_i^f = \begin{cases} [0, 0, \ldots, \Delta_j^f, \ldots, 0]^\top, & \text{if} \quad (i+1) \bmod u = 1 \\ [0, 0, \ldots, 0, \ \ldots, 0]^\top, & \text{otherwise} \end{cases}.$$

These definitions of $\mathbf{K}$ affect directly the format and size of data structures arguments of the DMSSA engine. For this reason a data adapter is needed for arranging the definitions depicted in Eq. (A.45) and Eq. (A.46) and to be included in the DMSSA simulation.

The next major step involves the creation of the matrix $\mathbf{R}^*$ of size $r \cdot k$ through orthogonal sampling. The matrix $\mathbf{R}^*$ must contain exactly r independently random generated samples from each distribution F_i. The i-th row of $\mathbf{R}^*$, namely $\mathbf{x}_i$ represents

a random sample of the k parameters. Here we use a different notation for the parameters of the MSE as in chapter 5. With the symbol $\mathbf{x}$ we denote those parameters for which the sensitivities are calculated. These parameters are not necessarily the same as the reaction kinetic parameters previously denoted as $\mathbf{k}^s$ or $\mathbf{k}^f$, but rather a superset of all the basic and composite parameters of the model. These includes the genes copy numbers c_m and the number of binding sites n_m for each promoter of the system.

The RTGM is then used to create the orientation matrices $\mathbf{B}^*_i$ of size $(k + 1) \cdot k$. These matrices are constructed from each row of $\mathbf{R}^*$ in the same way as shown in section 5.3.1.

The function implementing this procedure is called CreateRandomTrajectory. Since the matrix $\mathbf{R}^*$ contains r rows, this means that the whole MSE creates r different trajectories. Each matrix $\mathbf{B}^*_i$ represents one trajectory or sample in the parameters hyperspace and is created inside the procedure main loop. Intuitively, larger values of r lead to more accurate estimations of the average μ^* and the standard deviation σ, but at the same time increases the computational effort needed to calculate the elementary effects.

Another fact must not be forgotten here and that is, the consideration we made above for the type of interval we are dealing with. If a parameter's interval is logarithmically spaced, then the values of the Δs are not the same everywhere. This fact must be taken into account, and this is the reason why, after the call to the function CreateRandomTrajectory, we intentionally create another matrix $\mathbf{D}^*_i$ through the function createDeltaSignMatrix, where we store all the interval perturbations (or "jumps") defined by the orientation matrix $\mathbf{B}^*_i$. This allows us to find by how much a parameter was increased or decreased when calculating the elementary effects.

At this point, we have all the data needed for performing simulations. The MSEDataAdapter is responsible to prepare the data structures needed by the DMSSA engine for performing the simulations with the parameters values, set by the orientation matrix. The engine executes groups of u threads performing the same DMSSA instance for each row of $\mathbf{B}^*_i$.

Algorithm A.4

The Morris screening experiment (MSE) pseudocode of the procedure implemented in the *ParMSSA* engine.

Input:
- M (the model data - sbml/yaml file),
- C (the configuration INI file).

Output:
- The matrix $\mathbf{Y}$ of responses for the input model,
- the matrix $\mathbf{G}$ of binding gradients responses for the input model,
- the timestamps matrix $\mathbf{T}$,
- the matrix $\mathbf{R}^*$,
- the collection $\mathscr{D}$ of delta signs matrices.

procedure [$\mathbf{Y}$,$\mathbf{G}$,$\mathbf{T}$,$\mathbf{R}^*$,$\mathscr{D}$] = MorrisExperiment(M, C)
  Call the descriptor for the model and configuration file. The descriptor creates a model Ω from 3.10 with the arrangements from Def. 4.2 and a collection of data structures θ, including the set of rules $\mathscr{R}$, from the model and configuration input files M and C respectively.
  [Ω, Θ] = Descriptor(M, C)
  Create a RTGM matrix $\mathbf{R}^*$ with the orthogonal sampling.
  $\mathbf{R}^*$ = OrthogonalSampling(Ω, Θ)

  for each row $\mathbf{x}_i$ in $\mathbf{R}^*$ *do*
    Create an orientation matrix $\mathbf{B}^*$ and a delta sign matrix $\mathbf{D}^*$.
    $\mathbf{B}^*_i$ = CreateRandomTrajectory($\mathbf{R}^*$)
    $\mathbf{D}^*_i$ = CreateDeltaSignMatrix($\mathbf{B}^*_i$, Δ)
    For each row of $\mathbf{B}^*_i$, apply changes to each composite parameter of the system.
    [$\mathbf{K}$, $\mathbf{X}$, N] = MSEDataAdapter(Ω, Θ, $\mathbf{B}^*_i$, u)
    Run N parallel simulations with the DMSSA OpenCL kernel.
    [Y_i, G_i, T_i] = DmssaOpenCL(Ω, $\mathscr{R}$, N, $\mathbf{K}$, $\mathbf{X}$)
    Add responses, timestamps and their D matrices to results.
    $\mathbf{Y} = \mathbf{Y} + Y_i$ $\mathbf{G} = \mathbf{G} + G_i$ $\mathbf{T} = \mathbf{T} + T_i$ $\mathscr{D} = \mathscr{D} + \mathbf{D}^*_i$
  end for
end procedure

Therefore an entire trajectory can be executed by the DMSSA engine at once. The OpenCL engine is fired with a total of $N = (k+1) \cdot u$ parallel threads. Here we took advantage of the capabilities of the OpenCL drivers which can distribute the workload over multiple threads efficiently and automatically, without the necessity of any user intervention. More importantly all these independent threads can run asynchronously until all the N simulations are computed. This ensures a high exploitation of all the parallel computing units available on the OpenCL device. Now, if the OpenCL device is capable to run all N threads concurrently, then an entire trajectory of the Morris ex-

periment can be evaluated in the same time complexity as a simple run of the DMSSA. The entire Morris experiment hence costs $r \cdot O_{DMSSA}$, where O_{DMSSA} is the time complexity of the DMSSA given the size of its input data arguments. The function call DmssaOpenCL returns the model output response of N parallel instances of DMSSA. It differs from the Alg. 4.2 only by the arguments that it receives. The arguments $\mathbf{K}$ and $\mathbf{X}$, contains the parameters intervals samples that are used by DMSSA for computing the simulations with different input values (factors).

The main loop concludes a cycle by storing the responses of the DMSSA simulations. The main loop ends when r trajectories are computed and the results are properly stored inside the results folder. The entire MSE procedure is shown in Alg. A.4.

Now, all the elementary effects can be computed for all the simulations performed by the MSE and the two metrics μ^* and σ can be computed according to [196]. Since we have all the results already stored, these calculation can be performed also by a third part numerical software, such as Matlab®.

Random numbers generation

Last but not least, an important consideration about the implementation of any Monte-Carlo based simulation, involves the type of *pseudo-random numbers generator* (PRNG) used for stochastic simulations. A good PRNG is essential for achieving statistically independent results over multiple repetitions. One of the most praised PRNG based on linear recurrence is the Matsumoto's and Nishimura's *Mersenne Twister* (MT or MT19937) [275], which besides providing an extremely long period of $2^{19937} - 1$, it has the following benefits:

- it is statistical independent,
- it generates almost equally distributed numbers in the range $[0, 1]$,
- it has a higher performance compared to the default PRNGs from the standard language libraries,
- it does not require large amount of memory and
- multiple generated numbers are highly well distributed over multiple dimensions.

Despite not being a cryptographically secure however, the MT is regarded as one of the best *state-of-the-art* PRNG for stochastic simulations. In the light of this fact and since a typical long run of the DMSSA usually requires at most a coupled of billions of quickly generated random numbers, equally distributed in the range $[0, 1]$, the MT has been chosen to be used also for our simulations.

However, because the DMSSA engine relies on a OpenCL kernel for performing simulations, and since OpenCL API does not provide by default a MT PRNG, we needed to implement a custom MT for our purposes. Moreover, since the simulations are carried out in parallel, there is also a need to provide pseudo-random generated numbers in multiple concurrent OpenCL threads. Generating pseudo-random numbers in parallel is a well known problem in computer simulations and several solutions have been proposed already [276, 277]. The problem consists of creating pseudo-random numbers independently between multiple parallel threads. Here, the concept of independence means that the sequence of random numbers generated in each thread must be independent by one and other. If we are going to use multiple instances of the same MT declared with the same parameters in all threads, we will eventually create the same sequence of pseudo-random numbers in all the simulations' instances. Multiple MT must be initialized by different seeds, thus providing different initial states from which begin the linear recurrence and hence independent sequences[10]. This is one of the main solution that is generally adopted by simulation designers: select n randomly generated seeds by a local MT and then initialize in the n threads the same MT, but with a different seed value. But there is a hidden caveat for which this technique will eventually fail for large scale simulations, i.e. for large values of n. This caveat refers to the possible collisions that can be yielded by multiple sequences generated with the same MT but with two initial states that may lie in the MT period too close from each other. In fact when using the MT, or by means of any other linear recurrence based PRNG, the sum of two or more sequences, initiated by different initial states, may results in a sequence of already generated pseudo-random numbers. Hence there may exist a possible dangerous situation (with a not small probability of happening), where we might perform two (or more) simulations with the same sequence of pseudo-

[10] This is true for a short length of sequences, since the period of a classical MT is $2^{19937}-1$, which is $\approx 10^{6001}$ numbers long. The high k-distributions of dimensions also helps to sparse the initial states far away between each other inside the period. We refer to [275] for details about the properties of the linear recurrence used in the MT.

random generating numbers. Such situation should be avoided at any cost, if we want to preserve statistical independence and accuracy between stochastic simulations. This is also one of the main reason why we should not use the MT for generating pseudo-random numbers for secure cryptography, where even the slightest occurrence of two identical sequences can lead to catastrophic effects.

But let us calculate the probability of such occurrence in the case of multiple parallel instances of the DMSSA OpenCL kernel. Let be m the bit length of the pseudo-random generated numbers of the MT, and let be u the number of different threads of the same simulation we want to perform in parallel (this value may be the same u-value of repetitions defined by the user for achieving statistical valid stochastic simulations). Now the number of all possible different seeds N from which the MT can define the initial state of its linear recurrence is 2^m, therefore for a value of $m = 32$, the number of all different seeds is $2^{32} = 4,294,967,296$. Now, if we attempt to define u different threads for which each thread would have a different sequence of random numbers from the same MT, it is clear that this number u is limited by N. If we choose the initial u seeds pseudo-randomly, with an uniformly distribution, between the interval $\left[0;\ 2^{32}-1\right]$, then the probability $Pr[u,2]$ of the event that at least two threads of u will be initiated with the same MT state, i.e. a collision event, will be

$$Pr[u,2] = 1 - \frac{\binom{N}{u}}{\binom{N+u-1}{u}}. \tag{A.48}$$

In the Tab. A.1 we list some values of Pr[u,2] for different u, selected for performing DMSSA. We can see clearly that with probability of more than 2% we may face collisions when u is greater than 10000 repeats. Unfortunately in statistics, large sample populations are needed to provide more accurate estimations of the mean and standard deviation regarding some property that we want to generalize usually when testing hypothesis. But because the DMSSA responses in general quickly converge to the median trajectory, usually the user does not need more repetitions than few hundred, which makes the collision probability almost negligible. Therefore we also opted for these strategy by applying uniformly distributed pseudo-random seed to all the MT19937 OpenCL implementations, that will provide pseudo-random numbers sequences to all the simulation's instances fired by the DMSSA engine.

In the case where the number of repetitions must be higher than the reference value

Table A.1

Some sample values of the probability of collision occurring when $m = 32$ and when the user selected u threads for running u instances of the same configuration of the DMSSA.

u	$Pr[u, 2]$
20	$8.84756 * 10^{-8}$
50	$5.70435 * 10^{-7}$
100	$2.30502 * 10^{-6}$
500	$5.80896 * 10^{-5}$
1000	0.000232
10000	0.02301

of 10000, or in the case where the number of parallel threads needs to be very close to or higher of the maximum number of different initial seeds, several studies recommended the utilization of the *Mersenne Twister Dynamic Creator* (DCMT) [276], a parallel implementation of the known PRNG in which the identification number, also called thread ID or processor ID, is embedded in the MT initialization parameters, enabling thus the creation of completely different sequences with no common period. The main shortcoming of this technique is that can be computational demanding, since it needs to search an exponential space of values for the initialization of all the different MTs. Fortunately, these space search can be done *a priori*, and the initialization of such MT can be done very quickly. This is the strategy that is implemented and suggested in several code examples, such as in [278, 279]. Anyway, not any present literature decided to opt for this technique. In [243] for instance, authors used a MRG32K3A PRNG described in [280] to perform stochastic simulations. The results indeed proved to be statistically sounded. While Li and Petzold in [242], adopted successfully the DCMT [276], with also impressive results.

Furtherer, other implementations of parallel PRNGs that can take advantage also of SIMD architectures found in GPGPU, have been proposed in [281, 282]. Implementation of fully independent MTs on GPGPU threads resulted to be difficult, mainly because of the memory architecture inside the graphic processing units. Hence the

use of a parallel implementation for GPGPUs of the MT proposed in [281] is recommended, if simulations such as the DMSSA need to performed on GPGPU.

Concluding, DMSSA engine declares for n threads, n MTs with n different initial seeds. If the number of threads is small compared to the state space size then the probability of collision may be negligible and statistical independent stochastic simulations may be performed. Otherwise, if the number of threads is very large, the DCMT should be called and n different MTs should be computed.

A.3.3 OpenCL

OpenCL is an open standard for parallel programming for heterogeneous platforms [261]. Similarly to CUDA, OpenCL provides an API for exploitation of GPGPU. As mentioned briefly in the previous section a single GPGPU may usually contain several hundreds of processing cores (e.g. CUDA Cores) [259], and hence its use is suitable for performing parallel computations. Since CUDA provides its API for solely Nvidia proprietary GPGPUs, OpenCL enlarges the usability of its API to almost all GPGPUs of the most common manufacturers, i.e. Nvidia, AMD and Intel. But one of the main *pros* of the OpenCL API is that it also supports multi-core architectures (e.g. AMD© Opteron™ or Intel© Core™ and Xeon™ CPUs) and many-core architectures as well, such as the Intel© Xeon™ Phi accelerator[11] [269, 283, 284]. Programming with OpenCL is similar as with CUDA. The OpenCL standard provides a C-language specification [286], that allow exploiting the powerfulness of the C language for parallel programming. A C++ wrapper is also available [287].

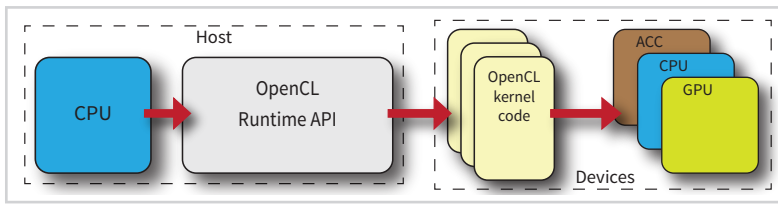


Figure A.1

The OpenCL API environment from Khronos™ Group [285]. The kernel code (once is compiled) can run directly on the devices, which can be either a GPGPU (GPU), an accelerator (ACC) or the CPU itself.

Unlike OpenMP, where the parallel programming is much like sequential programming (e.g. parallelisation is achieved by definitions of simple *pragmas* before some loop

[11] https://goo.gl/UcRL4O

initializations), OpenCL programming consists in dividing the program in two conceptual different parts. The first part contains the sequential code, which has to be run by the host CPU. The second part instead contains a short function called *kernel*, which is responsible to implement the task that needs to be accelerated on the target device(s). This differentiation is depicted in Fig. A.1. When an OpenCL program is running, the host computer first initializes the device and then it copies the data, needed to be processed in parallel, to the device global memory (see Fig. A.2). The device driver then distributes the same kernel code to all processing cores on the device. When the kernel function is then called by the host CPU code, all the processing cores on the device execute the same kernel code. Hence each core can execute a *thread* of the kernel. Usually these threads are organized within a specific matrix geometry, that user can directly configure. Every thread can be uniquely identified by an index that defines the thread's position inside the geometry matrix. These indices can be used for a variety of

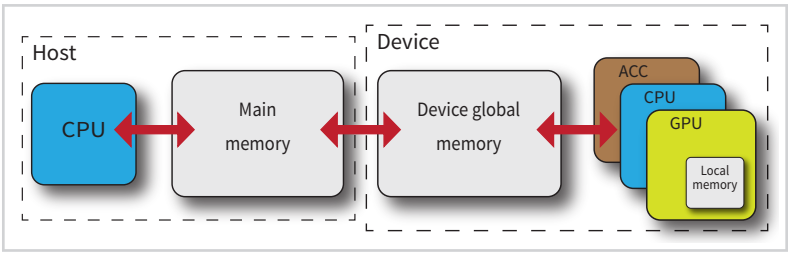


Figure A.2

The OpenCL memory hierarchy. Notice, that if the chosen OpenCL device is a multi-core CPU then the host's and the device's main memory are the same.

purposes. Most commonly they represent the easiest way to access specific portion of data by each thread running the same kernel. More rarely they are used to differentiate operations that the kernel may specify. This is not however a recommend technique for GPGPU systems, since by differentiation of the kernel execution, the threads synchronization is often lost. Threads synchronization is one of the reason why GPGPU systems can achieve so great parallelisation performances. We refer to [288–290] for a more complete explanation of the OpenCL thread geometry configuration, as well as a complete introduction to OpenCL programming.

A.4 ParMSSA user guide

A.4.1 Command line options

ParMSSA accepts a plethora of command line options, which are all listed below. All the options, can be provided by a configuration INI file, except for the general and the required options.

```
General options:
-v [ --version ]                Print version string and exit.
-h [ --help ]                   Produce the help message and exit.


Required options:
-m [ --model ] arg              The name of a SBML or YAML file containing
                                 the model description (e.g. model.yaml or
                                 model.sbml).

Configuration:
-c [ --config ] arg             The name of an ini file containing the
                                 configuration for the simulations (e.g.
                                 config.ini).

-l [ --samples ] arg (=10)      The number of points that the simulation
                                 should generate. Default value: 10

--sampling-step arg (=1000)     The number of SSA steps between two data logs.
                                 This value is used for sampling the SSA
                                 results. Usefull for decreasing results'
                                 resolution and overall results' size.
                                 Default value: 1000

-n [ --threads ] arg (=1)       The number of parallel threads "n" that the
                                 simulation will perform in order to generate
                                 exactly "n" simulations. This flag is
                                 overwritten when performing simulations for
                                 sensitivity analysis (-s flag) in the case
                                 the number of needed simulations are greater
                                 than "n". Default value: 1

-p [ --path ] arg (=results)    The path (folder) in which store the
                                 simulation results. Default value: results
```

```
--with-stats                    Print on the stnadard output the current
                                telemetry from the engine. Default: false

--with-log-file arg (=stdout)   The log file in which store the simulation
                                log. Default value: stdout

-d [ --device ] arg (=cpu)      The device to use for simulations. Default
                                device: cpu

--species-to-log arg            The name of all the species for which we want
                                to log their simulation responses. In the
                                case of -s flag these are the species over
                                which the sensitivity analysis will be
                                performed. By default all species are logged.

--gradients-to-log arg          The names of all the promoter species for
                                which we want to log their simulation
                                gradients.

--activation-rule arg           The rule used to determine the activation
                                state of promoters with multiple binding
                                sites. The possible values are: 'majority',
                                'all-or-none', 'singular' and 'additive'.
                                If no rule is provided then by default, the
                                majority rule is used.

--no-multiscale                 Perform a classic SSA simulation instead of
                                using the multiscale engine. Warning: the
                                use of the flag MUST BE USED ONLY for models
                                with no multiple binding sites promoters,
                                i.e. for custom reactions definitions ONLY!

-s [ --sensitivity ]            Perform a Morris simulation experiment over
                                the model. The results contain all the data
                                for global Morris sensitivity analysis
                                (Matlab™ required). By default no
                                sensitivity is perform on the model, unless
                                the flag -s is given in command line.
                                Default: false

--repetition-factor arg (=1)    The number of times a simulation with the
                                same set of parameters should be run.
                                Default: 1
```

```
--p-value arg (=10)            An even number representing the p-value in
                               the Morris experiment. Default: 10

--r-sample arg (=4)            The number of different random samples
                               (trajectories) from each elementary effect
                               distribution to perform in the Morris
                               experiment. Default: 4

-D [ --offset ] arg            The delta offset (D) of the p-value. The
                               delta value (Δ) in the Morris experiment is
                               then calculated as: D/(p-1). Default value:
                               p-value/2

--print-model                  Print in verbose mode the model data
                               (species and reactions) constructed from
                               the given model configuration file (i.e.
                               sbml file). Default: false
```

A.4.2 Model description

ParMSSA allows model definitions in both the *systems biology markup language* (SBML) and the *yet another markup language* (YAML). SBML is an XML language extension developed for model definitions in systems biology, while YAML is a simple JavaScript Object Notation (JSON) derived language. The engine accept model definitions provided by an external file `.yaml` or `.xml` which has to be provided as command line argument when running the engine.

Model definition with YAML

The model description can be defined as a simple YAML file for which *ParMSSA* requires the following fields:

1. *name*; the model name defined in header, mandatory for every YAML file,
2. *proteins*; the list of proteins involved in the model,
3. *bindingsites*; the list of all binding sites occurring in the model,
4. *genes*; the list of all the complexes *binding site–promoter–coding sequence*,
5. *reactions*; the list of all additional reactions not specified by the model implicitly,
6. *params*; the list of all qualitative parameters of the model and

7. *quantitative_params*; the list of all quantitative parameters such as concentration and copy number of the model.

Proteins

The *proteins* field contains the description of all the proteins involved in the model. Each protein is defined as a list containing: name, the protein length in base pairs, the name of the parameter representing the degradation rate, the name of the parameter of the protein's mRNA degradation rate, the concentration parameter and the mature/de-mature mRNA parameters names. Each of the proteins parameters must however be defined in the *params* field.

Code A.1

A simple proteins list containing one protein definition.

```
proteins:
    -   name:           protein_A
        length:         1071
        kdeg:           k_deg_protein_A
        kdegmrna:       k_deg_mRNA
        kmrnamature:    k_protein_A_mrna_mat
        kmrnademature:  k_protein_A_mrna_demat
        concentration:  protein_A_concentration
```

Binding sites

Binding sites list definition is made in a similar way. Each binding site contain the information about the protein kinetic binding velocity (k_{on}) and the unbinding rate (k_{off}). Each binding site need however to specify the type of effect that the protein should have while bounded to the binding site. This effect is specified within the type attribute. This attribute can acquire the following values:

$$\begin{cases} 0 & \rightarrow \text{ the bounded protein has no effect} \\ 1 & \rightarrow \text{ the bounded protein has inhibition effects} \\ 2 & \rightarrow \text{ the bounded protein has activation effects} \end{cases}$$

Code A.2

A binding sites list containing two definitions.

```
bindingsites:
    -   name:       bs_A
        protein:    protein_A
        type:       1
        kon:        k_on_protein_A
        koff:       k_off_protein_A

    -   name:       bs_B
        protein:    protein_A
        type:       2
        kon:        k_on_protein_A
        koff:       k_off_protein_A
```

Genes

Genes list definition contains the name, the products and the list of TFBSs of each gene of the GRN. In the gene definition there must be also defined the two reference kinetic constants parameters of transcription and translation. The initial copy number of the each gene can also be defined as a parameter attribute "copyn".

Code A.3

A genes list containing one gene definition.

```
genes:
    -   name:           gen_A_1
        bindingsites:   A A A A A A A A A A
        products:       TAL_A_VP16
        ktrsc:          k_trsc_tal_vp16
        ktrsl:          k_trsl_tal_vp16
        copyn:          N_plasmids
```

Reactions

Reactions list definition allows one to define any reaction that occur in a GRN. Usually are those defined as in 3.3, however the engine is able to process any other reaction, as long as it is defined as a list containing the following attributes: "type", "products", "reactants" and "params". The following code illustrates an example.

Code A.4

A reactions list containing one reaction definition.

```
reactions:
    -   reaction:
        type:           0

        products:
            -   name:       PROTEIN_FOR_NEGATIVE_CONTROL
                quantity:   1

        reactants:
            -   name:       protein_A
                quantity:   2

        params:
            -   kconst:     k_protein_for_negative_control
```

Parameters

The parameters list contains all the parameters definitions (name, values, measurement unit, type) for all the kinetic constants parameters used in the model. Here there is an example:

Code A.5

A parameter list containing two parameters definitions.

```
params:
    -   name:       k_deg_protein_A
        type:       0
        value:      0.0308065
        unit:       s^-1

    -   name:       k_protein_for_negative_control
        type:       0
        value:      1e-05
```

Quantitative parameters

Quantitative parameters refer mainly to the "non-kinetic constants" parameters needed by the model, e.g. initial concentrations, number of binding sites of a particular cis-regulatory module, or the initial successful plasmid transformation concentration.

Code A.6

A quantitative parameters list containing three parameters definitions.

```
quantitative_params:
    -   name:       k_protein_concentration
        type:       1
        value:      100000

    -   name:       N_binding_sites
        type:       1
        value:      10

    -   name:       N_plasmids
        type:       1
        value:      100
```

ParMSSA allows one model definition per YAML file.

Model definition with SBML

The model definition with SBML is similar as the YAML definition. The DMSSA requires detail descriptions of the binding sites and the relative gene data. Unfortunately SBML does not provide a default included fitting object structure such as for the protein and kinetic parameters. Hence we defined the model description inside the `<annotation>` tag inside the SBML default model object to enable compatibility for external XML parsers:

Code A.7

SBML model description template parsed by the *ParMSSA*.

```
<sbml>                  <!-- the main xml tag -->
<model>             <!-- the default SBML model tag -->
<annotation>    <!-- the main SBML annotation tag for model -->

<parmssa xmlns="http://lrss.fri.uni-lj.si/bio" name="myModel">
<description>
<!-- the model description -->
</description>
</parmssa>
</annotation>
</model>
</sbml>
```

```
<proteins>
<protein name="PROTEIN_FOR_NEGATIVE_CONTROL" length="2700">
<parameters>
<kdeg name="k_deg_TAL" type="0" value="0.00308065" default="0" default_max_range
      ="0" default_min_range="0" range_max_factor="0" range_min_factor="0"
      range_max_value="0" range_min_value="0"/>
<concentration name="k_protein_concentration" type="0" value="100000" default="0"
       default_max_range="0" default_min_range="0" range_max_factor="0"
      range_min_factor="0" range_max_value="0" range_min_value="0"/>
</parameters>
</protein>

<protein name="TAL_A_KRAB" length="2800" sequence="
      ATGCCTTTGGTAAATTGGGGCCCGGCAAAGCG">
<parameters>
<kdeg name="k_deg_TAL" type="0" value="0.00308065" default="0" default_max_range
      ="0" default_min_range="0" range_max_factor="0" range_min_factor="0"
      range_max_value="0" range_min_value="0"/>
<kdegmrna name="k_deg_mRNA" type="0" value="0.00154033" default="0"
      default_max_range="0" default_min_range="0" range_max_factor="0"
      range_min_factor="0" range_max_value="0" range_min_value="0"/>
</parameters>
</protein>
</proteins>
```

```
<bindingsites>
<bindingsite name="A" protein="TAL_A_VP16" type="1">
<parameters>
<kon name="k_on_vp16" type="0" value="0.1" default="0" default_max_range="0"
      default_min_range="0" range_max_factor="0" range_min_factor="0"
      range_max_value="0" range_min_value="0"/>
<koff name="k_off_vp16" type="0" value="1e-09" default="0" default_max_range="0"
      default_min_range="0" range_max_factor="0" range_min_factor="0"
      range_max_value="0" range_min_value="0"/>
</parameters>
</bindingsite>
</bindingsites>
```

```
<genes>
<gene name="gen_A_1" promoterType="3" modelPolymerase="no">
<bindingsites>
<bindingsite name="A"/>
```

```
<bindingsite name="A"/>
<bindingsite name="A"/>
<bindingsite name="A"/>
<bindingsite name="A"/>
<bindingsite name="A"/>
<bindingsite name="A"/>
<bindingsite name="A"/>
<bindingsite name="A"/>
<bindingsite name="A"/>
</bindingsites>
<products>
<product name="TAL_A_VP16"/>
</products>
<parameters>
<ktrsc name="k_trsc_tal_vp16" type="0" value="0.0185185" default="0"
      default_max_range="0" default_min_range="0" range_max_factor="0"
      range_min_factor="0" range_max_value="0" range_min_value="0"/>

<ktrsl name="k_trsl_tal_vp16" type="0" value="0.0444444" default="0"
      default_max_range="0" default_min_range="0" range_max_factor="0"
      range_min_factor="0" range_max_value="0" range_min_value="0"/>

<copyn name="N_plasmids_gen_A_1" type="1" value="100" default="0"
      default_max_range="0" default_min_range="0" range_max_factor="0"
      range_min_factor="0" range_max_value="0" range_min_value="0"/>
</parameters>
</gene>
</genes>
```

```
<reactions>
<reaction type="0">
<reactants/>
<products>
<product name="PROTEIN_FOR_NEGATIVE_CONTROL" quantity="1"/>
</products>
<parameters>
<kconst name="k_protein_for_negative_control_param" type="0" value="1e-05"
      default="0" default_max_range="0" default_min_range="0" range_max_factor="0"
       range_min_factor="0" range_max_value="0" range_min_value="0"/>
</parameters>
</reaction>
</reactions>
```

Razširjen povzetek

B

V zadnjem desetletju je sintezna biologija dosegla izjemne uspehe tako na različnih znanstvenih področjih, kot tudi v prvih komercialnih produktih, kot na primer v farmakoloških, okoljevarstvenih in energentskih aplikacijah [21]. Primer iz farmakologije so biološka zdravila zasnovana na bioloških (genskih) stikalih. Genska stikala predstavljajo jedro kakršnegakoli sistema zmožnega procesiranja informacij v bioloških sistemih [15, 59]. Ti so v večini primerov zasnovani na *gensko regulatornih omrežjih* (GRO), katerih delovanje lahko do določene mere prilagajamo našim potrebam [29]. Načrtovanje gensko regulatornih omrežij z vnaprej določenimi funkcionalnostmi večinoma ni premočrten postopek in zato ponavadi zahteva veliko količino eksperimentalnega dela. Pristopi računalniškega modeliranja lahko ključno pripomorejo k minimizaciji časa in stroškov načrtovanja tovrstnih sistemov. Računalniški modeli predstavljajo osnovo za kvantitativno ocenjevanje odzivanja načrtovanih sistemov pred njihovo eksperimentalno realizacijo [66–68].

Biološki sistemi večinoma izražajo robustno obnašanje, ki je posledica dolgega procesa evolucije [69]. Pri načrtovanju sintetičnih bioloških sistemov je po drugi strani zelo težko predvideti vse možne interakcije, ki jih bo načrtovani sistem imel z okoljem, v katerega bo umeščen. Njihovo delovanje je tako lahko zelo občutljivo na vplive iz okolja kot so sprememba temperature, pH faktorja ter sevanje, ki se pogosto odražajo v šumu. Omejevanje vplivov šuma in posledično napak v delovanju sistema oziroma preklopnih funkcij, ki jih z njim realiziramo, je ključnega pomena pri načrtovanju sintetičnih bioloških sistemov. Razširjen nabor metrik, ki omogočajo vzpostavitev neposrednejšega pristopa za načrtovanje gensko regulatornih omrežij kot preklopnih gradnikov, je v preteklosti že bil vzpostavljen [67]. Pomanjkljivost teh metrik je predvsem v odsotnosti kriterija za ocenjevanje robustnosti sistema v različnih okoljskih pogojih. Robustnost predstavlja ključni faktor adaptacije v evolucijskem procesu bioloških sistemov [70]. V biologiji celice je robustnost lastnost, ki omogoča ustreznost njenega odziva tudi v nepredvidljivih okoliščinah (npr. pri bakterijski kemotaksi) [71]. Kvantitativno vrednotenje robustnosti je ključnega pomena pri razvoju zanesljivih in skalabilnih bioloških preklopnih sistemov z novimi funkcionalnostmi.

Čeprav splošna kvantitativna mera za ocenjevanje robustnosti še ni bila vzpostavljena, teorija kontrole (angl. *control theory*) že veliko let ponuja metodologije za njeno ocenjevanje. V zadnjem desetletju je bilo predlaganih veliko metrik za vrednotenje robustnosti na področju bioloških sistemov [72–78]. Nekatere raziskave so pokazale, da se lahko robustnost vrednoti tudi na podlagi metod občutljivostne analize (angl. *sensiti-*

vity analysis) [79, 80]. Primer take metode je analiza občutljivosti parametrov [81–83], katere prednost je v njeni zmožnosti usmerjanja eksperimentalnega dela. Rezultati analize so namreč tisti vhodni parametri modela, ki najbolj vplivajo na delovanje sistema in jih je zato smiselno najbolj temeljito preučiti [84].

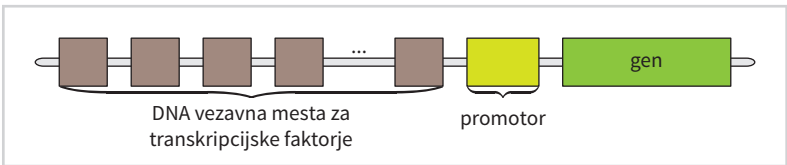
Večino obstoječih metod za izvedbo občutljivostne analize lahko apliciramo zgolj na deterministični modelih, ki pa so nezmožni upoštevanja dinamike šuma. V literaturi je sicer možno zaslediti nekatere izjeme [85–87]. Degasperi in ostali [206] so predlagali metodo za izvedbo občutljivostne analize v stohastičnih modelih, ki izražajo bistabilnost. Gunawan in ostali [205] so predlagali metodo za direktno izvajanje občutljivostne analize za enostavne diskretne stohastične modele gensko regulatornih omrežij, kot je na primer preklopno stikalo. Komorowski in ostali [207, 291] so z ovrednotenjem koeficientov lokalne občutljivosti posameznih parametrov izvajali občutljivostno analizo na nekaterih ključnih zgledih modelov v sistemski biologiji. Večina predlaganih metod v literaturi je zasnovana na skupni strategiji, pri kateri je občutljivostna analiza aplicirana na deterministične modele. Do teh modelov lahko pridemo z večkratnimi aproksimacijami kemijske glavne enačbe (angl. *chemical master equation*). Glavne slabosti te strategije so, da z njo ne moremo izvesti globalne občutljivostne analize in da je ne moremo aplicirati na določene vrste modelov v sintezni biologiji, kot so na primer stohastični večnivojski (angl. *multi-scale)* modeli [94, 95].

Preklopne funkcije so v bioloških sistemih ponavadi implementirane v okolju z veliko količino šuma. To lahko predstavlja velik problem pri načrtovanju kompleksnih in robustnih logičnih struktur. Načrtovanje bioloških sistemov z visoko stopnjo skalabilnosti in zanesljivosti delovanja tako postane zelo zahteven izziv. Stohastično modeliranje omogoča implicitno obravnavanje šuma znotraj odziva preklopa logičnih struktur [88–90]. Metode za stohastično modeliranje so obenem lahko računsko zelo zahtevne, predvsem ko je število prostih vezavnih mest na promotorju veliko. Take sisteme na primer predstavljajo platforme na osnovi transkripcijsko-aktivacijskih efektorjev, t.i. TAL efektorjev (angl. *transcription activator like effectors*) [42, 91]. Učinkovita strategija za izboljšanje računske kompleksnosti je vzpostavitev in izraba stohastičnega večnivojskega modeliranja [93–95]. Ta pristop predpostavlja, da lahko veliko število sicer odvisnih kemijsko-kinetičnih podsistemov pri določenih pogojih privzamemo za medsebojno neodvisne. Podsistemi se posledično lahko obravnavajo v ločenih časovnih okvirjih. S stohastičnim modeliranjem hkrati ohranjamo implicitno obravnavanje šuma znotraj vzpostavljenega modela. Obravnavanje podsistemov v različnih časov-

nih okvirjih omogoča kvalitetno aproksimacijo kinetike sistema in zmanjšuje računsko zahtevnost reševanja problema. Kljub temu, da je bilo v zadnjih letih vzpostavljenih veliko število metodologij za občutljivostno analizo tudi na področju sintezne biologije, metod za izvedbo občutljivostne analize stohastičnih večnivojskih modelov še vedno nimamo na razpolago. Vzpostavitev metodologije, ki bi te pomanjkljivosti odpravila, je ključnega pomena za načrtovanje robustnih sintetičnih gensko regulatornih omrežij in posledično za razvoj kompleksnejših sintetičnih bioloških sistemov zmožnih procesiranja informacij.

V disertaciji smo razvili metodologijo za izvedbo občutljivostne analize v stohastičnih večnivojskih modelih gensko regulatornih omrežij, ki vsebujejo večkratna nekooperativna DNA vezavna mesta za transkripcijske faktorje. Metodologijo je možno aplicirati neposredno v proces načrtovanja sintetičnih bioloških sistemov zmožnih procesiranja informacij.

Gensko Regulatorna Omrežja Kontrola genske ekspresije lahko poteka na osnovi pospešitve ali inhibicije transkripcije DNA v *sporočilno RNA* molekulo (angl. *messenger RNA* – mRNA). To je možno regulirati na osnovi vezave dveh vrst transkripcijskih faktorjev, tj. aktivatorjev in represorjev, na vezavna mesta DNA v neposredni bližini promotorjev. Aktivatorji povečajo afiniteto vezave RNA polimeraze na promotor, kar pospeši pričetek transkripcije opazovanega gena. Represorji po drugi strani preprečijo vezavo RNA polimeraze na promotor, kar posledično onemogoči transkripcijo gena. Rezultat transkripcije je mRNA molekula, ki vsebuje informacijo o določenem proteinu. Ta informacija se nato v ribosomih dekodira v procesu translacije, katere produkt je končni sintetizirani protein. V primeru, da mRNA vsebuje informacijo o določenem transkripcijskem faktorju, ki aktivno sodeluje pri regulaciji nekeka drugega gena, opazovani geni sestavljajo t.i. *gensko regulatorno omrežje* (angl. *gene regulatory network* – GRN).



Slika B.1

Večkratna vezavna mesta za transkripcijske faktorje v neposredni bližini promotorja.

V določenih primerih imajo promotorji lahko več kot eno samo vezavno mesto za

transkripcijske faktorje (glej sliko B.1). V teh primerih postane aktivnost promotorjev odvisna od zaporedja vezanih tranksripcijskih faktorjev na vezavna mesta v bližini promotorjev. Primeri pravil, ki določajo aktivnost promotorjev reguliranih z vezavo aktivatorskih proteinov so [105]:

- pravilo *vsi-ali-nihče*,
- *singularno* pravilo,
- pravilo *aditivnosti*.

Pri pravilu vsi-ali-nihče je promoter aktiviran šele, ko so aktivatorji vezani na vsa vezavna mesta, pri pravilu singularnosti pa je promotor aktiviran že v primeru, ko je na vezavnem mestu vezan en sam aktivator. Pravilo aditivnosti opisuje primer, ko vsak vezani transkripcijski faktor pripomore k večji aktivaciji promotorja. Genska regulacija na osnovi več DNA vezavnih mest predstavlja poseben izziv v računski biologiji, saj ga je zelo težko natančno modelirati.

Stohastično modeliranje Osnova stohastičnega modeliranja gensko regulatornih omrežij je stohastični simulacijski algoritem (angl. *stochastic simulation algorithm* – SSA) [175]. Ta je zasnovan na predstavitvi delovanja gensko regulatornih omrežij z množico kemijskih reakcij. Te reakcije opisujejo spreminjanje koncentracij kemijskih zvrsti, ki predstavljajo posamezno komponento sistema. Med opazovane kemijske zvrsti lahko uvrstimo proteine, transkripcijske faktorje, promotorje, gene in mRNA molekule. Algoritem SSA omogoča pridobitev časovne evolucije sistema glede na podani model, njegove parametre in njegovo začetno stanje. Koraki algoritma simulacije so prikazani v Alg. B.1.

Algoritem B.1

Stohastični simulacijski algoritem (SSA).

Vhodni podatki:
- množica reakcij, ki predstavlja delovanje izbranega GRO
- največji dovoljeni čas simulacije t_{max}

Izhodni podatki:
- časovna evolucija koncentracij x skozi čas t.

procedura SSA

1. Določi vse funkcije nagnjenosti $a_j(x)$ in njihovo vsoto $a_0(x)$.

2. Monte Carlo korak: izračunaj τ ter indeks j s pomočjo enačb (B.1) in (B.2).
3. Povečaj časovni korak $t \leftarrow t + \tau$ in izvedi spremembe koncentracij v vektorju koncentracij x glede na izbrane reakcije.
4. Shrani par vrednosti (x, t) in se vrni na korak 1, če je čas $t < t_{max}$.

konec

Algoritem določi čas med izvedbo dveh reakcij τ, kot

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

Iz množice M različnih reakcij, algoritem SSA izbere j-to reakcijo na osnovi ruletnega pravila, ki ga lahko formalno predstavimo z enačbo

$$\min_j \left\{ \sum_{k=1}^{j} a_k(x) \geq r_2 a_0(x) \right\}. \tag{B.2}$$

Pri tem sta r_1 in r_2 dve naključni števili, ki sta enakomerno porazdeljeni v intervalu $(0, 1)$. x predstavlja vektor trenutnih koncentracij vseh kemijskih zvrsti sistema, $a_j(x)$ pa predstavlja *funkcijo nagnjenosti* (angl. *propensity function*) j-te reakcije, tj. utež, ki posredno določa verjetnost izvedbe j-te reakcije. $a_0(x)$ predstavlja vsoto vseh uteži.

Enačbi (B.1) in (B.2) predstavljata *Monte Carlo* korak algoritma SSA. Časovno evolucijo sistema lahko dobimo z izvajanjem teh dveh korakov za poljuben čas.

Algoritem SSA predstavlja natančno aproksimacijo rešitve *kemijske glavne enačbe* (angl. *chemical master equation* - CME), zato lahko z njim simuliramo poljubno gensko regulatorno omrežje.

Stohastični večnivojski simulacijski algoritem Prednost algoritma SSA je enostavnost in natančnost pri izračunu časovne evolucije sistema kemijskih reakcij. Glavna pomanjkljivost je počasnost izvedbe simulacij pri velikem številu opazovanih kemijskih reakcij in pri visokih koncentracijah kemijskih zvrsti. Delovanje algoritma lahko pohitrimo z večnivojskimi pristopi, ki jih lahko uporabimo, ko se nekatere reakcije izvajajo veliko bolj pogosto kot ostale. Sistemom, ki izkazujejo to lastnost, pravimo tudi *togi sistemi* (angl. *stiff systems*). Reakcije lahko v tem primeru razdelimo na hitre in počasne. Sistemi kemijskih reakcij gensko regulatornih omrežij, ki vsebujejo več DNA

vezavnih mest za transkripcijske faktorje, predstavljajo tipičen primer togih sistemov. Pri teh sistemih je potrebno simulirati zelo veliko hitrih reakcij vezave in disociacije transkripcijskih faktorjev v primerjavi z bolj počasnimi reakcijami, ki zadevajo gensko ekspresijo in degradacijo opazovanih kemijskih zvrsti. Njihove simulacije lahko pohitrimo z *večnivojskimi* (angl. *multi-scale*) simulacijskimi algoritmi. Ti predpostavljajo, da lahko časovno evolucijo hitrih reakcij aproksimiramo z njihovim stabilnim stanjem (angl. *steady state*). Dodatno komplikacijo obravnave velikega števila nekooperativnih vezavnih mest predstavlja odvisnost stanja promotorja od trenutnega zaporedja vezanih transkripcijskih faktorjev. V primeru n-nekooperativnih vezavnih mest, je število stanj enako

$$(k+1)^n, \tag{B.3}$$

kjer je k število različnih tranksripcijskih faktorjev, ki tekmujejo za vezavo na ista DNA vezavna mesta. V tem primeru je število različnih reakcij vezave in disociacije, ki jih mora SSA simulirati, enako

$$\sum_{i=1}^{n} \binom{n}{i} i\, k^{n-i+1}. \tag{B.4}$$

Problem aplikacije večnivojskih metod na gensko regulatorna omrežja, ki vsebujejo več DNA vezavnih mest, postane določanje stanj promotorjev, ki poveča število reakcij do te mere, da le-teh v modelu ne moremo učinkovito predstaviti.

V disertaciji smo predlagali učinkovit algoritem na osnovi vgnezdenih SSA, ki rešuje predhodno opisani problem. Algoritem predstavi vsa DNA vezavna mesta za m-ti promotor v gensko regulatornem omrežju ob določenem času t z matriko $\mathbf{B}_m(t)$, katere elementi so definirani kot

$$b_{i,j}^m(t) = \begin{cases} 0; & \text{če je vezavno mesto } \textit{nezasedeno}, \\ 1; & \text{če je na vezavno mesto vezan } \textit{aktivator}, \\ -1; & \text{če je na vezavno mesto vezan } \textit{represor}. \end{cases} \tag{B.5}$$

Velikost matrike $\mathbf{B}_m(t)$ je odvisna od števila kopij m-tega promotorja v celici, ki določa število vrstic, in od števila DNA vezavnih mest, ki določa število stolpcev matrike. Takih matrik imamo toliko, kolikor je različnih promotorjev z več DNA vezavnimi mesti ($m = 1, .., M_p$). Algoritem simulira časovno evolucijo sistema, tako da izvede dve različni instanci SSA: glavno in vgnezdeno. Z uporabo vgnezdene SSA lahko simuliramo vezavo in disociacijo transkripcijskih faktorjev na posamezno DNA vezavno mesto, z

uporabo glavnega SSA pa počasne reakcije, to so reakcije transkripcije, translacije in degradacije. Vgnezdeni SSA se tako uporablja za simulacijo hitrih reakcij. Ta se izvede za čas, ki ga določi zunanji SSA. Vgnezdeni SSA za GRO z več neekoperativnimi DNA vezavnimi mesti smo poimenovali *bseSSA* (angl. *binding sites evolution stochastic simulation algorithm*) in je orisan v Alg. B.2.

Algoritem B.2

Algoritem *bseSSA* za simulacijo hitrih reakcij vezave in disociacije transkripcijskih faktorjev na več DNA vezavnih mest.

Vhodni podatki:
- vektor kemijskih zvrsti (z začetnim stanjem $X^f(0)$), ki se pojavljajo v hitrih reakcijah
- množica matrik vezavnih mest $\mathscr{B}$ s kardinalnostjo M_p

Izhodni podatki:
- časovna evolucija sistema $X^f(t)$.

procedura BSESSA($X^f(0)$, $\mathscr{B}$)
Določi začetni čas simulacije.
$t_f = 0$
Preštej število aktiviranih, represiranih in praznih vezavnih mest za vsako matriko $\mathbf{B}_m$ v $\mathscr{B}$ in določi koncentracijo aktiviranih, represiranih in praznih promotorjev v $X^f(0)$ na osnovi teh števil.
for all $m = 1, \ldots, M_p$ *do*
  Za prazne promotorje
  $X^f_{pr^{f}_{m}}(0) = \left|\{b^m_{ij}\}\right|$
  Za aktivirane promotorje
  $X^f_{pr^{f+}_{m}}(0) = \left|\{b^{m+}_{ij}\}\right|$
  Za represirane promotorje
  $X^f_{pr^{f-}_{m}}(0) = \left|\{b^{m-}_{ij}\}\right|$
end for

Izvedi časovno evolucijo z vsako matriko $\mathbf{B}_m$ v $\mathscr{B}$
while $t_f < T^f_{\text{MAX}}$ *do*
  Izvedi en korak SSA na hitrem sistemu $X^f(t)$:
    1) izberi j-to reakcijo, ki jo bo sistem simuliral na osnovi enačbe (B.2)
    2) izračunaj τ_f na osnovi enačbe (B.1)
  $[j, \tau_f]$ = SSA_korak($X^f(t_f)$)
  Preberi indeks m promotorja iz j-te reakcije.
  m = getPromoterIndex(j)

Izvedi spremembe v matrikah $\mathbf{B}_m(t_f)$
if (j = "indeks reakcije vezave aktivatorja na promotor") *then*
Izberi naključno prazno mesto b^m v $\mathbf{B}_m(t_f)$ in zamenjaj njegovo vrednost na 1 (po enačbi (B.5), b^m postane b^{m+})
$b^m = 1$
else if (j = "indeks reakcije disociacije aktivatorja iz promotor") *then*
Izberi naključno mesto b^{m+} v $\mathbf{B}_m(t_f)$ in zamenjaj njegovo vrednost na 0
$b^{m+} = 0$
else if (j = "indeks reakcije vezave represorja na promotor") *then*
Izberi naključno prazno mesto b^m v $\mathbf{B}_m(t_f)$ in zamenjaj njegovo vrednost na -1
$b^m = -1$
else if (j = "indeks reakcije disociacije represorja iz promotor") *then*
Izberi naključno mesto b^{m-} v $\mathbf{B}_m(t_f)$ in zamenjaj njegovo vrednost na 0
$b^{m-} = 0$
else
Zahtevana ni nobena sprememba v $\mathbf{B}_m(t_f)$ pri j-ti reakciji
end if

Spremeni koncentracije kemijskih zvrsti v sistemu $X^f(t_f)$ glede na vrednosti stehiometričnega vektorja ν_j^f
$X^f(t_f + \tau_f) = X^f(t_f) + \nu_j^f$
Povečaj časovni korak
$t_f = t_f + \tau_f$
end while
konec

Po izvedbi vgnezdenega SSA lahko v zunanjem (glavnem) SSA določimo stanje posameznega promotorja v sistemu z upoštevanjem določenega pravila aktivacije. Na ta način se izognemo obravnavi $(k+1)^n$ različnih stanj promotorjev. Algoritem, ki opredeljuje delovanje obeh instanc SSA, smo poimenovali *dinamični večnivojski stohastični simulacijski algoritem* – DMSSA (angl. *dynamic multi-scale stochastic simulation algorithm*). Algoritem je orisan v Alg. B.3.

Algoritem B.3

Dinamični večnivojski stohastični simulacijski algoritem.

Vhodni podatki:
- sistem kemijskih reakcij Ω, ki predstavljajo GRO z več DNA vezavnimi mesti za transkripcijske faktorje
- $\mathscr{K}$ (množica aktivacijskih pravil za vse promotorje v GRO)

Izhodni podatki:
- evolucija sistema koncentracij kemijskih zvrsti X

procedura DynamicMultiscaleSSA (Ω, $\mathscr{K}$)

Definiraj vse matrike $\mathbf{B}$ iz množice $\mathscr{B}$

$$\mathscr{B} = \left\{\mathbf{B}_1, \mathbf{B}_2, \ldots \mathbf{B}_{M_p}\right\}$$

Loči hitre in počasne reakcije, ter kemijske zvrsti, ki nastopajo v njih

$$X^f = \left\{X_{\mathscr{P}^f}, X_{\mathscr{T}^f}\right\}$$

$$X^s = \left\{X_{\mathscr{P}^s}, X_{\mathscr{T}^s}, X_{\mathscr{Y}^s}\right\}$$

$$X = \left\{X^f, X^s\right\}$$

Postavi začetni čas na 0

$$t_s = 0$$

while $t_s < T^s_{\text{MAX}}$ *do*

Osveži koncentracije skupnih kemijskih zvrsti glede na izvedbo počasnih kemijskih reakcij

$$X^f_{\mathscr{T}_{s\cap f}} = X^s_{\mathscr{T}_{s\cap f}}$$

Izvedi časovno evolucijo matrik vezavnih mest oz. na sistemu $X^f(t)$

$$X^f(t_s + T^f_{\text{MAX}}) = bseSSA\left(X^f(t_s), \mathscr{B}\right)$$

Izračunaj stanje promotorjev, na osnovi posameznih funkcij pravil h iz množice pravil $\mathscr{K}$ in shrani njihovo stanje (+1 = aktivno, -1 = represirano) v matrike $\mathbf{A}_m$. Število pojavitev vrednosti +1 v matrikah $\mathbf{A}_m$ določa koncentracijo aktivnih promotorjev v času t.

for all $(m = 1, \ldots, M_p)$ *do*

$$\mathbf{A}_m(t_s) = h_m\left(\mathbf{B}_m(t_s)\right)$$

$$pr^{s+}_m = f_1\left(\mathbf{A}_m(t_s)\right)$$

end for

Osveži koncentracije skupnih kemijskih zvrsti glede na izvedbo hitrih reakcij.

$$X^s_{\mathscr{T}_{s\cap f}} = X^f_{\mathscr{T}_{s\cap f}}$$

Simuliraj en korak SSA algoritma za počasne reakcije, tj. izračunaj τ in j na osnovi enačb (B.1) in (B.2)

$$\left[j, \tau_s\right] = SSA_korak\left(X^s(t_s)\right)$$

Spremeni stanje sistema in povečaj časovni korak t_s
$X^s(t_s + \tau_s) = X^s(t_s) + v_j^s$
$t_s = t_s + \tau_s$
end while
konec

Občutljivostna analiza V disertaciji smo se osredotočili na ocenjevanje občutljivosti stohastičnih modelov GRO, ki vsebujejo več nekooperativnih DNA vezavnih mest za transkripcijske faktorje. Potrebno je torej oceniti občutljivost večnivojskih stohastičnih modelov na osnovi algoritma Alg. B.3. Predlagali smo izboljšano različico Morrisove metode [197], ki omogoča ocenjevanje časovno neodvisnih elementarnih učinkov posameznih parametrov na odziv modela.

V Morrisovi metodi je *elementarni učinek* (angl. *elementary effect*) i-tega parametra v modelu z odzivno funkcijo y definiran kot

$$d_i(\mathbf{x}) = \frac{y(x_1, \dots, x_{i-1}, x_i + \Delta, x_{i+1}, \dots, x_k) - y(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_k)}{\Delta}, \quad \text{(B.6)}$$

kjer Δ označuje velikost perturbacije parametra z indeksom i. Občutljivost modela na i-ti parameter lahko izračunamo kot povprečje elementarnih učinkov:

$$\mu_i = \frac{1}{r} \sum_{j=1}^{r} d_i(\mathbf{x}_j),$$

ki jih izračunamo za r različnih vzorčnih vektorjev parametrov $\mathbf{x}$. Saltelli in ostali predlagajo uporabo povprečja absolutne vrednosti elementarnih učinkov

$$\mu_i^* = \frac{1}{r} \sum_{j=1}^{r} \left| d_i(\mathbf{x}_j) \right|, \quad \text{(B.7)}$$

zaradi možnosti pojavitve negativnih vrednosti v enačbi (B.6) [196]. Povezanost vpliva i-tega parametra z ostalimi parametri lahko izračunamo na osnovi standardne deviacije r elementarnih učinkov, tj.

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{r} \left(d_i(\mathbf{x}_l) - \mu_i \right)^2}{r}}. \quad \text{(B.8)}$$

Vrednost μ_i (ali μ_i^*) se uporablja za sortiranje parametrov glede na njihov vpliv na odzivno (izhodno) funkcijo modela. Večja kot je vrednost μ_i (ali μ_i^*), večji je vpliv parametra na izhod sistema. V tem primeru pravimo, da je sistem občutljiv na perturbacije parametra z indeksom i. Majhna vrednost standardnega odklona σ_i označuje podobnost med r-timi elementarnimi efekti, kar pomeni, da je učinek i-tega parametra na sistem neodvisen od vrednosti ostalih parametrov. Po drugi strani velika vrednost σ_i pomeni, da so elementarni učinki odvisni od vrednosti, ki jih zavzamejo ostali parametri.

V primeru, da je odzivna funkcija y časovno odvisna, kot pri odzivih modelov pridobljenih s simulacijskim algoritmom DMSSA, postane tudi elementarni učinek časovno odvisen, tj. $d_i(\mathbf{x}, t)$. Za namen izogibanja časovne odvisnosti, smo v disertaciji predlagali alternativne izhodne funkcije y. Med temi je pričakovana vrednost

$$\mathbb{E}\left(y[n] \right) = \frac{1}{N} \sum_{n=0}^{N-1} y[n] = \overline{y}[n] = \overline{y}, \tag{B.9}$$

kjer $y[n]$ označuje časovno diskretni signal izhoda sistema dolžine N. Pričakovana vrednost je v tem primeru časovno povprečje vseh vrednosti izbranega izhoda. Alternativna izhodna funkcija predstavlja t.i. *vezavni gradient* (angl. *binding gradient*) i-te kopije m-tega promotorja. Ta je definiran z izrazom:

$$g_{m,i}(t) = \sum_{j=1}^{n_m} b_{i,j}^m(t), \tag{B.10}$$

kjer je $b_{i,j}^m(t)$ vrednost vezavnega mesta v matriki $\mathbf{B}_m$ v i-ti vrstici in j-tem stolpcu. Z n_m označujemo število vezavnih mest za transkripcijske faktorje na m-tem promotorju GRO. Če seštejemo vse vezavne gradiente za vse kopije promotorja v celici, dobimo izraz

$$g_m(t) = \sum_{i=1}^{c_m} g_{m,i}(t) = \sum_{i=1}^{c_m} \sum_{j=1}^{n_m} b_{i,j}^m(t). \tag{B.11}$$

Pri stohastičnih simulacijah je običajno potrebno izvajati veliko število instanc iste simulacije z istim naborom parametrom za pridobitev statistično reprezentativnih rezultatov. To je potrebno tudi pri računanju vrednosti $y[n]$ in $g_m(t)$, če so le-te pridobljene na osnovi stohastičnih simulacij. Pričakovano vrednost za vse ponovitve simulacij lah-

ko predstavimo z matriko

$$\mathbb{E}(\mathbf{Y}) = \begin{bmatrix} \overline{y}_1[n_1] \\ \overline{y}_2[n_2] \\ \vdots \\ \overline{y}_M[n_M] \end{bmatrix} = \overline{\mathbf{Y}}. \tag{B.12}$$

Pričakovana vrednost matrike $\overline{\mathbf{Y}}$ je skalar $\mathbb{E}\left(\overline{\mathbf{Y}}\right)$, ki ga izračunamo kot:

$$s_y(\mathbf{x}) = \mathbb{E}\left(\overline{\mathbf{Y}}\right) = \frac{1}{M}\sum_{i=1}^{M} \overline{y}_i[n_i]. \tag{B.13}$$

Vrednost $s_y(\mathbf{x})$ predstavlja časovno neodvisno statistiko, ki nam pove, kakšno je pričakovano povprečje izhodnih odzivov sistema skozi več ponovitev simulacije. Tovrstno statistiko lahko uporabljamo neposredno za izračun elementarnih učinkov:

$$d_i^{s_y}(\mathbf{x}) = \frac{s_y(\mathbf{x} + \Delta_i) - s_y(\mathbf{x})}{\Delta}. \tag{B.14}$$

Tu z notacijo $\mathbf{x} + \Delta_i$ označujemo $[x_1, \ldots, x_{i-1}, x_i + \Delta, x_{i+1}, \ldots, x_k]$. Z elementarnimi učinki $d_i^{s_y}(\mathbf{x})$ pridobljenimi na osnovi nove odzivne funkcije s_y, lahko na koncu izračunamo še občutljivostni metriki μ_i^* in σ_i po izrazik

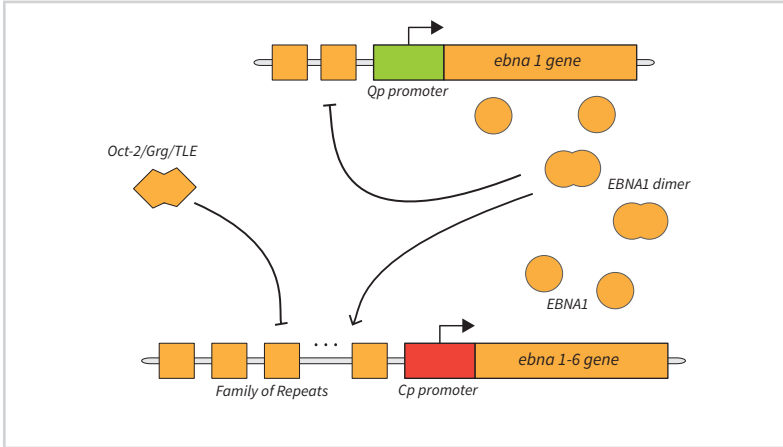
$$\begin{aligned} \mu_i^*\left(s_y\right) &= \frac{1}{r}\sum_{l=1}^{r}\left|d_i^{s_y}(\mathbf{x}_l)\right|, \\ \sigma_i\left(s_y\right) &= \sqrt{\frac{\sum_{l=1}^{r}\left(d_i^{s_y}(\mathbf{x}_l) - \mu_i\right)^2}{r}}, \end{aligned} \tag{B.15}$$

kjer je

$$\mu_i\left(s_y\right) = \frac{1}{r}\sum_{l=1}^{r} d_i^{s_y}(\mathbf{x}_l).$$

Podobno, lahko ocenimo elementarne učinke in občutljivosti parametrov na osnovi vezavnega gradienta $g[n]$. Tudi pri vezavnih gradientih se lahko izognemo časovni odvisnosti z enostavnim povprečenjem pridobljenih izhodnih vrednosti, tj. na podlagi vrednosti $s_g(\mathbf{x})$. Tako s_y kot tudi s_g lahko uporabljamo torej ne le za natančno sta-

tistično predstavitev odziva modela, ampak tudi za ocenjevanje občutljivosti modela na posamezen parameter. Za pospešitev izračuna povprečnih odzivov sistema smo v sklopu disertacije razvili paralelno simulacijsko orodje *ParMSSA*, ki je zmožno paralelnega izvajanja velikega števila stohastičnih simulacij za potrebe pridobivanja izhodnih vrednosti.



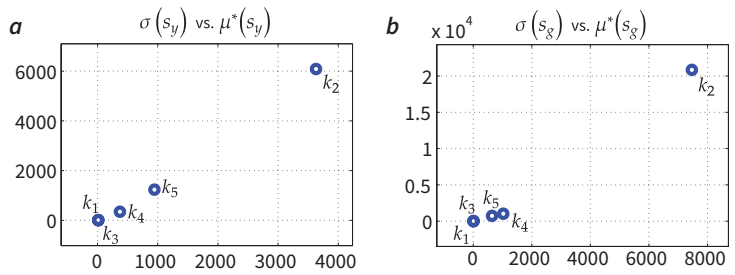
Slika B.2

Shema genskega preklopnega stikala virusa Epstein-Barr [191].

Rezultati Metodo za ocenjevanje občutljivosti modelov smo uporabili na dveh realnih primerih: na modelu GRO preklopa virusa Epstein-Barr, in na modelu sintetičnega oscilatorja. V prvem primeru smo občutljivostno analizo uporabili za identifikacijo parametrov z največjim vplivom na izhode modela. Pri sintetičnem oscilatorju smo občutljivostno analizo uporabili tako za razvrščanje vhodnih parametrov modela glede na njihov vpliv na izhode modela, kot tudi za ocenitev robustnosti modela na osnovi perturbiranja najbolj občutljivih parametrov.

Model preklopnega stikala v virusu Epstein-Barr je prikazan na sliki B.2. Model vsebuje negativno in pozitivno povratno zanko pri regulaciji dveh promotorjev, in sicer promotorja Cp in Qp. Oba promotorja regulira več nekooperativnih DNA vezavnih mest. Promotor Cp je reguliran z 20 vezavnimi mesti, promotor Qp pa z dvema vezavnima mestoma. Dinamiko modela smo simulirali s pomočjo algoritma DMSSA. Dobljene rezultate smo validirali na že obstoječih podatkih iz literature [191]. Naknadno smo izvedeli občutljivostno analizo z uporabo izboljšane Morrisove metode.

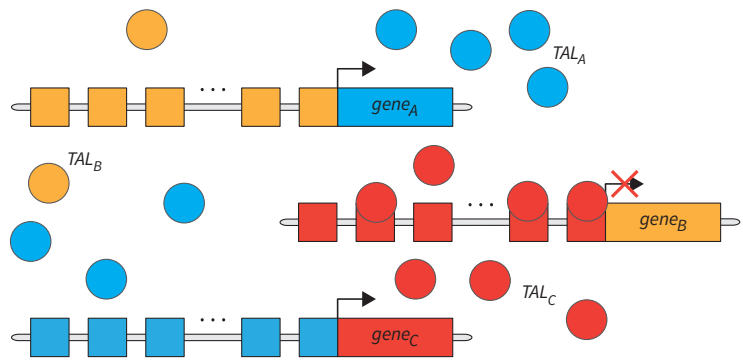
Rezultati analize so prikazani na sliki B.3.



Standardna deviacija proti absolutno povprečje elementarnih učinkov: (a) časovnega povprečja od odziva EBNA1 (s_y), in (b) časovnega povprečja od vezavnega gradients Cp promotorja. k1 predstvlja dissociacijsko konstanto K_{dE} za dimerizacijo EBNA1, k2 in k3 predstavljata razpolovni čas EBNA1 dimera in EBNA1 proteina. k4 je Oct2+Grg/TLE razpolovna doba in k5 označuje število kopij Cp promotorja.

Slika B.3

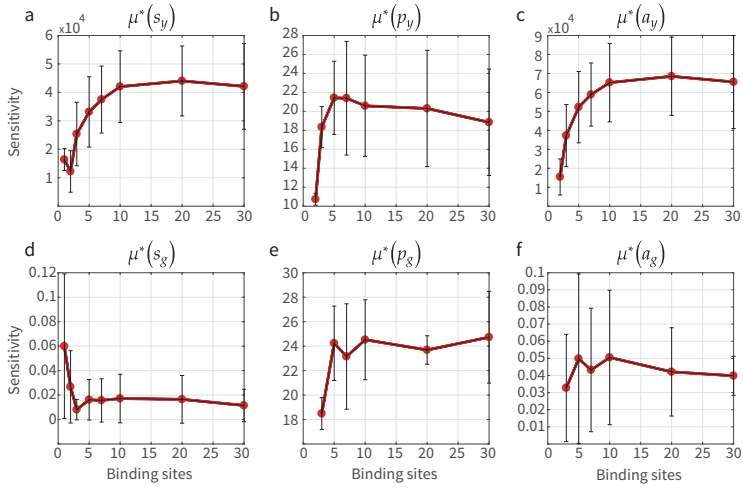
Občutljivosti parametrov pridobljene glede na odziv transkripcijskega faktorja EBNA1.



Slika B.4

Shema sintetičnega genetskega oscilatorja na osnovi represilatorja [34]. Vezje je sestavljeno iz treh sintetičnih genov, ki so regulirani z več DNA vezavnimi mesti ter s tremi sintetičnimi transkripcijskimi faktorji: TAL_A, TAL_B in TAL_C.

Sintetični genetski oscilator predstavlja osrčje sintezne biologije. Shema oscilatorja, ki smo ga analizirali, je prikazana na sliki B.4. V njem se sintetični proteini, ki igrajo vlogo represorjev, kompetitivno vežejo na prosta vezavna mesta in tako regulirajo transkripcijo ostalih represorjev.



Slike prikazujejo občutljivost sistema na parameter, ki določa razpolovno dobo oziroma hitrost degradacije prostega proteina. Slika (a) prikazuje občutljivost koncentracije enega izmed represorjev, slika (b) prikazuje občutljivost povprečne periode oscilacij, slika (c) prikazuje občutljivost amplitude glavne frekvence odziva, slika (d) prikazuje občutljivost povprečnega odziva vezavnega gradienta enega izmed promotorjev, slika (e) prikazuje občutljivost periode glavne frekvence odziva vezavnega gradienta in slika (f) prikazuje občutljivost amplitude glavne frekvence odziva vezavnega gradienta. Vse občutljivosti so prikazane s 95 % intervalom zaupnosti in se nanašajo na singularno pravilo.

Slika B.5

Občutljivost sistema v odvisnosti od števila vezavnih mest.

Predlagani oscilator smo simulirali za različne čase in parametre z namenom ocene robustnosti oscilatorja in njene odvisnosti od števila vezavnih mest. Najprej smo izvedli občutljivostno analizo za identifikacijo najvplivnejšega parametra, za katerega se je izkazala hitrost degradacije prostega proteina. V nadaljevanju smo analizirali kako število vezavnih mest vpliva na občutljivost sistema glede na ta parameter. Rezultati analize so pokazali, da število vezavnih mest robustnosti oscilacij ne povečuje (glej sliko B.5), poveča pa zmožnost natančne kalibracije amplitude in periode oscilacij.

Zaključek V disertaciji smo razvili metodo za stohastično modeliranje in simulacijo gensko regulatornih omrežij, ki vsebujejo več nekooperativnih DNA vezavnih mest za transkripcijske faktorje. Razvili smo algoritem DMSSA za večnivojsko stohastično simuliranje dinamike tovrstnih gensko regulatornih omrežij.

Izboljšali smo Morrisovo metodo za analizo občutljivosti parametrov stohastičnih modelov na osnovi uporabe algoritma DMSSA in orodja *ParMSSA* za pohitritev simulacij. Analizo občutljivosti smo testirali na dveh vzorčnih modelih s področja sistemske in sintezne biologije, tj. na modelu preklopnega stikala virusa Epstein-Barr in na sintetičnem genetskem oscilatorju. Rezultati so potrdili ustreznost metode in na koncu pokazali kako se občutljivost genetskega oscilatorja z večanjem števila DNA vezavnih mest za transkripcijske faktorje spremeni. Tovrstno strategijo lahko tako uporabimo pri načrtovanju robustnih sintetičnih bioloških sistemov z zmožnostjo procesiranja informacij.

BIBLIOGRAPHY

[1] M. Mitchell Waldrop. The chips are down for Moore's law. *Nature News*, 530(7589):144, 2016. doi: 10.1038/530144a.

[2] James D. Meindl, Qiang Chen, and Jeffrey A. Davis. Limits on Silicon Nanoelectronics for Terascale Integration. *Science*, 293(5537):2044–2049, 2001. doi: 10.1126/science.293.5537.2044.

[3] Baojun Wang, Richard I. Kitney, Nicolas Joly, and Martin Buck. Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology. *Nature communications*, 2:508, 2011. doi: 10.1038/ncomms1516.

[4] Rok Gaber, Tina Lebar, Andreja Majerle, Branko Šter, Andrej Dobnikar, Mojca Benčina, and Roman Jerala. Designable DNA-binding domains enable construction of logic circuits in mammalian cells. *Nature Chemical Biology*, 10(3):203–208, 2014. doi: 10.1038/nchembio.1433.

[5] Rok Gaber. *Use of synthetic DNA binding proteins for information processing and process directing in biological systems*. PhD thesis, University of Ljubljana, Faculty of Medicine, 2014.

[6] Tina Lebar, Urban Bezeljak, Anja Golob, Miha Jerala, Lucija Kadunc, Boštjan Pirš, Martin Stražar, Dušan Vučko, Uroš Zupančič, Mojca Benčina, et al. A bistable genetic switch based on designable DNA-binding domains. *Nature communications*, 5, 2014. doi: 10.1038/ncomms6007.

[7] Tina Lebar and Roman Jerala. Benchmarking of TALE-and CRISPR/dCas9-based transcriptional regulators in mammalian cells for the construction of synthetic genetic circuits. *ACS Synthetic Biology*, 5 (10), 2016. doi: 10.1021/acssynbio.5b00259.

[8] Jerome Bonnet, Peter Yin, Monica E. Ortiz, Pakpoom Subsoontorn, and Drew Endy. Amplifying genetic logic gates. *Science*, 340(6132):599–603, 2013. doi: 10.1126/science.1232758.

[9] Tamara Niazov, Ronan Baron, Eugenii Katz, Oleg Lioubashevski, and Itamar Willner. Concatenated logic gates using four coupled biocatalysts operating in series. *Proceedings of the National Academy of Sciences*, 103(46):17160–17163, 2006. doi: 10.1073/pnas.0608319103.

[10] Ernesto Andrianantoandro, Subhayu Basu, David K Karig, and Ron Weiss. Synthetic biology: new engineering rules for an emerging discipline. *Molecular Systems Biology*, 2(1), 2006. ISSN 1744-4292. doi: 10.1038/msb4100073.

[11] George M. Church, Michael B. Elowitz, Christina D. Smolke, Christopher A. Voigt, and Ron Weiss. Realizing the potential of synthetic biology. *Nature Reviews Molecular Cell Biology*, 15(4):289–294, 2014. doi: 10.1038/nrm3767.

[12] D. Ewen Cameron, Caleb J. Bashor, and James J. Collins. A brief history of synthetic biology. *Nature Reviews Microbiology*, 12(5):381–390, 2014. doi: 10.1038/nrmicro3239.

[13] Eileen R. Choffnes, David A. Relman, Leslie Pray, et al. *The Science and Applications of Synthetic and Systems Biology: Workshop Summary*. National Academies Press, 2011. doi: 10.17226/13239.

[14] Robert A. Meyers, editor. *Synthetic Biology: Advances in Molecular Biology and Medicine*. John Wiley & Sons, 2015. ISBN 978-3-527-33482-7.

[15] Ahmad S. Khalil and James J. Collins. Synthetic biology: applications come of age. *Nature Reviews Genetics*, 11(5):367–379, 2010. doi: 10.1038/nrg2775.

[16] Evan Appleton, Jenhan Tao, Traci Haddock, and Douglas Densmore. Interactive assembly algorithms for molecular cloning. *Nature Methods*, 11:657–662, 2014. doi: 10.1038/nmeth.2939.

[17] Daniel G. Gibson. Programming biological operating systems: genome design, assembly and activation. *Nature methods*, 11(5):521–526, 2014. doi: 10.1038/nmeth.2894.

[18] Sriram Kosuri and George M. Church. Large-scale de novo DNA synthesis: technologies and applications. *Nature methods*, 11(5):499–507, 2014. doi: 10.1038/nmeth.2918.

[19] James M. Carothers, Jonathan A. Goler, Darmawi Juminaga, and Jay D. Keasling. Model-driven engineering of RNA devices to quantitatively program gene expression. *Science*, 334(6063):1716–1719, 2011. doi: 10.1126/science.1212209.

[20] Jennifer A. N. Brophy and Christopher A. Voigt. Principles of genetic circuit design. *Nature Methods*, 11(5): 508–520, May 2014. doi: 10.1038/nmeth.2926.

[21] Wilfried Weber and Martin Fussenegger. Emerging biomedical applications of synthetic biology. *Nature reviews. Genetics*, 13(1):21–35, 2012. doi: 10.1038/nrg3094.

[22] Markus Schmidt, editor. *Synthetic biology: industrial and environmental applications*. John Wiley & Sons, 2012. ISBN 978-3-527-33183-3.

[23] Richard Kitney and Paul Freemont. Synthetic biology – the state of play. *FEBS Letters*, 586(15):2029–2036, 2012. doi: 10.1016/j.febslet.2012.06.002.

[24] Alvin Tamsir, Jeffrey J Tabor, and Christopher A Voigt. Robust multicellular computing using genetically encoded nor gates and chemical 'wires'. *Nature*, 469(7329):212–215, 2011. doi: 10.1038/nature09565.

[25] Simon Ausländer, David Ausländer, Marius Müller, Markus Wieland, and Martin Fussenegger. Programmable single-cell mammalian biocomputers. *Nature*, 487(7405):123–127, 2012. doi: 10.1038/nature11149.

[26] Florian Lienert, Joseph P. Torella, Jan-Hung Chen, Michael Norsworthy, Ryan R. Richardson, and Pamela A. Silver. Two- and three-input tale-based and logic computation in embryonic stem cells. *Nucleic Acids Research*, 41(21):9967, 2013. doi: 10.1093/nar/gkt758.

[27] Alexander Hunziker, Csaba Tuboly, Péter Horváth, Sandeep Krishna, and Szabolcs Semsey. Genetic flexibility of regulatory networks. *Proceedings of the National Academy of Sciences*, 107(29):12998–13003, 2010. doi: 10.1073/pnas.0915003107.

[28] Robert W. Bradley, Martin Buck, and Baojun Wang. Recognizing and engineering digital-like logic gates and switches in gene regulatory networks. *Current Opinion in Microbiology*, 33(10):74–82, 2016. doi: 10.1016/j.mib.2016.07.004.

[29] Timothy S. Gardner, Charles R. Cantor, and James J. Collins. Construction of a genetic toggle switch in Escherichia coli. *Nature*, 403(6767):339–342, 2000. doi: 10.1038/35002131.

[30] Mariette R. Atkinson, Michael A. Savageau, Jesse T. Myers, and Alexander J. Ninfa. Development of genetic circuitry exhibiting toggle switch or oscillatory behavior in Escherichia coli. *Cell*, 113(5):597–607, 2003. doi: 10.1016/s0092-8674(03)00346-5.

[31] Samir M. Hamdan, Boriana Marintcheva, Timothy Cook, Seung-Joo Lee, Stanley Tabor, and Charles C. Richardson. A unique loop in T7 DNA polymerase mediates the binding of helicase-primase, DNA binding protein, and processivity factor. *Proceedings of the National Academy of Sciences of the United States of America*, 102(14):5096–5101, 2005. doi: 10.1073/pnas.0501637102.

[32] Jacob R. Rubens, Gianluca Selvaggio, and Timothy K. Lu. Synthetic mixed-signal computation in living cells. *Nature communications*, 7, 2016. doi: 10.1038/ncomms11658.

[33] Subhayu Basu, Yoram Gerchman, Cynthia H. Collins, Frances H. Arnold, and Ron Weiss. A synthetic multicellular system for programmed pattern formation. *Nature*, 434(7037):1130–1134, 2005. doi: 10.1038/nature03461.

[34] Michael B. Elowitz and Stanislas Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000. doi: 10.1038/35002125.

[35] Marcel Tigges, Nicolas Dénervaud, David Greber, Joerg Stelling, and Martin Fussenegger. A synthetic low-frequency mammalian oscillator. *Nucleic acids research*, 38(8):2702–2711, 2010. doi: 10.1093/nar/gkq121.

[36] Ari E. Friedland, Timothy K. Lu, Xiao Wang, David Shi, George Church, and James J. Collins. Synthetic Gene Networks That Count. *Science*, 324(5931):1199–1202, 2009. ISSN 0036-8075. doi: 10.1126/science.1172005.

[37] Nick Goldman, Paul Bertone, Siyuan Chen, Christophe Dessimoz, Emily M Leproust, Botond Sipos, and Ewan Birney. Information storage in synthesized DNA. *Nature*, 494(7435):77–80, 2013. ISSN 0028-0836. doi: 10.1038/nature11875.

[38] George M. Church, Yuan Gao, and Sriram Kosuri. Next-Generation Digital Information Storage in DNA. *Science*, 337(6102):1628–1628, 2012. doi: 10.1126/science.1226355.

[39] Caroline M. Ajo-Franklin, David A. Drubin, Julian A. Eskin, Elaine P.S. Gee, Dirk Landgraf, Ira Phillips, and Pamela A. Silver. Rational design of memory in eukaryotic cells. *Genes & Development*, 21(18): 2271–2276, 2007. doi: 10.1101/gad.1586107.

[40] Piro Siuti, John Yazbek, and Timothy K Lu. Synthetic circuits integrating logic and memory in living cells. *Nature biotechnology*, 31(5):448–452, 2013. doi: 10.1038/nbt.2510.

[41] Roberta Kwok. Five hard truths for synthetic biology. *Nature*, 463(7279):288–291, 2010. doi: 10.1038/463288a.

[42] Jens Boch, Heidi Scholze, Sebastian Schornack, Angelika Landgraf, Simone Hahn, Sabine Kay, Thomas Lahaye, Anja Nickstadt, and Ulla Bonas. Breaking the Code of DNA Binding Specificity of TAL-Type III Effectors. *Science*, 326(5959):1509–1512, 2009. doi: 10.1126/science.1178811.

[43] J Keith Joung and Jeffry D Sander. TALENs: a widely applicable technology for targeted genome editing. *Nature reviews Molecular cell biology*, 14(1): 49–55, 2013. doi: 10.1038/nrm3486.

[44] Rajat M. Gupta and Kiran Musunuru. Expanding the genetic editing tool kit: ZFNs, TALENs, and CRISPR-Cas9. *The Journal of Clinical Investigation*, 124:4154–4161, 10 2014. doi: 10.1172/JCI72992.

[45] John Van der Oost, Matthijs M. Jore, Edze R. Westra, Magnus Lundgren, and Stan J.J. Brouns. CRISPR-based adaptive and heritable immunity in prokaryotes. *Trends in biochemical sciences*, 34(8):401–407, 2009. doi: 10.1016/j.tibs.2009.05.002.

[46] David Bikard, Wenyan Jiang, Poulami Samai, Ann Hochschild, Feng Zhang, and Luciano A Marraffini. Programmable repression and activation of bacterial gene expression using an engineered crispr-cas system. *Nucleic acids research*, 41(15):7429–7437, 2013. doi: 10.1093/nar/gkt520.

[47] Thomas Gaj, Charles A Gersbach, and Carlos F Barbas. ZFN, TALEN, and CRISPR/Cas-based methods for genome engineering. *Trends in biotechnology*, 31(7):397–405, 2013. doi: 10.1016/j.tibtech.2013.04.004.

[48] Le Cong, F. Ann Ran, David Cox, Shuailiang Lin, Robert Barretto, Naomi Habib, Patrick D. Hsu, Xuebing Wu, Wenyan Jiang, Luciano A. Marraffini, and Feng Zhang. Multiplex Genome Engineering Using CRISPR/Cas Systems. *Science*, 339(6121):819–823, 2013. doi: 10.1126/science.1231143.

[49] Prashant Mali, Luhan Yang, Kevin M. Esvelt, John Aach, Marc Guell, James E. DiCarlo, Julie E. Norville, and George M. Church. RNA-Guided Human Genome Engineering via Cas9. *Science*, 339(6121): 823–826, 2013. doi: 10.1126/science.1232033.

[50] Lei S. Qi, Matthew H. Larson, Luke A. Gilbert, Jennifer A. Doudna, Jonathan S. Weissman, Adam P. Arkin, and Wendell A. Lim. Repurposing CRISPR as an RNA-Guided Platform for Sequence-Specific Control of Gene Expression. *Cell*, 152(5):1173–1183, 2013. doi: 10.1016/j.cell.2013.02.022.

[51] Martin Jinek, Alexandra East, Aaron Cheng, Steven Lin, Enbo Ma, and Jennifer Doudna. RNA-programmed genome editing in human cells. *eLife*, 2 (e00471), 2013. doi: 10.7554/eLife.00471.

[52] Silvana Konermann, Mark D Brigham, Alexandro E Trevino, Julia Joung, Omar O Abudayyeh, Clea Barcena, Patrick D Hsu, Naomi Habib, Jonathan S Gootenberg, Hiroshi Nishimasu, et al. Genome-scale transcriptional activation by an engineered CRISPR-Cas9 complex. *Nature*, 2014. doi: 10.1038/nature14136.

[53] Alec A. K. Nielsen and Christopher A. Voigt. Multi-input CRISPR/Cas genetic circuits that interface host regulatory networks. *Molecular systems biology*, 10(11): 763, 2014. doi: 10.15252/msb.20145735.

[54] Lior Nissim, Samuel D. Perli, Alexandra Fridkin, Pablo Perez-Pinera, and Timothy K. Lu. Multiplexed and Programmable Regulation of Gene Networks with an Integrated RNA and CRISPR/Cas Toolkit in Human Cells. *Molecular Cell*, 54(4):698–710, 2014. doi: 10.1016/j.molcel.2014.04.022.

[55] Luisa Bortesi and Rainer Fischer. The crispr/-cas9 system for plant genome editing and beyond. *Biotechnology advances*, 33(1):41–52, 2015. doi: 10.1016/j.biotechadv.2014.12.006.

[56] Antonia A Dominguez, Wendell A Lim, and Lei S Qi. Beyond editing: repurposing CRISPR-Cas9 for precision genome regulation and interrogation. *Nature Reviews Molecular Cell Biology*, 17(1):5–15, 2016. ISSN 1471-0072. doi: 10.1038/nrm.2015.2.

[57] Devashish Rath, Lina Amlinger, Archana Rath, and Magnus Lundgren. The CRISPR-Cas immune system: Biology, mechanisms and applications. *Biochimie*, 117:119–128, 2015. doi: 10.1016/j.biochi.2015.03.025.

[58] Miles Winston Gander, Justin D Vrana, William E. Voje, James M Carothers, and Eric Kalvins. Robust digital logic circuits in eukaryotic cells with CRISPR/dCas9 NOR gates. *bioRxiv*, 2016. doi: 10.1101/041871.

[59] Timothy K. Lu, Ahmad S. Khalil, and James J. Collins. Next-generation synthetic gene networks. *Nature Biotechnology*, 27(12):1139–1150, 2009. doi: 10.1038/nbt.1591.

[60] Benjamin A. Blount, Tim Weenink, and Tom Ellis. Construction of synthetic regulatory networks in yeast. *FEBS Letters*, 586(15):2112–2121, 2012. doi: 10.1016/j.febslet.2012.01.053.

[61] Akiyoshi Higo, Atsuko Isu, Yuki Fukaya, and Toru Hisabori. Designing Synthetic Flexible Gene Regulation Networks Using RNA Devices in Cyanobacteria. *ACS Synthetic Biology*, In press, 2016. doi: 10.1021/acssynbio.6b00201.

[62] Maria Karlsson, Wilfried Weber, and Martin Fussenegger. Design and construction of synthetic gene networks in mammalian cells. In Wilfried Weber and Martin Fussenegger, editors, *Synthetic Gene Networks: Methods and Protocols*, pages 359–376. Humana Press, Totowa, NJ, 2012. ISBN 978-1-61779-412-4. doi: 10.1007/978-1-61779-412-4_22.

[63] Yolanda Schaerli and Mark Isalan. Building synthetic gene circuits from combinatorial libraries: screening and selection strategies. *Molecular BioSystems*, 9: 1559–1567, 2013. doi: 10.1039/C2MB25483B.

[64] Alexander J. Hartemink. Reverse engineering gene regulatory networks. *Nature Biotechnology*, 23(5):554–555, May 2005. ISSN 087-0156. doi: 10.1038/nbt0505-554.

[65] Ramesh Ram and Madhu Chetty. Generating synthetic gene regulatory networks. In Madhu Chetty, Alioune Ngom, and Shandar Ahmad, editors, *Pattern Recognition in Bioinformatics: Third IAPR International Conference, PRIB 2008, Melbourne, Australia, October 15-17, 2008. Proceedings*, pages 237–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-88436-1. doi: 10.1007/978-3-540-88436-1_21.

[66] Hidde de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of Computational Biology*, 9(1):67–103, 2002. doi: 10.1089/10665270252833208.

[67] Miha Moškon. *Computer structures perspective on switching dynamics of simple biological systems.* PhD thesis, University of Ljubljana, Faculty of Computer and Information Science, 2012. URL http://eprints.fri.uni-lj.si/1804/.

[68] Miha Moškon, Jure Bordon, Miha Mraz, Nikolaj Zimic, and Mattia Petroni. Computational approaches in synthetic and systems biology. In Andrè X. C. N Valente, Abhijit Sarkar, and Yuan Gao, editors, *Recent Advances in Systems Biology Research*. Nova Science Publishers Inc., Hauppauge NY, USA, 2013. ISBN 978-1-62948-736-6.

[69] Hiroaki Kitano. Biological Robustness. *Nature Reviews Genetics*, 5(11):826–837, 2004. doi: 10.1038/nrg1471.

[70] Hiroaki Kitano. Towards a theory of biological robustness. *Molecular Cell Biology*, 3(137):1–7, 2007. doi: 10.1038/msb4100179.

[71] Uri Alon. *An Introduction to Systems Biology*. Chapman & Hall/CRC Mathematical and Computational Biology Series. Chapman & Hall/CRC, Taylor & Francis Group, 6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL 33487-2742, USA, 2007. ISBN 978-1-58488-642-6.

[72] Pontus Melke, Henrik Jonsson, Evangelia Pardali, Peter ten Dijke, and Carsten Peterson. A Rate Equation Approach to Elucidate the Kinetics and Robustness of the TGF-β Pathway. *Biophysical Journal*, 91(12):4368–4380, 2006. doi: 10.1529/biophysj.105.080408.

[73] E W Jacobsen and G Cedersund. Structural robustness of biochemical network models-with application to the oscillatory metabolism of activated neutrophils. *IET Systems Biology*, 2(1):39–47, 2008. doi: 10.1049/iet-syb:20070008.

[74] Mochamad Apri, Jaap Molenaar, Maarten De Gee, and George Van Voorn. Efficient Estimation of the Robustness Region of Biological Models with Oscillatory Behavior. *PLoS ONE*, 5(4), 2010. doi: 10.1371/journal.pone.0009865.

[75] R. Ghaemi and D. Del Vecchio. Evaluating the robustness of a biochemical network model. In *Decision and Control, 2007 46th IEEE Conference on*, pages 615–620, Dec 2007. doi: 10.1109/CDC.2007.4434348.

[76] Jongrae Kim, Declan G. Bates, Ian Postlethwaite, Lan Ma, and Pablo A. Iglesias. Robustness Analysis of Biochemical Networks using μ-Analysis and Hybrid Optimisation. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 6234–6239, 2005. doi: 10.1109/CDC.2005.1583160.

[77] J. Kim, D. G. Bates, I. Postlethwaite, L. Ma, and P. A. Iglesias. Robustness analysis of biochemical network models. *IEE Proceedings Systems Biology*, 153 (3):96–104, 2006. doi: 10.1049/ip-syb:20050024.

[78] Aurélien Rizk, Gregory Batt, François Fages, and Sylvain Soliman. A general computational method for robustness analysis with applications to synthetic gene networks. *Bioinformatics*, 25(12):i169, 2009. doi: 10.1093/bioinformatics/btp200.

[79] Jörg Stelling, Ernst Dieter Gilles, and Francis J. Doyle. Robustness properties of circadian clock architectures. *Proceedings of the National Academy of Sciences*, 101(36):13210–13215, 2004. doi: 10.1073/pnas.0401463101.

[80] Neda Bagheri, Jörg Stelling, and Francis J. Doyle. Quantitative performance metrics for robustness in circadian rhythms. *Bioinformatics*, 23(3):358–364, 2007. doi: 10.1093/bioinformatics/btl627.

[81] N. Barkai and S. Leibler. Robustness in simple biochemical networks. *Nature*, 387(6636):913–917, Jun 1997. ISSN 0028-0836.

[82] Marc Hafner, Heinz Koeppl, Martin Hasler, and Andreas Wagner. Glocal Robustness Analysis and Model Discrimination for Circadian Oscillators. *PLoS Computational Biology*, 5(10), 2009. doi: 10.1371/journal.pcbi.1000534.

[83] T. Sumner, E. Shephard, and I. D. L. Bogle. A methodology for global-sensitivity analysis of time-dependent outputs in systems biology modelling. *Journal of The Royal Society Interface*, 9(74):2156–2166, 2012. doi: 10.1098/rsif.2011.0891.

[84] Zhike Zi. Sensitivity analysis approaches applied to systems biology models. *IET Systems Biology*, 5(6): 336–346, 2011. doi: 10.1049/iet-syb.2011.0015.

[85] D. Anderson. An efficient finite difference method for parameter sensitivities of continuous time markov chains. *SIAM Journal on Numerical Analysis*, 50(5): 2237–2258, 2012. doi: 10.1137/110849079.

[86] Muruhan Rathinam, Patrick W. Sheppard, and Mustafa Khammash. Efficient computation of parameter sensitivities of discrete stochastic chemical reaction networks. *The Journal of Chemical Physics*, 132(3), 2010. doi: 10.1063/1.3280166.

[87] Patrick W. Sheppard, Muruhan Rathinam, and Mustafa Khammash. Spsens: a software package for stochastic parameter sensitivity analysis of biochemical reaction networks. *Bioinformatics*, 29(1):140–142, 2013. doi: 10.1093/bioinformatics/bts642.

[88] Hana El Samad, Mustafa Khammash, Linda Petzold, and Dan Gillespie. Stochastic modelling of gene regulatory networks. *International Journal of Robust and Nonlinear Control*, 15:691–711, 2005. doi: 10.1002/rnc.1018.

[89] Eugenio Cinquemani, Andreas Milias-Argeitis, Sean Summers, and John Lygeros. Stochastic dynamics of genetic networks: modelling and parameter identification. *Bioinformatics*, 24(23):2748–2754, 2008. doi: 10.1093/bioinformatics/btn527.

[90] Darren J. Wilkinson. Stochastic modelling for quantitative description of heterogeneous biological systems. *Nature Reviews Genetics*, 10(2):122–133, 2009. doi: 10.1038/nrg2509.

[91] Adam J. Bogdanove and Daniel F. Voytas. TAL Effectors: Customizable Proteins for DNA Targeting. *Science*, 333(6051):1843–1846, 2011. doi: 10.1126/science.1204094.

[92] Jeffrey C. Miller, Siyuan Tan, Guijuan Qiao, Kyle A. Barlow, Jianbin Wang, Danny F. Xia, Xiangdong Meng, David E. Paschon, Elo Leung, Sarah J. Hinkley, Gladys P. Dulay, Kevin L. Hua, Irina Ankoudinova, Gregory J. Cost, Fyodor D. Urnov, H. Steve Zhang, Michael C. Holmes, Lei Zhang, Philip D. Gregory, and Edward J. Rebar. A TALE nuclease architecture for efficient genome editing. *Nature Biotechnology*, 29(2):143–148, 2011. doi: 10.1038/nbt.1755.

[93] Weinan E, Di Liu, and Eric Vanden-Eijnden. Nested stochastic simulation algorithms for chemical kinetic systems with multiple time scales. *Journal of Computational Physics*, 221(1):158–180, 2007. doi: 10.1016/j.jcp.2006.06.019.

[94] Yang Cao, Daniel T. Gillespie, and Linda R. Petzold. The slow-scale stochastic simulation algorithm. *Journal of Chemical Physics*, 122(1):014116(1–18), 2005. doi: 10.1063/1.1824902.

[95] Yang Cao, Dan Gillespie, and Linda Petzold. Multiscale stochastic simulation algorithm with stochastic partial equilibrium assumption for chemically reacting systems. *Journal of Computational Physics*, 206(2): 395–411, 2005. doi: 10.1016/j.jcp.2004.12.014.

[96] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell*. Garland Science, New York, 4th edition, 2002. ISBN 0-8153-3218-1.

[97] Uri Moran, Rob Phillips, and Ron Milo. SnapShot: Key Numbers in Biology. *Cell*, 141(7):1262–1262.e1, June 2010. doi: 10.1016/j.cell.2010.06.019.

[98] Harvey Lodish, Arnold Berk, S. Lawrence Zipursky, Paul Matsudaira, David Baltimore, and James Darnell. *Molecular Cell Biology*. W. H. Freeman, New York, 5th edition, 2003. ISBN 0-71674366-3.

[99] Gašper Tkačik and Aleksandra M. Walczak. Information transmission in genetic regulatory networks: a review. *Journal of Physics: Condensed Matter*, 23(15):153102, 2011. doi: 10.1088/0953-8984/23/15/153102.

[100] Kevin Chen and Nikolaus Rajewsky. The evolution of gene regulation by transcription factors and micrornas. *Nature Reviews Genetics*, 8(2):93–103, Feb 2007. doi: 10.1038/nrg1990.

[101] Valer Gotea, Axel Visel, John M. Westlund, Marcelo A. Nobrega, Len A. Pennacchio, and Ivan Ovcharenko. Homotypic clusters of transcription factor binding sites are a key component of human promoters and enhancers. *Genome Research*, 20(5): 565–577, 2010. doi: 10.1101/gr.104471.109.

[102] Stanford University. Saccharomyces Genome Database. http://www.yeastgenome.org/, 2016.

[103] J. Almqvist, J. Zou, Y. Linderson, C. Borestrom, E. Altiok, H. Zetterberg, L. Rymo, S. Pettersson, and I. Ernberg. Functional interaction of Oct transcription factors with the family of repeats in Epstein-Barr virus oriP. *Journal of General Virology*, 86(5):1261–1267, 2005. doi: 10.1099/vir.0.80620-0.

[104] N. Ohara, K. Hayashi, N. Teramoto, T. Oka, K. Fujimoto, Y. Yoshikawa, E. Castanos-Velez, P. Biberfeld, and T. Akagi. Sequence Analysis and Variation of EBNA-1 in Epstein-Barr Virus-Related Herpesvirus of Cynomolgus Monkey. *Intervirology*, 43(2):102–106, 2000. doi: 10.1159/000025031.

[105] Luca Giorgetti, Trevor Siggers, Guido Tiana, Greta Caprara, Samuele Notarbartolo, Teresa Corona, Manolis Pasparakis, Paolo Milani, Martha L. Bulyk, and Gioacchino Natoli. Noncooperative Interactions between Transcription Factors and Clustered DNA Binding Sites Enable Graded Transcriptional Responses to Environmental Inputs. *Molecular Cell*, 37(3):418–428, 2010. doi: 10.1016/j.molcel.2010.01.016.

[106] Nicolae Radu Zabet, Robert Foy, and Boris Adryan. The influence of transcription factor competition on the relationship between occupancy and affinity. *PLoS ONE*, 8(9):1–12, 09 2013. doi: 10.1371/journal.pone.0073714.

[107] Nicolae Radu Zabet and Boris Adryan. The effects of transcription factor competition on gene regulation. *Frontiers in Genetics*, 4:197, Oct 2013. doi: 10.3389/fgene.2013.00197.

[108] Iván Lengyel, Daniele Soroldoni, Andrew Oates, and Luis Morelli. Nonlinearity arising from noncooperative transcription factor binding enhances negative feedback and promotes genetic oscillations. *Papers in Physics*, 6, dec 2014. doi: 10.4279/PIP.060012.

[109] Jeff Hasty, David McMillen, and J. J. Collins. Engineered gene circuits. *Nature*, 420(6912):224–230, Nov 2002. doi: 10.1038/nature01257.

[110] Takao Inoue, Minqin Wang, Ted O. Ririe, Jolene S. Fernandes, and Paul W. Sternberg. Transcriptional network underlying Caenorhabditis elegans vulval development. *Proceedings of the National Academy of Sciences of the United States of America*, 102(14):4972–4977, 2005. doi: 10.1073/pnas.0408122102.

[111] Sorin Istrail and Eric H. Davidson. Logic functions of the genomic cis-regulatory code. *Proceedings of the National Academy of Sciences of the United States of America*, 102(14):4954–4959, 2005. doi: 10.1073/pnas.0409624102.

[112] Tetsuya Koide, Tadayoshi Hayata, and Ken WY Cho. Xenopus as a model system to study transcriptional regulatory networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(14):4943–4948, 2005. doi: 10.1073/pnas.0408125102.

[113] Alexander P. Lifanov, Vsevolod J. Makeev, Anna G. Nazina, and Dmitri A. Papatsenko. Homotypic regulatory clusters in drosophila. *Genome Research*, 13(4):579–588, 2003. doi: 10.1101/gr.668403.

[114] Amanda Ochoa-Espinosa, Gozde Yucel, Leah Kaplan, Adam Pare, Noel Pura, Adam Oberstein, Dmitri Papatsenko, and Stephen Small. The role of binding site cluster strength in bicoid-dependent patterning in drosophila. *Proceedings of the National Academy of Sciences of the United States of America*, 102(14):4960–4965, 2005. doi: 10.1073/pnas.0500373102.

[115] Steven A. Benner and A. Michael Sismour. Synthetic biology. *Nature Reviews Genetics*, 6(7):533–543, Jul 2005. doi: 10.1038/nrg1637.

[116] Abhishek Garg, Jason J. Lohmueller, Pamela A. Silver, and Thomas Z. Armel. Engineering synthetic tal effectors with orthogonal target sites. *Nucleic Acids Research*, 40(15):7584, 2012. doi: 10.1093/nar/gks404.

[117] Pablo Perez-Pinera, David G. Ousterout, Jonathan M. Brunger, Alicia M. Farin, Katherine A. Glass, Farshid Guilak, Gregory E. Crawford, Alexander J. Hartemink, and Charles A. Gersbach. Synergistic and tunable human gene activation by combinations of synthetic transcription factors. *Nature Methods*, 10(3):239–242, Mar 2013. doi: 10.1038/nmeth.2361.

[118] Annekatrin Richter and Jens Boch. Designer tales team up for highly efficient gene induction. *Nature Methods*, 10(3):207–208, Mar 2013. doi: 10.1038/nmeth.2373.

[119] Morgan L. Maeder, Samantha J. Linder, Deepak Reyon, James F. Angstman, Yanfang Fu, Jeffry D. Sander, and J. Keith Joung. Robust, synergistic regulation of human gene expression using tale activators. *Nature Methods*, 10(3):243–245, Mar 2013. doi: 10.1038/nmeth.2366.

[120] Benjamin A. Blount, Tim Weenink, Serge Vasylechko, and Tom Ellis. Rational Diversification of a Promoter Providing Fine-Tuned Expression and Orthogonal Regulation for Synthetic Biology. *PLoS ONE*, 7(3):1–11, 03 2012. doi: 10.1371/journal.pone.0033279.

[121] Takashi Sera. Zinc-finger-based artificial transcription factors and their applications. *Advanced Drug Delivery Reviews*, 61(7-8):513 – 526, 2009. doi: 10.1016/j.addr.2009.03.012.

[122] Natalia J. Martinez and Albertha J. M. Walhout. The interplay between transcription factors and microrna in genome-scale regulatory networks. *Bioessays*, 31(4):435–445, Apr 2009. doi: 10.1002/bies.200800212.

[123] Petr V. Nazarov, Susanne E. Reinsbach, Arnaud Muller, Nathalie Nicot, Demetra Philippidou, Laurent Vallar, and Stephanie Kreis. Interplay of micrornas, transcription factors and target genes: linking dynamic expression changes to function. *Nucleic Acids Research*, 41(5):2817–2831, Mar 2013. doi: 10.1093/nar/gks1471.

[124] Feng Zhang, Le Cong, Simona Lodato, Sriram Kosuri, George M. Church, and Paola Arlotta. Efficient construction of sequence-specific TAL effectors for modulating mammalian transcription. *Nature Biotechnology*, 29(2):149–153, Feb 2011. doi: 10.1038/nbt.1775.

[125] Erin Lynn Doyle. *Computational and experimental analysis of TAL effector-DNA binding*. PhD thesis, Iowa State University, 2013. URL http://lib.dr.iastate.edu/etd/13132/.

[126] Samira Kiani, Jacob Beal, Mohammad R. Ebrahimkhani, Jin Huh, Richard N. Hall, Zhen Xie, Yinqing Li, and Ron Weiss. CRISPR transcriptional repression devices and layered circuits in mammalian cells. *Nature Methods*, 11(7):723–726, Jul 2014. doi: 10.1038/nmeth.2969.

[127] Michael Boettcher and Michael T. McManus. Choosing the Right Tool for the Job: RNAi, TALEN, or CRISPR. *Molecular Cell*, 58(4):575–585, 2015. ISSN 1097-2765. doi: 10.1016/j.molcel.2015.04.028.

[128] Andriy Didovyk, Bartłomiej Borek, Jeff Hasty, and Lev Tsimring. Orthogonal Modular Gene Repression in Escherichia coli Using Engineered CRISPR/-Cas9. *ACS Synthetic Biology*, 5(1):81–88, 2016. doi: 10.1021/acssynbio.5b00147. PMID: 26390083.

[129] Stuart Huntley, Daniel M. Baggott, Aaron T. Hamilton, Mary Tran-Gyamfi, Shan Yang, Joomyeong Kim, Laurie Gordon, Elbert Branscomb, and Lisa Stubbs. A comprehensive catalog of human krab-associated zinc finger genes: Insights into the evolutionary history of a large family of transcriptional repressors. *Genome Research*, 16(5):669–677, 2006. doi: 10.1101/gr.4842106.

[130] Daniel B. Hall and Kevin Struhl. The vp16 activation domain interacts with multiple transcriptional components as determined by protein-protein cross-linking in vivo. *Journal of Biological Chemistry*, 277(48): 46043–46050, 2002. doi: 10.1074/jbc.M208911200.

[131] Anil K. Maini. *Digital electronics: principles, devices and applications*. John Wiley & Sons, 2007. ISBN 978-0-470-03214-5.

[132] Paul Horowitz and Winfield Hill. *The art of electronics*. Cambridge University Press, New York, NY, USA, 3 edition, 2015. ISBN 978-0-521-80926-9.

[133] Registry of standard parts. http://parts.igem.org/Main_Page, 2017.

[134] Adam Arkin. Setting the standard in synthetic biology. *Nature Biotechnology*, 26(7):771–774, Jul 2008. doi: 10.1038/nbt0708-771.

[135] Barry Canton, Anna Labno, and Drew Endy. Refinement and standardization of synthetic biological parts and devices. *Nature Biotechnology*, 26(7):787–793, Jul 2008. doi: 10.1038/nbt1413.

[136] Nicolas E. Buchler, Ulrich Gerland, and Terence Hwa. On schemes of combinatorial transcription logic. *Proceedings of the National Academy of Sciences*, 100(9):5136–5141, 2003. doi: 10.1073/pnas.0930314100.

[137] Rafael Silva-Rocha and Victor de Lorenzo. Mining logic gates in prokaryotic transcriptional regulation networks. *FEBS Letters*, 582(8):1237–1244, 2008. doi: 10.1016/j.febslet.2008.01.060.

[138] Virgil A. Rhodius, Thomas H. Segall-Shapiro, Brian D. Sharon, Amar Ghodasara, Ekaterina Orlova, Hannah Tabakh, David H. Burkhardt, Kevin Clancy, Todd C. Peterson, Carol A. Gross, and Christopher A. Voigt. Design of orthogonal genetic switches based on a crosstalk map of σs, anti-σs, and promoters. *Molecular Systems Biology*, 9(1):702–n/a, 2013. doi: 10.1038/msb.2013.58.

[139] 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. doi: 10.1002/bit.20142.

[140] Alec A. K. Nielsen, Bryan S. Der, Jonghyeon Shin, Prashant Vaidyanathan, Vanya Paralanov, Elizabeth A. Strychalski, David Ross, Douglas Densmore, and Christopher A. Voigt. Genetic circuit design automation. *Science*, 352(6281), 2016. doi: 10.1126/science.aac7341.

[141] Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Academic press, 2000. ISBN 0-12-778455-1.

[142] J. D. Murray. *Mathematical biology: I. An introduction*. Springer-Verlag Berlin Heidelberg, 3rd edition, 2002. ISBN 0-387-95223-3.

[143] Hamid Bolouri. *Computational Modeling of Gene Regulatory Networks - A Primer*. Imperial College Press, 2008. ISBN 978-1-84816-220-4.

[144] Ivan V. Maly, editor. *Systems Biology*, volume 500 of *Methods in Molecular Biology*. Humana Press, Springer Science + Business Media, LLC, 2009. ISBN 978-1-934115-64-0. doi: 10.1007/978-1-59745-525-1.

[145] Pablo A. Iglesias and Brian P. Ingalls, editors. *Control Theory and Systems Biology*. MIT Press, 55 Hayward Street, Cambridge, MA 02142, USA, 2010. ISBN 978-0-262-01334-5.

[146] Wilfried Weber and Martin Fussenegger, editors. *Synthetic Gene Networks: Methods and Protocols*, volume 813 of *Methods in Molecular Biology*. Humana Press, Springer Science + Business Media, LLC, 2012. ISBN 978-1-61779-411-7. doi: 10.1007/978-1-61779-412-4.

[147] Paola Lecca, Ian Laurenzi, and Ferenc Jordan. *Deterministic versus stochastic modelling in biochemistry and systems biology*. Woodhead Publishing Series in Biomedicine. Woodhead publishing, 1518 Walnut Street, Suite 1100, Philadelphia, PA 19102-3406, USA, 2013. ISBN 978-1-907568-62-6.

[148] Herbert M Sauro. *Enzyme kinetics for systems biology*. Ambrosius Publishing, 2011. ISBN 978-0982477335.

[149] Jeremy M. Berg, John L. Tymoczko, and Lubert Stryer. *Biochemistry*, chapter 29th. W. H. Freeman, 8th edition, 2015. ISBN 978-1-4641-2610-9.

[150] Gary M. Skinner, Christoph G. Baumann, Diana M. Quinn, Justin E. Molloy, and James G. Hoggett. Promoter Binding, Initiation, and Elongation By Bacteriophage T7 RNA Polymerase: A single-molecule view of the transcription cycle. *Journal of Biological Chemistry*, 279(5):3239–3244, 2004. doi: 10.1074/jbc.M310471200.

[151] Xavier Darzacq, Yaron Shav-Tal, Valeria De Turris, Yehuda Brody, Shailesh M Shenoy, Robert D Phair, and Robert H Singer. In vivo dynamics of RNA polymerase II transcription. *Nature structural & molecular biology*, 14(9):796–806, 2007. doi: 10.1038/nsmb1280.

[152] Iris Jonkers and John T. Lis. Getting up to speed with transcription elongation by rna polymerase ii. *Nature Reviews Molecular Cell Biology*, 16(3):167–177, Mar 2015. doi: 10.1038/nrm3953.

[153] M. Behfar Ardehali and John T. Lis. Tracking rates of transcription and splicing in vivo. *Nature Structural & Molecular Biology*, 16(11):1123–1124, Nov 2009. doi: 10.1038/nsmb1109-1123.

[154] Jarnail Singh and Richard A Padgett. Rates of in situ transcription and splicing in large human genes. *Nature structural & molecular biology*, 16(11):1128–1133, 2009.

[155] Furqan M. Fazal, Cong A. Meng, Kenji Murakami, Roger D. Kornberg, and Steven M. Block. Real-time observation of the initiation of RNA polymerase II transcription. *Nature*, 525(7568):274–277, Sep 2015. doi: 10.1038/nature14882.

[156] Kate L. Henderson, Lindsey C. Felth, Cristen M. Molzahn, Irina Shkel, Si Wang, Munish Chhabra, Emily F. Ruff, Lauren Bieter, Joseph E. Kraft, and M. Thomas Record. Mechanism of transcription initiation and promoter escape by E. coli RNA polymerase. *Proceedings of the National Academy of Sciences*, 114(15):E3032–E3040, 2017. doi: 10.1073/pnas.1618675114.

[157] Hong Yin, Irina Artsimovitch, Robert Landick, and Jeff Gelles. Nonequilibrium mechanism of transcription termination from observations of single rna polymerase molecules. *Proceedings of the National Academy of Sciences*, 96(23):13124–13129, 1999. doi: 10.1073/pnas.96.23.13124.

[158] P. H. von Hippel and T. D. Yager. Transcript elongation and termination are competitive kinetic processes. *Proceedings of the National Academy of Sciences*, 88(6): 2307–2311, 1991. doi: 10.1073/pnas.88.6.2307.

[159] Sandra J. Greive, Steven E. Weitzel, Jim P. Goodarzi, Lisa J. Main, Zvi Pasman, and Peter H. von Hippel. Monitoring rna transcription in real time by using surface plasmon resonance. *Proceedings of the National Academy of Sciences*, 105(9):3315–3320, 2008. doi: 10.1073/pnas.0712074105.

[160] Sabine Arnold, Martin Siemann, Kai Scharnweber, Markus Werner, Sandra Baumann, and Matthias Reuss. Kinetic modeling and simulation of in vitro transcription by phage T7 RNA polymerase. *Biotechnology and Bioengineering*, 72(5):548–561, 2001. doi: 10.1002/1097-0290(20010305)72:5<548::AID-BIT1019>3.0.CO;2-2.

[161] J. E. Bergmann and H. F. Lodish. A kinetic model of protein synthesis. application to hemoglobin synthesis and translational control. *Journal of Biological Chemistry*, 254(23):11927–37, 1979.

[162] Sophia Rudorf, Michael Thommen, Marina V. Rodnina, and Reinhard Lipowsky. Deducing the Kinetics of Protein Synthesis In Vivo from the Transition Rates Measured In Vitro. *PLoS Computational Biology*, 10(10):1–17, 2014. doi: 10.1371/journal.pcbi.1003909.

[163] Ian Crowlesmith and Konrad Gamon. Rate of Translation and Kinetics of Processing of Newly Synthesized Molecules of Two Major Outer-Membrane Proteins, the OmpA and OmpF Proteins, of Escherichia coli K12. *European Journal of Biochemistry*, 124(3):577–583, 1982. ISSN 1432-1033. doi: 10.1111/j.1432-1033.1982.tb06633.x.

[164] Ron Milo and Rob Phillips. *Cell Biology by the Numbers*. Garland Science, Taylor & Francis Group, 1st edition, 2015. ISBN 9780815345374.

[165] Ying-Ja Chen, Kevin Clancy, and Christopher A. Voigt. *Quantitative Biology: From Molecular to Celullar Systems*, chapter Modeling Genetic Parts for Synthetic Biology. Chapman & Hall/CRC Mathematical and Computational Biology Series. CRC Press, Taylor & Francis Group, 6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL 33487-2742, USA, 2013. ISBN 978-1-4398-2723-9.

[166] Joana Pinto Vieira, Julien Racle, and Vassily Hatzimanikatis. Analysis of Translation Elongation Dynamics in the Context of an Escherichia coli Cell. *Biophysical Journal*, 110(9):2120–2131, 2016. doi: 10.1016/j.bpj.2016.04.004.

[167] Erwin Knecht, Carmen Aguado, Jaime Cárcel, Inmaculada Esteban, Juan Miguel Esteve, Ghita Ghislat, José Félix Moruno, José Manuel Vidal, and Rosana Sáez. Intracellular protein degradation in mammalian cells: recent developments. *Cellular and Molecular Life Sciences*, 66(15):2427–2443, 2009. doi: 10.1007/s00018-009-0030-6.

[168] Kamalendu Nath and Arthur L. Koch. Protein Degradation in Escherichia coli. *Journal of Biological Chemistry*, 245(11):2889–2900, 1970. URL http://www.jbc.org/content/245/11/2889.abstract.

[169] Verena Wolf, Rushil Goel, Maria Mateescu, and Thomas A. Henzinger. Solving the chemical master equation using sliding windows. *BMC Systems Biology*, 4(1):42, 2010. doi: 10.1186/1752-0509-4-42.

[170] Donald A. McQuarrie. Stochastic approach to chemical kinetics. *Journal of Applied Probability*, 4(3): 413–478, 1967. doi: 10.2307/3212214.

[171] Daniel T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188(1):404–425, 1992. doi: 10.1016/0378-4371(92)90283-V.

[172] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977. doi: 10.1021/j100540a008.

[173] Daniel T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58(1):35–55, 2007. doi: 10.1146/annurev.physchem.58.032806.104637.

[174] Hong Li and Linda Petzold. Logarithmic Direct Method for discrete stochastic simulation of chemically reacting systems. Technical report, Department of Computer Science, University of California Santa Barbara, Santa Barbara, CA 93106, 2006. Available at: http://goo.gl/bt0lvW.

[175] 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. doi: 10.1016/0021-9991(76)90041-3.

[176] 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(9):1876–1889, 2000. doi: 10.1021/jp993732q.

[177] Yang Cao, Hong Li, and Linda Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *The Journal of Chemical Physics*, 121(9):4059–4067, 2004. doi: 10.1063/1.1778376.

[178] James M. McCollum, Gregory D. Peterson, Chris D. Cox, Michael L. Simpson, and Nagiza F. Samatova. The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. *Computational Biology and Chemistry*, 30(1):39–49, 2006. doi: 10.1016/j.compbiolchem.2005.10.007.

[179] Rajesh Ramaswamy, Nélido González-Segredo, and Ivo F. Sbalzarini. A new class of highly efficient exact stochastic simulation algorithms for chemical reaction networks. *The Journal of Chemical Physics*, 130(24): 244104, 2009. doi: 10.1063/1.3154624.

[180] Alexander Slepoy, Aidan P. Thompson, and Steven J. Plimpton. A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks. *The Journal of Chemical Physics*, 128(20): 205101, 2008. doi: 10.1063/1.2919546.

[181] Daven Sanassy, Paweł Widera, and Natalio Krasnogor. Meta-Stochastic Simulation of Biochemical Models for Systems and Synthetic Biology. *ACS Synthetic Biology*, 4(1):39–47, 2015. doi: 10.1021/sb5001406.

[182] Christian Kuehn. *Multiple Time Scale Dynamics*, volume 191. Springer International Publishing, 2015. ISBN 978-3-319-12315-8. doi: 10.1007/978-3-319-12316-5.

[183] Andre S. Ribeiro and Jason Lloyd-Price. SGN Sim, a Stochastic Genetic Networks Simulator. *Bioinformatics*, 23(6):777, 2007. doi: 10.1093/bioinformatics/btm004.

[184] Joseph O. Dada and Pedro Mendes. Multi-scale modelling and simulation in systems biology. *Integrative Biology*, 3:86–96, 2011. doi: 10.1039/coib00075b.

[185] Jessica S. Yu and Neda Bagheri. Multi-class and multi-scale models of complex biological phenomena. *Current Opinion in Biotechnology*, 39:167–173, Jun 2016. doi: 10.1016/j.copbio.2016.04.002.

[186] Eric L. Haseltine and James B. Rawlings. Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics. *The Journal of Chemical Physics*, 117(15):6959–6969, 2002. doi: 10.1063/1.1505860.

[187] Hye-Won Kang and Thomas G. Kurtz. Separation of time-scales and model reduction for stochastic reaction networks. *The Annals of Applied Probability*, 23(2):529–583, 2013. doi: 10.1214/12-AAP841.

[188] Muruhan Rathinam, Linda R. Petzold, Yang Cao, and Daniel T. Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *The Journal of Chemical Physics*, 119(24): 12784–12794, 2003. doi: 10.1063/1.1627296.

[189] Tamás Székely Jr. and Kevin Burrage. Stochastic simulation in systems biology. *Computational and Structural Biotechnology Journal*, 12(20–21):14–25, 2014. doi: https://doi.org/10.1016/j.csbj.2014.10.003.

[190] Iván M. Lengyel and Luis G. Morelli. Multiple binding sites for transcriptional repressors can produce regular bursting and enhance noise suppression. *Physical Review E*, 95:042412, Apr 2017. doi: 10.1103/PhysRevE.95.042412.

[191] Maria Werner, Ingemar Ernberg, JieZhi Zou, Jenny Almqvist, and Erik Aurell. Epstein-Barr virus latency switch in human B-cells: a physico-chemical model. *BMC Systems Biology*, 1(1), 2007. doi: 10.1186/1752-0509-1-40.

[192] Weinan E, Di Liu, and Eric Vanden-Eijnden. Nested stochastic simulation algorithm for chemical kinetic systems with disparate rates. *The Journal of Chemical Physics*, 123(19):194107, 2005. doi: 10.1063/1.2109987.

[193] Hermannus Kempe, Anne Schwabe, Frédéric Crémazy, Pernette J. Verschure, and Frank J. Bruggeman. The volumes and transcript counts of single cells reveal concentration homeostasis and capture biological noise. *Molecular Biology of the Cell*, 26(4):797–804, 2015. doi: 10.1091/mbc.E14-08-1296.

[194] kstest2: Two-sample Kolmogorov-Smirnov test. https://www.mathworks.com/help/stats/kstest2.html, 2017.

[195] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global Sensitivity Analysis. The Primer*. John Wiley & Sons, Ltd, 2008. ISBN 978-0-470-05997-5. doi: 10.1002/9780470725184.

[196] Andrea Saltelli, Stefano Tarantola, Francesca Campolongo, and Marco Ratto. *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. John Wiley & Sons, Ltd, 2004. ISBN 9780470870952. doi: 10.1002/0470870958.fmatter.

[197] Max D. Morris. Factorial Sampling Plans for Preliminary Computational Experiments. *Technometrics*, 33 (2):161–174, 1991.

[198] Francesca Campolongo, Jessica Cariboni, and Andrea Saltelli. An effective screening design for sensitivity analysis of large models. *Environmental Modelling & Software*, 22(10):1509–1518, 2007. doi: 10.1016/j.envsoft.2006.10.004.

[199] Gürkan Sin and Krist V. Gernaey. Improving the morris method for sensitivity analysis by scaling the elementary effects. *Computer Aided Chemical Engineering*, 26:925–930, 2009. doi: 10.1016/S1570-7946(09)70154-3.

[200] Thomas Sumner. *Sensitivity analysis in systems biology modelling and its application to a multi-scale model of blood glucose homeostasis*. PhD thesis, Centre for Mathematics and Physics in the Life Sciences and Experimental Biology, University College London, 2010.

[201] Andrea Saltelli, Marco Ratto, Stefano Tarantola, and Francesca Campolongo. Sensitivity Analysis for Chemical Models. *Chemical Reviews*, 105(7): 2811–2828, 2005. doi: 10.1021/cr040659d.

[202] Kenny Q. Ye. Orthogonal Column Latin Hypercubes and Their Application in Computer Experiments. *Journal of the American Statistical Association*, 93(444):1430–1439, 1998. doi: 10.1080/01621459.1998.10473803.

[203] Boxin Tang. Orthogonal Array-Based Latin Hypercubes. *Journal of the American Statistical Association*, 88(424):1392–1397, 1993. doi: 10.1080/01621459.1993.10476423.

[204] Elías Zamora-Sillero, Marc Hafner, Ariane Ibig, Joerg Stelling, and Andreas Wagner. Efficient characterization of high-dimensional parameter spaces for systems biology. *BMC Systems Biology*, 5(1):142, 2011. doi: 10.1186/1752-0509-5-142.

[205] Rudiyanto Gunawan, Yang Cao, Linda Petzold, and Francis J Doyle III. Sensitivity Analysis of Discrete Stochastic Systems. *Biophysical Journal*, 88(4):2530–2540, 2005. doi: 10.1529/biophysj.104.053405.

[206] Andrea Degasperi and Stephen Gilmore. *Sensitivity Analysis of Stochastic Models of Bistable Biochemical Reactions*, pages 1–20. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi: 10.1007/978-3-540-68894-5_1.

[207] Michał Komorowski, Maria J. Costa, David A. Rand, and Michael P. H. Stumpf. Sensitivity, robustness, and identifiability in stochastic chemical kinetics models. *Proceedings of the National Academy of Sciences*, 108(21):8645–8650, 2011. doi: 10.1073/pnas.1015814108.

[208] Patrick W. Sheppard, Muruhan Rathinam, and Mustafa Khammash. A pathwise derivative approach to the computation of parameter sensitivities in discrete stochastic chemical systems. *The Journal of Chemical Physics*, 136(3):034115, 2012. doi: 10.1063/1.3677230.

[209] Ankit Gupta and Mustafa Khammash. Unbiased Estimation of Parameter Sensitivities for Stochastic Chemical Reaction Networks. *SIAM Journal on Scientific Computing*, 35(6):A2598–A2620, 2013. doi: 10.1137/120898747.

[210] Elizabeth Skubak Wolf and David F. Anderson. Hybrid pathwise sensitivity methods for discrete stochastic models of chemical reaction systems. *The Journal of Chemical Physics*, 142(3):034103, 2015. doi: 10.1063/1.4905332.

[211] Package "Sensitivity". https://cran.r-project.org/web/packages/sensitivity/sensitivity.pdf, 2013.

[212] Araz Hashemi, Marcel Núñez, Petr Plecháč, and Dionisios G. Vlachos. Stochastic averaging and sensitivity analysis for two scale reaction networks. *The Journal of Chemical Physics*, 144(7):074104, 2016. doi: 10.1063/1.4942008.

[213] M. Núñez and D. G. Vlachos. Steady state likelihood ratio sensitivity analysis for stiff kinetic Monte Carlo simulations. *The Journal of Chemical Physics*, 142(4): 044108, 2015. doi: 10.1063/1.4905957.

[214] Oleg V. Demin, Hans V. Westerhoff, and Boris N. Kholodenko. Control analysis of stationary forced oscillations. *The Journal of Physical Chemistry B*, 103 (48):10695–10710, 1999. doi: 10.1021/jp991597b.

[215] Sofia Wichert, Konstantinos Fokianos, and Korbinian Strimmer. Identifying periodically expressed transcripts in microarray time series data. *Bioinformatics*, 20(1):5–20, 2004. doi: 10.1093/bioinformatics/btg364.

[216] Yoav Benjamini and Yosef Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1): 289–300, 1995.

[217] Lawrence S Young and Alan B Rickinson. Epstein-Barr virus: 40 years on. *Nature Reviews Cancer*, 4(10): 757–768, 2004. doi: 10.1038/nrc1452.

[218] Jeffrey I. Cohen, Anthony S. Fauci, Harold Varmus, and Gary J. Nabel. Epstein-Barr Virus: An Important Vaccine Target for Cancer Prevention. *Science Translational Medicine*, 3(107):107fs7, 2011. doi: 10.1126/scitranslmed.3002878.

[219] Eriko Maeda, Masaaki Akahane, Shigeru Kiryu, Nobuyuki Kato, Takeharu Yoshikawa, Naoto Hayashi, Shigeki Aoki, Manabu Minami, Hiroshi Uozaki, Masashi Fukayama, and Kuni Ohtomo. Spectrum of Epstein-Barr virus-related diseases: a pictorial review. *Japanese Journal of Radiology*, 27(1):4–19, 2009. doi: 10.1007/s11604-008-0291-2.

[220] E.S. Robertson. *Epstein-Barr Virus: Latency and Transformation*. Caister Academic Press, 2010.

[221] K. Goldsmith, L. Bendell, and L. Frappier. Identification of EBNA1 amino acid sequences required for the interaction of the functional elements of the Epstein-Barr virus latent origin of DNA replication. *Journal of Virology*, 67(6):3418–3426, 1993.

[222] Lori Frappier. The Epstein-Barr Virus EBNA1 Protein. *Scientifica*, 2012, 2012.

[223] L. Frappier and M. O'Donnell. Overproduction, purification, and characterization of EBNA1, the origin binding protein of Epstein-Barr virus. *Journal of Biological Chemistry*, 266(12):7819–7826, 1991.

[224] Edward Yang, Erik van Nimwegen, Mihaela Zavolan, Nikolaus Rajewsky, Mark Schroeder, Marcelo Magnasco, and James E. Darnell. Decay Rates of Human mRNAs: Correlation With Functional Characteristics and Sequence Attributes. *Genome Research*, 13(8): 1863–1872, 2003. doi: 10.1101/gr.1272403.

[225] Derek F. J. Ceccarelli and Lori Frappier. Functional analyses of the EBNA1 origin DNA binding protein of Epstein-Barr virus. *Journal of Virology*, 74(11): 4939–4948, 2000. doi: 10.1128/JVI.74.11.4939-4948.2000.

[226] The UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 45 (D1):D158–D169, 2017. doi: 10.1093/nar/gkw1099.

[227] R. F. Ambinder, W. A. Shah, D. R. Rawlins, G. S. Hayward, and S. D. Hayward. Definition of the sequence requirements for binding of the EBNA-1 protein to its palindromic target sites in Epstein-Barr virus DNA. *Journal of Virology*, 64(5):2369–2379, 1990.

[228] T. Tsurumi. Primer terminus recognition and highly processive replication by Epstein-Barr virus DNA polymerase. *Biochemical Journal*, 280(Pt 3):703–708, 1991. doi: 10.1042/bj2800703.

[229] Jelena Levitskaya, Anatoly Sharipo, Ainars Leonchiks, Aaron Ciechanover, and Maria G. Masucci. Inhibition of ubiquitin/proteasome-dependent protein degradation by the Gly-Ala repeat domain of the Epstein–Barr virus nuclear antigen 1. *Proceedings of the National Academy of Sciences*, 94(23):12616–12621, 1997.

[230] Matthew G. Davenport and Joseph S. Pagano. Expression of EBNA-1 mRNA Is Regulated by Cell Cycle during Epstein-Barr Virus Type I Latency. *Journal of Virology*, 73(4):3154–3161, 1999.

[231] Pratik C. Shah, Eric Bertolino, and Harinder Singh. Using altered specificity Oct-1 and Oct-2 mutants to analyze the regulation of immunoglobulin gene transcription. *The EMBO Journal*, 16(23):7105–7117, 1997. doi: 10.1093/emboj/16.23.7105.

[232] Jingyue Jia, Yongxin Jin, Ting Bian, Donghai Wu, Lijun Yang, Naohiro Terada, Weihui Wu, and Shouguang Jin. Bacterial Delivery of TALEN Proteins for Human Genome Editing. *PLoS ONE*, 9(3):1–9, 03 2014. doi: 10.1371/journal.pone.0091547.

[233] Stefan C. Materna and Wolfgang Marwan. Estimating the number of plasmids taken up by a eukaryotic cell during transfection and evidence that antisense RNA abolishes gene expression in Physarum polycephalum. *FEMS Microbiology Letters*, 243(1):29–35, 2005. doi: 10.1016/j.femsle.2004.11.035.

[234] Richard N. Cohen, Marieke AEM van der Aa, Nichole Macaraeg, Ai Ping Lee, and Francis C. Szoka. Quantification of Plasmid DNA Copies in the Nucleus after Lipoplex and Polyplex Transfection. *Journal of Controlled Release*, 135(2):166–174, Apr 2009. doi: 10.1016/j.jconrel.2008.12.016.

[235] Kathleen E. McGinness, Tania A. Baker, and Robert T. Sauer. Engineering Controllable Protein Degradation. *Molecular Cell*, 22(5):701–707, 2006. ISSN 1097-2765. doi: 10.1016/j.molcel.2006.04.027.

[236] Didier Gonze and Wassim Abou-Jaoudé. The Goodwin Model: Behind the Hill Function. *PLOS ONE*, 8(8):1–15, 08 2013. doi: 10.1371/journal.pone.0069573.

[237] Benedicte Mengel, Alexander Hunziker, Lykke Pedersen, Ala Trusina, Mogens H. Jensen, and Sandeep Krishna. Modeling oscillatory control in NF-κB, p53 and Wnt signaling. *Current Opinion in Genetics & Development*, 20(6):656–664, 2010. doi: 10.1016/j.gde.2010.08.008.

[238] D. T. Gillespie, L. R. Petzold, and Y. Cao. Comment on "Nested stochastic simulation algorithm for chemical kinetic systems with disparate rates". *The Journal of Chemical Physics*, 126(13):137101, 2007. doi: 10.1063/1.2567036.

[239] Weinan E, Di Liu, and Eric Vanden-Eijnden. Response to "Comment on 'Nested stochastic simulation algorithm for chemical kinetic systems with disparate rates'. *The Journal of Chemical Physics*, 126 (13):137102, 2007. doi: 10.1063/1.2567071.

[240] B. P. Ingalls. Autonomously oscillating biochemical systems: parametric sensitivity of extrema and period. *IEE Proceedings - Systems Biology*, 1(1):62–70, 2004. doi: 10.1049/sb:20045005.

[241] Bao-Yun Lu and Hong Yue. Developing Objective Sensitivity Analysis of Periodic Systems: Case Studies of Biological Oscillators. *Acta Automatica Sinica*, 38(7):1065–1073, 2012. doi: 10.1016/S1874-1029(11)60280-2.

[242] Hong Li and Linda Petzold. Efficient Parallelization of the Stochastic Simulation Algorithm for Chemically Reacting Systems On the Graphics Processing Unit. *International Journal of High Performance Computing Applications*, 24(2):107–116, May 2010. ISSN 1094-3420. doi: 10.1177/1094342009106066.

[243] Marco S Nobile, Paolo Cazzaniga, Daniela Besozzi, Dario Pescini, and Giancarlo Mauri. cuTauLeaping: A GPU-Powered Tau-Leaping Stochastic Simulator for Massive Parallel Analyses of Biological Systems. *PLoS One*, 9(3), 2014. doi: 10.1371/journal.pone.0091963.

[244] Ivan Komarov and Roshan M. D'Souza. Accelerating the Gillespie exact stochastic simulation algorithm using hybrid parallel execution on graphics processing units. *PloS one*, 7(11):e46693, 2012. doi: 10.1371/journal.pone.0046693.

[245] Paolo Cazzaniga. *Stochastic algorithms for biochemical processes*. PhD thesis, Università degli Studi di Milano-Bicocca, Milano, Italy, 2010.

[246] M. Aldinucci, M. Drocco, F. Tordini, M. Coppo, and M. Torquati. Parallel Stochastic Simulators in System Biology: The Evolution of the Species. In *21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 410–419, February 2013. doi: 10.1109/PDP.2013.66.

[247] K. Burrage, P. M. Burrage, N. Hamilton, and T. Tian. *Compute-Intensive Simulations for Cellular Models*, pages 79–119. John Wiley & Sons, Inc., 2005. ISBN 9780471756507. doi: 10.1002/0471756504.ch4.

[248] Message Passing Interface Forum. `https://www.mpi-forum.org/`, 2016.

[249] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-passing Interface*. Scientific and engineering computation. MIT Press, 2014. ISBN 9780262527392.

[250] Barbara Chapman, Gabriele Jost, and Ruud Van Der Pas. *Using OpenMP: portable shared memory parallel programming*, volume 10. MIT press, 2008.

[251] The Open Multi Processing API. `http://www.openmp.org/`, 2016.

[252] Vijaya Smitha Kolli, Hui Liu, Jieyue He, Michelle Hong Pan, and Yi Pan. Calculating genomic distances in parallel using OpenMP. In *Transactions on Computational Systems Biology II*, pages 113–123. Springer, 2005.

[253] Timothy Chapman and Ananth Kalyanaraman. An OpenMP Algorithm and Implementation for Clustering Biological Graphs. In *Proceedings of the 1st Workshop on Irregular Applications: Architectures and Algorithms*, IA3 2011, pages 3–10. ACM, 2011. ISBN 978-1-4503-1121-2. doi: 10.1145/2089142.2089146.

[254] Tianhai Tian and K. Burrage. Parallel implementation of stochastic simulation for large-scale cellular processes. In *Eighth International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05)*, pages pp. 621–626, 2005. doi: 10.1109/HPCASIA.2005.67.

[255] Martin J. Chorley and David W. Walker. Performance analysis of a hybrid MPI/OpenMP application on multi-core clusters. *Journal of Computational Science*, 1(3):168–174, 2010. ISSN 1877-7503. doi: 10.1016/j.jocs.2010.05.001.

[256] R. Rabenseifner, G. Hager, and G. Jost. Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes. In *17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 427–436, 2009. doi: 10.1109/PDP.2009.43.

[257] E. de Araujo Macedo and A. Boukerche. Hybrid MPI/OpenMP Strategy for Biological Multiple Sequence Alignment with DIALIGN-TX in Heterogeneous Multicore Clusters. In *IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, pages 418–425, 2011. doi: 10.1109/IPDPS.2011.169.

[258] Daniele D'Agostino, Giulia Pasquale, Andrea Clematis, Carlo Maj, Ettore Mosca, Luciano Milanesi, and Ivan Merelli. Parallel Solutions for Voxel-Based Simulations of Reaction-Diffusion Systems. *BioMed research international*, 2014, 2014.

[259] Nvidia Corporation. NVIDIA Tesla P100. Whitepaper. https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf, 2016.

[260] Nvidia Corporation. Compute Unified Device Architecture. https://developer.nvidia.com/cuda-zone, 2016.

[261] The open standard for parallel programming of heterogeneous systems. https://www.khronos.org/opencl/, 2016.

[262] J. Fang, A. L. Varbanescu, and H. Sips. A Comprehensive Performance Comparison of CUDA and OpenCL. In *International Conference on Parallel Processing*, pages 216–225, 2011. doi: 10.1109/ICPP.2011.45.

[263] Sain-Zee Ueng, Melvin Lathara, Sara S. Baghsorkhi, and Wen-mei W. Hwu. CUDA-Lite: Reducing GPU Programming Complexity. In Xipeng Shen, Frank Mueller, and James Tuck, editors, *Languages and Compilers for Parallel Computing: 21th International Workshop, LCPC 2008, Edmonton, Canada, July 31 - August 2, 2008, Revised Selected Papers*, pages 1–15. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-89740-8. doi: 10.1007/978-3-540-89740-8.

[264] Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming, Portable Documents*. Addison-Wesley Professional, 2010.

[265] Jakub Kurzak, David A Bader, and Jack Dongarra. *Scientific computing with multicore and accelerators*. CRC Press, 2010.

[266] Lorenzo Dematté and Davide Prandi. GPU computing for systems biology. *Briefings in Bioinformatics*, 11 (3):323–333, 2010. doi: 10.1093/bib/bbq006.

[267] Jim Jeffers and James Reinders. *Intel Xeon Phi Coprocessor High Performance Programming*. Morgan Kaufmann, 2013. ISBN 978-0-12-410414-3. doi: 10.1016/B978-0-12-410414-3.00017-7.

[268] James Reinders. An Overview of Programming for Intel© Xeon™ processors and Intel Xeon© Phi™ coprocessors. Intel Corporation. https://goo.gl/V0q95y, 2012.

[269] Intel Corporation. Intel© Xeon Phi™ Coprocessor x100 Product Family, Datasheet. http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/xeon-phi-coprocessor-datasheet.pdf, 2015.

[270] Victor Malyshkin. *Parallel Computing Technologies: 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31-September 4, 2015, Proceedings*, volume 9251. Springer, 2015.

[271] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524, 2003. doi: 10.1093/bioinformatics/btg015.

[272] Benjamin J. Bornstein, Sarah M. Keating, Akiya Jouraku, and Michael Hucka. LibSBML: an API library for SBML. *Bioinformatics*, 24(6):880–881, 2008. doi: 10.1093/bioinformatics/btn051.

[273] Mathworks®inc., matlab®simbiology toolbox. http://www.mathworks.com/products/simbiology/, 2016.

[274] Stefan Hoops, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. COPASI—a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067–3074, 2006. doi: 10.1093/bioinformatics/btl485.

[275] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8 (1):3–30, 1998.

[276] Makoto Matsumoto and Takuji Nishimura. Dynamic creation of pseudorandom number generators. *Monte Carlo and Quasi-Monte Carlo Methods*, 2000:56–69, 1998.

[277] J. Passerat-Palmbach, C. Mazel, and C. D. R. Hill. Pseudo-random streams for distributed and parallel stochastic simulations on GP-GPU. *Journal of Simulation*, 6(3):141–151, 2012. doi: 10.1057/jos.2012.8.

[278] Dynamic creator for mersenne twister. https://github.com/MersenneTwister-Lab/dcmt, 2015.

[279] Parallelizing mersenne twister. https://www.fixstars.com/en/opencl/book/OpenCLProgrammingBook/mersenne-twister/, 2016.

[280] Pierre L'Ecuyer. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, 47(1): 159–164, 1999. doi: 10.1287/opre.47.1.159.

[281] Mutsuo Saito and Makoto Matsumoto. Variants of Mersenne twister suitable for graphic processors. *ACM Transactions on Mathematical Software (TOMS)*, 39(2):12, 2013.

[282] Jonathan Passerat-Palmbach, Claude Mazel, Antoine Mahul, and David Hill. Reliable initialization of GPU-enabled parallel stochastic simulations using mersenne twister for graphics processors. *arXiv:1501.07701*, 2015.

[283] OpenCL™ and the AMD APP SDK. Whitepaper. http://developer.amd.com/resources/documentation-articles/articles-whitepapers/opencl-and-the-amd-app-sdk/, 2011.

[284] Intel Corporation. Intel© Xeon Phi™ Coprocessor, Developer's Quick Start Guide. Whitepaper. https://software.intel.com/sites/default/files/managed/26/d6/Intel_Xeon_Phi_Quick_Start_Developers_Guide-MPSS-3.4.pdf, 2011.

[285] Khronos™ Group. OpenCL Overview. https://www.khronos.org/assets/uploads/developers/library/overview/opencl_overview.pdf, 2015.

[286] Khronos™ Group. The opencl specification, version 1.2. https://www.khronos.org/registry/cl/specs/opencl-1.2.pdf, 2012.

[287] Benedict R. Gaster and Lee Howes. The OpenCL C++ Wrapper API. https://www.khronos.org/registry/cl/specs/opencl-cplusplus-1.2.pdf, 2012.

[288] Benedict Gaster, Lee Howes, David R Kaeli, Perhaad Mistry, and Dana Schaa. *Heterogeneous Computing with OpenCL: Revised OpenCL 1.2 Edition*. Morgan Kaufmann, 2013.

[289] Ravishekhar Banger and Koushik Bhattacharyya. *OpenCL Programming by Example*. Packt Publishing Ltd, 2013.

[290] Aaftab Munshi, Benedict Gaster, Timothy G Mattson, and Dan Ginsburg. *OpenCL programming guide*. Pearson Education, 2011.

[291] Michał Komorowski, Justina Žurauskiene, and Michael P. H. Stumpf. StochSens – matlab package for sensitivity analysis of stochastic chemical systems. *Bioinformatics*, 28(5):731–733, 2012. doi: 10.1093/bioinformatics/btr714.

NOMENCLATURE

M_p	the number of all the different promoters in the GRN
$\mathbf{A}$	activation matrix
$\mathbf{B}^*$	the orientation matrix
$\mathbf{D}^*$	the sign delta matrix
$\mathbf{R}^*$	random trajectory generator matrix
c_m	the copy number of the m-th promoter of the GRN
n_m	the number of transcription factor binding sites in the m-th promoter of the GRN

ABBREVIATIONS

API	application programming interface
ATP	adenosine triphosphate
bseSSA	binding sites evolution stochastic simulation algorithm
CLE	chemical Langevin equation
CME	chemical master equation
CMOS	complementary metal oxid semiconductor
CPU	central processing unit
CRISPR	clustered regularly interspaced short palindromic repeats
CRN	cis-regulatory network
CUDA	Compute Unify Device Architecture
DCMT	Dynamic Creator for Mersenne Twister
DFS	Discrete Fourier series
DFT	Discrete Fourier Transform
DM	SSA direct method
DMSSA	dynamic multi-scale stochastic simulation algorithm
DNA	deoxyribonucleic acid
EBNA1	Epstein–Barr nuclear antigen 1
EBV	Epstein-Barr virus
FCS	functional complete set of logic operators
FFT	Fast Fourier Transform

FR	Family of Repeats
FRM	first reaction method
GFP	green fluorescent protein
GMO	genetically modified organism
GPGPU	general purpose graphic processing unit
GRN	gene regulatory network
ITRS	International Technology Roadmap for Semiconductors
KRAB	Krüppel associated box repressor
LDM	SSA logarithmic direct method
LHS	Latin Hypercube sampling
MNO	minimal normal form
MPI	message passing interface
mRNA	messenger RNA
MSE	Morris screening experiment
MSSA	multi-scale stochastic simulation algorithm
MT	Mersenne Twister
NRM	next reaction method
nSSA	nested stochastic simulation algorithm
OAT	on at a time approach
ODE	ordinary differential equation
ODM	optimized direct method
OpenCL	open computing language
OpenMP	open multi-processing
ORF	open reading frame
ParMSSA	a parallel multi-scale stochastic simulation algorithm engine
PCA	Principal component analyses

PDF	probability density function
PDM	partial propensities direct method
PEA	partial equilibrium approximation
PPI	protein-protein interactions network
PRNG	pseudo-random numbers generator
QSSA	quasy steady-state assumption
RNA	ribonucleic acid
RNAP	RNA polymerase enzyme
RRE	reaction rate equation
RTGM	random trajectory generator matrix
SBML	systems biology markup language
SDM	sorting direct method
SGRN	synthetic GRN
SIMD	single instruction multiple data
SSA	stochastic simulation algorithm
SSA-CR	SSA composition rejection algorithm
ssSSA	slow-scale stochastic simulation algorithm
TALE	transcription activator-like effector
TF	transcription factor
TFBS	transcription factor binding site
tRNA	transfer RNA
ULSI	ultra large scale integration
YAML	yet another markup language
ZFTF	zinc finger transcription factor

INDEX