**University of Milano-Bicocca**
**Doctorate School of Sciences**
**Ph.D Program in Computer Science**
**XXVI Cycle**


# Sensitivity Analysis for Computational Models of Biochemical Systems

Doctoral Thesis of
Carlo Maj

Supervisors: Prof. Giancarlo Mauri
Dr. Luciano Milanesi
Tutor: Prof. Marco Antoniotti
Coordinator: Prof. Stefania Bandini

January 2011 - December 2013

*If the doors of perception were cleansed, everything would appear to man as it is: infinite*

William Blake

# Acknowledgements

There are many people that I want and need to thank for their support in the years of my doctorate. First of all, I would like to thank my supervisors, Giancarlo Mauri and Luciano Milanesi for their scientific support and for have hosted me in their research groups, at the the Department of Informatics, Systems, and Communications of Milan-Bicocca University (DISCo), and at the Institute of Biomedical Technologies (ITB) of the Italian National Research Center (CNR).

Then, I have to thanks all the researchers that have guided me in my scientific activity, starting from Ettore Mosca and Ivan Merelli at ITB, to Daniela Besozzi, Dario Pescini, and Paolo Cazzaniga for the works at DISCo. A special thanks also to all the staff of Bioinformatics group at ITB, Roberta Alfieri, Federica Chiappori, Pasqualina D'Ursi Ivan Kel, Matteo Gnocchi, John Hatton, Alessandra Mezzelani, Marco Moscatelli, Alessandro Orro, Gabriele Trombetti, and Federica Viti for the nice moments spent together.

I am also grateful to all the PhD students at DISCo with whom I shared beautiful and stressful experiences, among others, Stefano Beretta, Enrico Porreca, Luca Manzoni, Luca Panziera, Daniele Ramazzotti, Davide Castaldi, Marco Nobile, Riccardo Colombo, Andrea Citrolo, Fabrizio Marini, and Luca Corolli.

I thanks also the other people with whom I had the pleasure to work with in different projects, among others Annamaria Bevilacqua, Gianfranco Canti, Claudia Raibulet, Giulia Pasquale, Andrea Clematis, Daniele D'Agostino, Anurada Lakshminarayana, and Jim Schaff.

A particular thank also to my tutor Marco Antoniotti for helping me in having a wider view of computer science.

I am very grateful to Ann Rundell and Greg Buzzard for have hosted me for 6 months in their research group at the Weldon school of Biomedical Engineering of Purdue University.

A big thanks to all the Purdue colleagues, Nimisha Bajaj, Mark Hamilton, Thembi Mdluli, Michael Pargett, Joyatee Sarker, and Ankush Chakrabarty, for the assistance they gave me and for making me fell as I was always be in the group.

A particular thanks to Vu Dihn for the productive collaboration that we carried on.

A big thanks also to the different people that supported me in different ways in my US experience, among others Kristin Gikas, Moe Steelman, Yan Lì.

A special thanks also to P(G) Sara Balzan who I met in West Lafayette.

To conclude, I have to thank my parents for all their love and support, I would certainly have never arrived to write a PhD dissertation without their constant encouragement to follow my passions despite the uncertain future.

# Contents

# Introduction

## Computational challenges in systems biology

The interaction between computer science and biological sciences is progressively becoming more relevant over time. Computing and biology have been converging in many aspects: from computer science perspective, biological mechanisms have provided the cues for new models of computation, while from biology perspective, the capability of computer science to simulate the dynamics of complex systems, has provided new insights in the understanding of biochemical systems behavior.

This is particularly true in systems biology, an inter-disciplinary field of science aimed at the analysis of biological systems using an holistic perspective, as opposed to the classic reductionism approach which has led biological science until the 20 century. The reductionism approach is mainly based on the descriptive analysis of the different elements composing a system, a strategy that has shown not to be able to completely describe the regulation of complex systems (e.g., biochemical systems.). In fact, the behavior in a complex system is often an emergent property arising from the interactions of the different parts composing the overall network.
As a matter of fact, computer science plays a major role in the transition of the approach in which biological systems are analyzed, from descriptive modalities to systematic methods (Figure 1).

In particular, the transition involves:

- from qualitative biology to quantitative science
- from static description to dynamics properties evaluation
- from single level analysis to system-level understanding

As a first computational step in the analysis of biochemical systems, a variety of data mining approaches can be exploited to analyze experimental data in order to infer potential connections in the networks underling the different biochemical processes. The retrieved information can then be used for the development of formal models, possibly mechanistic, that can be subsequently simulated using different kind of algorithms (e.g., ODE solvers,

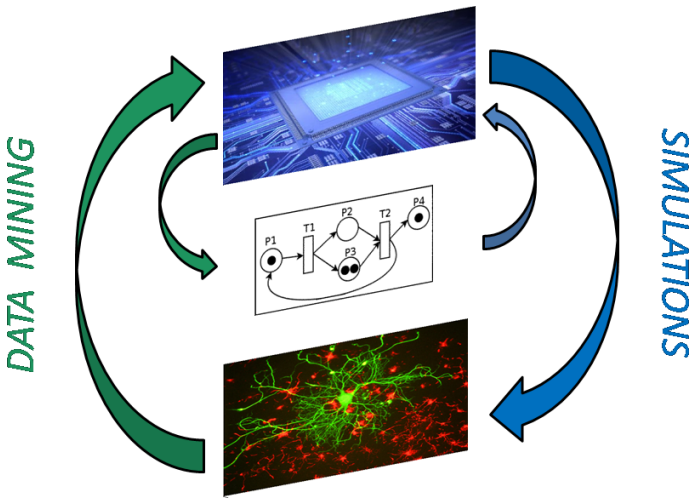


Figure 1: **Computer science and systems biology**
Computer science plays a pivotal role in systems biology, starting from data mining techniques for the analysis of high-throughput biological data (e.g., gene expression, sequence analysis), to the implementation and simulation of computational models for studying biochemical systems functioning.

simulator of rule based systems). Concerning this topic, a number of different computational approaches have been developed to analyze and simulate the dynamic evolutions of biochemical systems exploiting computational models.

The advantage to deal with computational models is that they allow a fast and efficient evaluation of properties that would be difficult to study otherwise, for both economic reasons (e.g., experiments requiring high investments), and technical motivations (e.g., analysis of phenomena at particular low or high temporal scales).

However, given the complexity of biological phenomena, the analysis of these models poses several issues both at theoretical level (e.g., big input factor space, stochasticity), and at implementation level (e.g., the high computational demand of simulation algorithms). Throughout the thesis a number of different computational aspects related to the analysis and the simulations of complex biochemical systems will be analyzed. In this context, sensitivity analysis, a technique describing how the output of a model is affected by its input values, can shed light in the understanding of computational models behavior. In effect, the main focus of the thesis concerns how to apply sensitivity analysis to study the system behavior according to the model type (e.g., deterministic rather than stochastic), and the property to analyze (e.g., steady state value rather than discriminating between different qualitative dynamics).

## Abstract

The thesis concerns the implementation, integration, and development of computational methods for the simulation and the analysis of biochemical systems models. In particular, different aspects related to the exploitation of sensitivity analysis are discussed, from both theoretical and implementation perspective.

The first issue that is analyzed regards how to properly apply the existent sensitivity analysis techniques in different analytical contexts, that is, how to set the analysis according to the aim(s) of the study (e.g., derivatives evaluation, output variance analysis), and to the system characteristics (e.g., deterministic vs. stochastic).

Among others, we propose a pipeline for the analysis of the regulation of a given system property, which is based on the integration of different approaches. The procedure first identifies the pivotal elements affecting the analyzed property using a screening (and less expensive) sensitivity analysis method and then, qualitatively and quantitatively reconstructs the relations between the key factors and the target property with an extensive but more demanding computational method.

Another critical issue that has been addressed is how to apply sensitivity analysis to study the specificity of system response in different conditions. The idea is based on the computation of global sensitivity coefficients in different regions of the model input space, which are associated with the distinct system states that one wants to compare. It is then possible to quantify how the same perturbation affects the system in such states. The method provides important information for the development of strategies aimed at the alteration of the system behavior according to the system state.

Then, we investigated which is the role of spatiality in the regulation of reaction diffusion processes, an aspect that is often neglected to simplify the analysis of biochemical systems dynamics. The analysis of spatial effects has been performed exploiting an innovative stochastic algorithm for the simulation of reaction diffusion processes in crowded environments. The application on a simple case study shows that the system spatiality can not be disregarded, but, on the contrary, the explicit consideration of spatial effects should be carefully evaluated for the correct simulation of the system.

Another aspect that has been taken into account is the analysis of how a system responds to variable perturbations over time, an aspect that allows to study whether model responses are time-dependent or time-independent.

Furthermore, we considered the effects of input variations over qualitative properties, that is, instead of studying the correlations between input values and a numerical property, we analyzed the relations between the input factor space and the attainment of qualitatively different outputs. More specifically, we focused on the analysis of how qualitatively different dynamics emerge from different model configurations. This analysis has been carried out

by coupling a methodology for qualitative output discrimination, and a procedure for the identification of the boundaries among regions within the input factor space, which are associated with qualitatively different dynamics. A simple but effective strategy for the discrimination between steady states and oscillatory regimes is proposed. The method produces a binary index that can be subsequently used in conjunction with an algorithm for detecting the input space sectors in which transitions in the index values occur, ultimately partitions the input factor space according to the corresponding output trends.

Finally, we tackled the issue related to the high computational cost that is usually associated with sensitivity analysis. To this end, different computational aspects concerning the simulation and analysis of computational models of complex biochemical systems are discussed. Most emphasis is placed in the definition of strategies for the reduction of the computational time required to stochastically simulate a model. In particular, we proved how by using parallel and distributed approaches it is possible to reduce the computation time required to compute a single or a set of model simulations, enabling modelers to perform more extensive analysis.

## Thesis overview

The thesis is structured as follows: In chapter 1 a general discussion of the computational techniques for the modeling and simulation of complex dynamic biochemical systems is presented. Chapter 2 concerns the state of the art of sensitivity analysis. In this part the description of some of the most common techniques of sensitivity analysis is provided. In chapter 3 it is described how to implement and integrate sensitivity analysis methods to study the behavior of computational models simulating biochemical systems functioning. In chapter 4 the explanation of an innovative strategy for the mapping between model configurations and system dynamics is provided, the proposed solution partitions the input factor space in different regions according to the output mode. Chapter 5 deals with computational issues related to the implementation and simulation of biochemical models. In particular, different approaches aimed at the reduction of the computation time to perform stochastic simulations are analyzed. Finally, in Chapter 6 a summary of the thesis contributions is provided, along with the identified critical issues that we deem deserve deeper attention in order to improve the capability to simulate and analyze the behavior of computational models of biochemical systems.

# Publications

Part of the work described in this thesis has led to the following publications:

## Journal articles

- Maj C., Mosca E., Merelli I., Mauri G., Milanesi L. *Sensitivity analysis for studying the relation between biochemical reactions and metabolic phenotypes.* (2013) Journal of Bioinformatics and Computational Biology, 11 (1), art. no. 1340002.

- Mosca E., Alfieri R., Maj C., Bevilacqua A., Canti, G., Milanesi L. *Computational modeling of the metabolic states regulated by the kinase Akt.* (2012) Frontiers in Physiology, 3 NOV, art. no. Article 418.

- Corolli L., Maj C., Marini F., Besozzi D., Mauri G. *An excursion in reaction systems: From computer science to biology.* (2012) Theoretical Computer Science, 454, pp. 95-108.

## Conferences

- Maj C., Mosca E., Merelli I., Pescini D., Cazzaniga P., Mauri G., Milanesi L., *Sensitivity analysis for inferring properties of deterministic and stochastic models.* Talk to Sysbiohealth Symposium, Interfacing Physics, Mathematics and Medicine, Bologna 14-15 December 2011. ISBN 978-88-7395-696-9.

- Ramazzotti D., Maj C., Antoniotti M. *A Model of Colonic Crypts using SBML Spatial.* Talk at Wivace 2013, Italian workshop on artificial life and evolutionary computation.

- Merelli I., Pescini D., Mosca E., Cazzaniga P., Maj C., Mauri G., Milanesi L., *Grid computing for sensitivity analysis of stochastic biological models.* Talk at 11th International Conference, PaCT,September 19-23, 2011. 6873 LNCS, pp. 62-73.

- Maj C., Mosca E., Merelli I., Mauri G., Milanesi L., *Identification of the key components to control the behavior of a complex pathway: a study on a model of mitochondrial bioenergetics.* Talk at VIII International Conference on Bioinformatics of Genome Regulation and Structure Systems Biology (BGRSSB-2012) Novosibirsk Russia June 25-29 2012.

- Alfieri R., Maj C., Mosca E., Milanesi L., *Understanding the molecular mechanisms of neurodegenerative diseases using a computational approach.* Poster at Basel Computational Biology Conference "Multiscale Modeling" on June 23- 24, 2011.

- Antoniotti M., Maj C., Ramazzotti D., *A SBML/Spatial Model of Colonic Crypts.* Poster at BITS Annual Meeting 2013 May 21–23, Udine, Italy.

- Antoniotti M., Lakshminarayana A., Ramazzotti D., Schaff J., *Modelling Colonic Crypts with VCell and SBML/Spatial.* Poster at International conference on systems biology, ICSB Copenhagen 2013.

- D'Agostino D., Pasquale G., Clematis A., Maj C., Mosca E., Milanesi L, Merelli I., *A CUDA implementation of the Spatial TAU-leaping in Crowded Compartments (STAUCC) simulator.* Accepted at 22nd Euromicro International Conference on parallel, distributed and network based processing, Turin February 12-14, 2014.

- Vu D., Maj C., Mauri G., Buzzard G., Rundell A., *Mapping the parameter space with oscillatory and steady state system dynamics.* Submitted to American control conference.

- Maj C., Raibulet C., Mauri G., *Self-adaptive simulation time for sensitivity analysis of a stochastic computational model* . Submitted to European control conference.

# Chapter 1

# Computational models of biochemical systems

Throughout the thesis we analyze the behavior of computational models simulating biochemical systems functioning. Given the complexity of biological systems, the development and the analysis of formal models for their description and simulation encompasses a variety of issues at both theoretical and implementation level.

In section 1.1 we discuss which are the typical features that make particularly difficult the analysis of the behavior of such models (e.g., large input factor space, non-linear responses). Subsequently, in section 1.2, we provide a description of the most popular approaches exploited for the formal representation of biochemical systems along with some hints on the corresponding simulative approaches. Finally in section 1.3 we face an annoying problem in the sharing of computational models, which is the availability of a standard format for models representation allowing the exchange of the different analytical and simulative tools. To this purpose, we verified how the new version of Systems Biology Markup Language (SBML), beside the core features based on the description of temporal aspects (i.e., system dynamics), can also be exploited for the definition of the spatial structure of a system.

## 1.1 Complexity and dynamism of biochemical systems

Biochemical systems are among the typical examples of complex, dynamical system. In fact in a biochemical network many elements usually interact with each other and with the environment generating possibly non linear dynamics [1]. Moreover, interactions among entities at a given scale may lead to emergent properties at higher scales in space and/or time. As a consequence, often the behavior of biochemical systems cannot be explained by decomposing the system into subparts.

In this context, systems biology, an integrated field of science which looks at biolog-

ical systems using an holistic perspective, is gaining position as the leading approach on bioscience. Systems biology focuses on the analysis of interactions within biological elements rather than studying biological systems basing on the description of the single parts composing the overall system.

Systems biology is strongly computationally driven because the data obtained from experimental investigations (i.e., High-throughput analysis) need extensive quantitative analysis to be informative [2].

A critical issue which makes particularly difficult the analysis of biochemical systems is that the complexity arises at several levels (Figure 1.1).

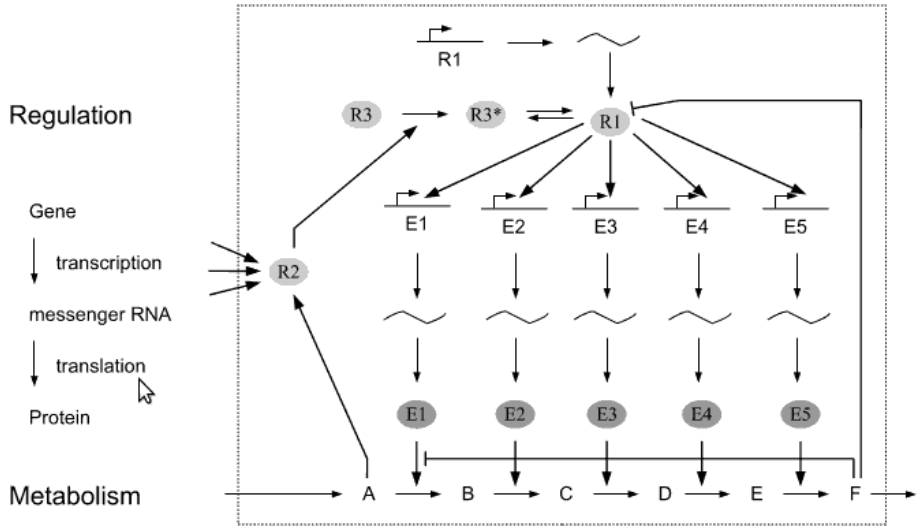


Figure 1.1: **Biochemical systems complexity**

Flow of information within a cell (left), and an example interaction network (right). Figure taken from [3].

Two major features of biochemical systems are robustness to environmental changes in some contexts (e.g., maintain proper metabolite concentration), and adaptivity to deal with modified conditions in other situations (e.g., antibody recognition). If we translate these properties at lower level, they are usually encoded by the presence of feedback loops in biochemical networks. For instance, negative feedback loops have been found to be responsible of homeostasis processes [4] while positive feedback loops have been found to have a role in immune system response [5]. Interpreting all these entangled connections by the solely random simulations of a model (i.e., randomly changing model configurations) is often not possible, instead, it is important to rely on systematic approaches aimed at the analysis of the relationship between input factor values and model behavior (e.g., sensitivity analysis).

## 1.2 Modeling approaches

There exist different methodologies for the computational simulation of biochemical systems. The definition of each simulation framework depends on the way in which a system is described, and in general it is not possible to state with absolute certainty that a method outperforms the others but rather that a given approach is more suited for a given condition.
In the thesis, we deal with four simulation frameworks:

- Continuous deterministic (i.e., ODE systems in chapter 3.1 and 3.3)
- Discrete stochastic (i.e., $\tau$ leaping methods in chapter 3.2 and 3.4 )
- Discrete deterministic (i.e., reaction system in chapter 3.5)
- Continuous stochastic (i.e., ODE+Gaussian noise in chapter 4).

The straightforward approach for the simulation of a biochemical system would be the definition of a mechanistic model based on the definition of all the causal relations present in the system. However, the incomplete knowledge of the biochemical mechanisms makes this kind of solution rarely applicable. Moreover, from a computational point of view, the simulation of each specific event would require a high computational cost to describe the overall system, therefore for a real system it is often unfeasible. As a consequence of the impracticability to provide a comprehensive description of biochemical systems, the different simulation frameworks usually make assumptions on the modeled system, but allow easier simulation in terms of both theoretical interpretation and computational cost.

The common strategy exploited for the description of a biochemical system is the definition of a system of Ordinary Differential Equations (ODEs) representing the system behavior. According to the classic approach, given a biochemical system consisting of a set of molecules interacting each other, the kinetic rate of each biochemical transformation (i.e., chemical reaction) is modeled as a deterministic process whose intensity depends on reactant concentrations.
Different formulas, empirically proven, have been defined to properly describe different biochemical processes (e.g., mass action, Michaelis-Menten). These formulas (i.e., kinetic rates) usually consider as variables reactant concentrations and some fixed input parameter(s) and are arranged to represent the specific kinetic processes (possibly considering the presence of regulatory factors, such as enzymes).
Then, by integrating the different kinetic rates in which each species is involved (i.e., summing formation reactions and subtracting the consumption reactions) it is possible to define an associated ordinary differential equation describing the evolution of the species concentration over time. As a consequence, the dynamic of the overall system is represented by an ODEs system and it can be simulated using the standard ODE solvers [6].

The descriptions of a biochemical system basing on the definition of an ODEs is in effect a way to average the biochemical transformations happening in a system by using kinetic laws (usually experimentally validated). However, in reality, biochemical reactions are triggered by random collision between molecules, therefore a system can be deterministically simulated only if several occurrences of each biochemical event simultaneously happen in the system (i.e., high number number of molecules for each species are present).

Differently from the deterministic modeling, in the classic discrete stochastic modeling the variation of the molecular quantities occur in a discrete way [7]. Formally we can describe a molecular network as a set of $N$ molecular species $\{S_1, \ldots, S_N\}$ which interact through a set of $M$ chemical reactions $\{R_1, \ldots, R_N\}$.
For simplification it is usually assumed that the biochemical system is well-stirred and in thermal equilibrium at some constant temperature and the volume $\Omega$ is constant. To study the dynamics of the system means analyze the evolution of the vector $X(t) = (X_1(t), \ldots, X_N(t))$ where the $X_n(t)$, $n = 1, \ldots, n = N$, is the number of $S_n$ molecules at time $t$. The dynamic of each reaction $R_i$ is defined by a state-change vector $v_i = (v_{i1}, \ldots, v_{iN},)$, where $v_{ij}$ represents the change in $S_j$ molecular population produced by $R_i$. The occurring of $R_i$ depends by the propensity function $a_i(x)dt$ representing the probability that $R_i$ reaction will occur in the next infinitesimal time interval $[t, t + dt]$. In order to analyze the system dynamics, we need to calculate $p(X, t)$, which is the probability that the biochemical system will be in state $X$ at a time $t$. The time evolution of the state probability $P(x, t)$ is governed by the chemical master equation (CME) [8], defined as:

$$\frac{\delta P(x, t|x_0, t_0)}{\delta t} = \sum_{j=1}^{M} [a_j(x - v_j)P(x - v_j, t|x_0, t_0) - a_j(x)P(x, t|x_0, t_0)] \tag{1.1}$$

The CME essentially says that the rate of change in $P(X, t)$ is equal to the probability of entering the state $x$ minus the probability of leaving the state $x$ in unit time. The CME represents a huge system of coupled ordinary differential equations in which there is one differential equation per state of the system, instead of the traditional reaction-rate approach where only one differential equation per species is required. As a consequence the computational cost required to solve the CME increases exponentially with number of molecular species and the number of molecules, therefore the resolution of CME could be achieved only for very small molecular systems (i.e. dimerizations).

The first approach to solve the chemical master equation is represented by the stochastic simulation algorithm (SSA) developed by Gillespie et al. [9]. SSA is essentially an exact procedure for generating realizations of the chemical master equation. Although SSA is in theory the best practice to simulate a biochemical system, due to the fact that it must simulate every reaction event, the computational cost of its resolution can be prohibitively high for many real biological systems.

Approximate accelerated stochastic methods have been developed to speed up the stochastic simulations. Probably, the most popular accelerated simulation approaches are the $\tau$ leaping methods, which compute the system evolution by advancing at every step by a preselected time $\tau$ chosen large enough to encompass more than one reaction events for each step of the algorithm [10]. Therefore, with a $\tau$ leaping method in each step a certain number of reactions can be selected and executed in parallel. So doing, faster simulations can be performed, though the obtained dynamic of the biochemical system is not exact, as in SSA, but it is approximated. To assess the simulation reliability of $\tau$ leaping methods due to approximation process, a threshold error representing a limit in the propensity functions change that is allowed at every step is considered [11].

Although the stochastic simulation framework is closer to real system processes than the classic continuous deterministic approach, this comes at the expense of a high computational cost. Noteworthy, the higher are the molecular quantities the more correct the simulation considering the average conditions are. As a matter of fact, as the number of molecules increases the similarity of the dynamics retrieved by deterministic and stochastic simulations increases as well (Figure 1.2).

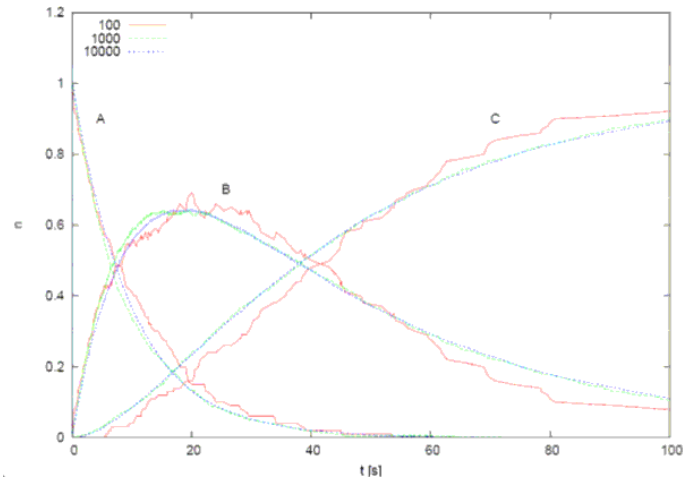


Figure 1.2: **Stochastic simulations**

Enzymatic process considering substrate (A), enzyme-substrate complex (B), and product (C) (here represented using normalized concentrations). Dynamics obtained by setting different molecules number of substrate and enzyme as initial condition (see legend). The stochasticity of the system increases as the number of molecules present in the system becomes lower.

Therefore, when applicable (i.e., in presence of higher concentrations), the deterministic framework should be preferred due to the lower computational cost, while stochastic approaches may be exploited for the evaluation of systems characterized by low amounts of molecules, that is, in which noise effects play a major role in the system dynamics.

The stochastic continuous simulations (another common class of simulation framework) can be placed somewhere in between the continuous deterministic and the stochastic dis-

crete approaches. Hybrid methods consider some level of noise in the system although its presence is simulated by introducing some mathematical terms rather than by considering the inherent biochemical events [12]. This solution can be seen as a compromise for considering in a coarse way the noise effects while preserving a limited computational cost.
In effect, hybrid solutions have been proposed to integrate the capability of discrete stochastic simulation and the computation efficiency of deterministic simulation, more specifically hybrid algorithms usually simulate each biochemical reaction either as a deterministic or as a stochastic process according to the molecular amounts of the reactants [13].

## 1.3 Standard format for systems biology

An important feature for the sharing of the computational tools within the scientific community is the possibility to have a common format in which encode models. In fact, writing models using the same file format enables the development of analytical softwares suited with the specific file structure, thus allowing scientists to share each other models and analytical techniques.

In the context of computational biology, the Systems Biology Markup Language (SBML) is a representation format, based on XML, for communicating and storing computational models of biological processes. SBML is a free and open standard with widespread software support and a community of users and developers [14]. SBML is in continuous evolution according with the progress happening the model developers community. The current most advanced version of SBML is the version 3 which introduces the spatial representation in the model definition [15]. SBML Level 3 Core has explicit support for multi-compartmental modeling where cellular organization is approximated by a small set of compartments (e.g., membrane-bound organelles) containing well-stirred populations of molecules.
As an example, to test the new features of SBML version 3, we developed a simple model representing a basic structure of a model for the description of the colonic crypt. Colonic crypts are invaginations (Figure 1.3) of the connective tissue of human intestine and are supposed to be the site where mutations affecting the stem cells can occur leading to the emergence and progression of colorectal Cancer [16].

The model represents the basic structure of a potential more complex model for the description of the cellular differentiation and the associated migration of the differentiated cells within the crypt. The dynamic part of the model concerns 8 cellular types and 12 cellular transformations, to have a simpler test case, we considered mass action as kinetics (Table 1.1). The representation with SBML core has been done by describing the cellular types as species and the cellular transformation as reactions. In addition to these 8 species, an empty cell is considered in order to represent the empty space in the colonic crypt.

Regarding the spatial extension of SBML3 we exploited the new tag named geometry.

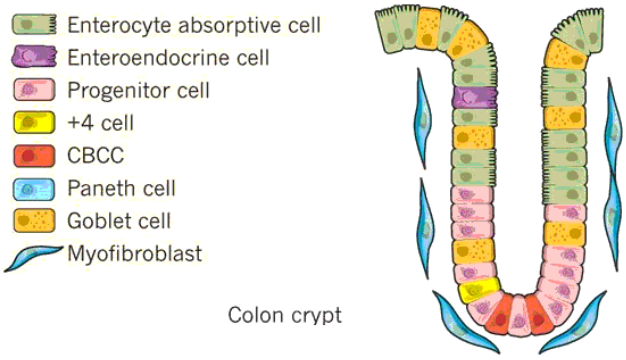


Figure 1.3: **Colonic crypt**

Spatial localization of the different cellular types within the colonic crypt. Figure taken from [17].

| Reaction | Kinetic | Constant |
|---|---|---|
| StemCell $\rightarrow$ 2 StemCell | $StemCell * k_1$ | $k_1 = 2$ |
| StemCell $\rightarrow$ Paneth | $StemCell * k_2$ | $k_2 = 1$ |
| StemCell $\rightarrow$ Ta1 | $StemCell * k_3$ | $k_3 = 1$ |
| 2 Paneth $\rightarrow$ Paneth | $Paneth^2 k_4$ | $k_4 = 1$ |
| Ta1 $\rightarrow$ Ta2a | $Ta1 * k_5$ | $k_5 = 1$ |
| Ta1 $\rightarrow$ Ta2b | $Ta1 * k_6$ | $k_6 = 1$ |
| Ta2a $\rightarrow$ Goblet | $Ta2a * k_7$ | $k_7 = 1$ |
| Ta2a $\rightarrow$ Entero | $Ta2a * k_8$ | $k_8 = 1$ |
| Ta2b $\rightarrow$ Abs | $Ta2b * k_9$ | $k_9 = 1$ |
| 2 Goblet $\rightarrow$ Goblet | $Goblet^2 * k_{10}$ | $k_{10} = 1$ |
| 2 Entero $\rightarrow$ Entero | $Entero^2 * k_{11}$ | $k_{11} = 1$ |
| 2 Abs $\rightarrow$ Abs | $Abs^2 * k_{12}$ | $k_{12} = 1$ |

Table 1.1: **Colonic crypt reactions list**

Each reaction represents a specific cellular transformation.

The geometry tag enables for explicit definition of a spatial environment for the simulation by using the following features:

- *ListOfCoordinateCompartments*: in this sub-tag, the spatial reference frame is defined. Different types of reference frames are permitted. In our case it is a 3-dimensional Cartesian System where the x-axis represents the width, the y-axis is the height and the z-axis is the depth.

- *ListOfDomainTypes*: in this sub-tag homogeneous spatial zones present in the system should be defined. Each spatial zone is intended as being anatomically and physiologically similar and the domain types defined in this tag can refer to multiple concrete domains.

- *ListOfDomains*: the domains represent contiguous regions identified by the same

domain type. For each domain is assigned a position in the reference frame defined before. The domains defined here should match the initial condition of the dynamic model.

- *ListOfAdjacentDomains*: adjacent domain types can be defined here. In our case we have only one domain type (i.e. cell domain type), hence there are not adjacent domains.

- *ListOfGeometryDefinitions*: here is defined the geometrical structure of each domain type. This is an abstract structure to be assigned to the real domains linked through the domain types definition. SBML 3 with spatial extension offers different possible ways to define the geometry. In our case we tested the AnalyticalGeometry option.

We represented the colonic crypt by defining a parallelepiped placed in a 3-dimensional xyz cartesian reference frame. The cells are parallelepipeds placed inside the parallelepiped (Figure 1.4).

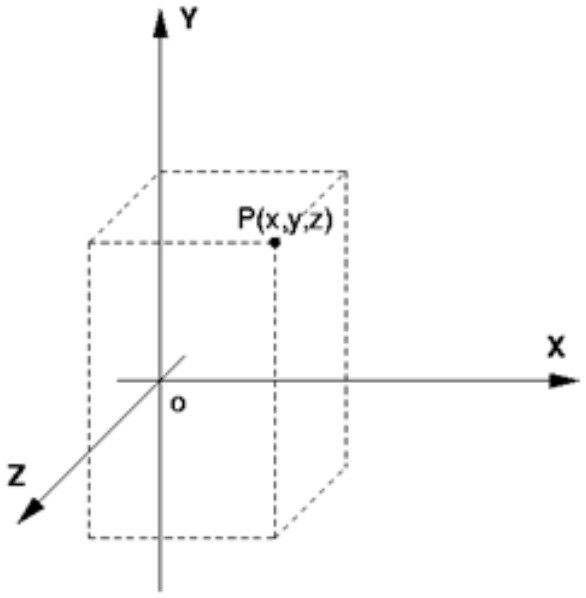


Figure 1.4: **Spatial cell definition in SBML3 model**
The space in discretized in a finite number of cells. Each cell may contain a cellular type or represent an empty space (this is encoded by using different species tag in SBML file).

The dynamic of system spatiality consists in movements upward and downward, which are carried out by changing the species in the corresponding cells.

Despite we developed an elementary and approximated model structure (and therefore less relevant from the biological viewpoint, compared to several already developed and published models), we verified how it is possible to exploit SBML3 in order to include both the dynamic and the spatial architecture in a model definition [18].

# Chapter 2

# Sensitivity analysis

The exploitation of models to study the behavior of a system is a widespread practice in science. A system is often described by mean of a mathematical formalization and the behavior is usually analyzed through computational simulations. Having a model of a system allows to perform computational analysis of its behavior rather than observing the real system states at different moment and/or in different conditions, thus potentially reducing time and cost for the investigation of the system functioning. Moreover, in some cases the use of a model can open new possibilities for the evaluation of properties otherwise impossible to assess (i.e., not possible to empirically measure).
Different statistical, mathematical and computational techniques have been developed to test the behavior of a model, among others sensitivity analysis, a technique that studies how the output of a model is affected by its inputs.

This chapter provides a general overview of sensitivity analysis followed by the description of the most common techniques currently developed for the analysis of different kinds of properties of computational models. More specifically, in section 2.1 we provide a general picture on what perform sensitivity analysis means, then in section 2.2 we introduce the readers to the major technical details and the common issues present in sensitivity analysis application. Finally in sections 2.3 and 2.4 we focus on the application of sensitivity analysis to analyze respectively, quantitative and qualitative properties (e.g., output variance, dynamics modes).

## 2.1 Introduction

Sensitivity analysis (SA) is the study of how the uncertainty in the output of a model can be apportioned to different sources of uncertainties in its inputs [19]. Given the complexity of several real systems, scientists have to deal with models whose behavior depends from many input factors whose values are adjustable, this variability in the model definition often leads to large mutability in the models outcomes.

In fact, it is common that the properties of a system are subject to different sources of uncertainties deriving from the shortage or from the presence of incorrect information, such as the poor understanding of the functioning of the system or the presence of errors in the empirical measurements. Beside the uncertainties of some inputs, the value of some parameters can be also associated with the representation of an intrinsically variable property, as such, in order to simulate different states of a system different model configurations (or model parameterizations) are needed. Moreover, a models may need to cope with a potential intrinsic variability of the system behavior, as the occurrence of stochastic events, to do so, the application of stochastic simulation approaches can be required, thus introducing an additional source of variability in the model behavior.

SA aims at the identification of some quantifiable relation between the input values and the output of model. In practice, this can mean a number of different things, such as the computation of sensitivity indexes or the partitioning of space according to the model behavior, the common goal is in any case the identification of some measurable properties classifying the input-output relation of a model (Figure. 2.1).

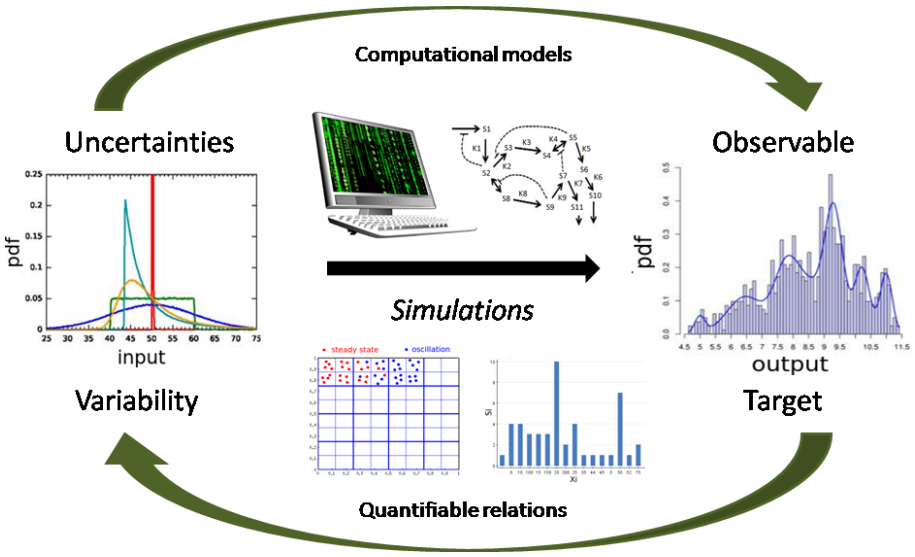


Figure 2.1: **Sensitivity analysis, a conceptual map**
Sensitivity analysis is designed to compute some properties explaining how the output variability of a model is affected by its inputs.

## 2.2 Methods

There are different different SA techniques which differ in the procedure followed to explore the input factor space and to compute the relation between the input values and the output of a model. The presence of many methodologies is a necessity depending from the existence of a variety of different types of problems; in fact, the sets of feasible model configurations (e.g., possible input factor values) along with the potential model behaviors (i.e., property

to be analyzed) can be extremely various from model to model. The aim of an analysis directs the SA class to choice while the simulation time limits the range of applicable SA methods.

As a consequence, what can be a profitable strategy in a specific case study may not be a good approach in a different context, for instance it can be computationally unfeasible (e.g., due to the presence of a higher dimensional input factor space), or does not provide the desired information on the model behavior (e.g., evaluate the peak value in a transient phase rather than the analysis of the steady state condition).

Therefore, SA is not a trivial task, indeed several issues have to be considered in order to develop an effective strategy to elucidate the input-output relation under analysis. First of all a systematic definition of the input factors and of the output property is required to evaluate a model behavior in accordance with the purpose of the analysis and the model characteristics. More precisely, the definition of the input factor space and of the target output depends from the level of uncertainty and variability of the different parameters, the level of knowledge of the model behavior (e.g., if the qualitative dynamics are known or not), and from the specific property that has to be analyzed (e.g., output variance rather than partial derivatives) [20].

The consideration of a particular parameter as input factor depends on several aspects, such as the availability of experimental measurements, the association with an unsteady characteristic, and results of previous SA [21].

Once the input factors are chosen it is necessary to define the range in which each input can change (i.e., the constraints of variations), taken together the input factors and the relative constraints represents the space of all the potential model configurations [22]. Noteworthy, the probability density function of each input factor within its constraints of variation can be different. Usually a normal distribution is used when an expected value is present and a uniform distribution when the distribution is unknown or there is not a more probable state [23]. However, any arbitrary continuous or discrete distribution of the input values may be used according to the specific case (e.g., a set of different integer numbers for a discrete variable). In effect, what ultimately matters is that the combinations of the input factor values represent potential states of the modeled system. It is important to remark that the distribution of values in turn affects the sampling of the input factor space since it associates different probabilities to different input values. In case input factors are not uniformly distributed, the best practice is to sample in the space of the quantiles and to obtain the inputs values using inverse cumulative distribution functions.

Once the input factor space has been defined, that is once the constraints and input distributions have been defined, different solutions can be adopted for the its exploration (i.e., selection of input factor values in which evaluate the model behavior), and in particular, two main SA classes can be defined, the local methods and the global methods. The

sensitivity measures calculated in a particular location of the input factor space are defined as local sensitivity methods [24], usually local SA is performed around a model configuration reproducing some experimental data. Instead, the average sensitivities calculated over the entire input factor space, that is taking into account the interactions among the input factors, are defined as global SA methods [25]. In general global methods are more informative and more computationally expensive compared with local methods.

In local SA the sampling of the input factor space is not a critical problem since it basically concerns the testing of model perturbations at a reference model configuration, therefore the issues are limited to the way in which it is possible to numerically compute derivatives. Instead, in global SA different techniques to explore the input factor space can be exploited, in particular it is possible to discern between two main categories, the one based on stratified sampling, in which the input factor space is discretized into a grid and the sampling must follow some rules, and the ones based on the generation of random and quasi-random numbers mapped over the input factor space [26]. The aim of the different sampling techniques is however always the same, optimize the exploration of the input factor space for the evaluation of the model behavior.

Regarding the selection of the target output to measure, it depends on the model property that needs to be assessed together with its potential variability. For instance, the target output could be a state variable in a given time, or a function describing the relation among some variables and representing a collective characteristic of the system, or it can be the time required to reach a given condition. In other words, also the selection of the target output is strongly dependent from the specific case study, and in particular it is possible to distinguish between conditions in which the target output represents either a continuous or a discrete graded property (quantitative ordinal SA), and the ones in which the target output can assume a set of different nominal states (qualitative nominal SA).

Noteworthy, in global SA the exploration of the input factor space and the computation of the sensitivity indexes are not independent tasks, in fact, as we will see, the computation of a particular sensitivity index requires that the model is simulated using set(s) of model configurations presenting specific correlations among the input factor values [23]. In the following of this chapter we focus on the description of global SA, first for the evaluation of graded properties and then for the analysis of nominal properties.

## 2.3 Quantitative ordinal sensitivity analysis

In quantitative SA the aim is to find a relation between the input values and an output property of a model that can be assessed by the evaluation of a variable whose values can change over a graded range of values (either in a discrete or continuous way). Indeed, the majority of the developed SA techniques have been developed to study how a property assuming progressive values is affected by the model inputs. In such cases, the most common SA techniques are based on the computation of sensitivity indexes mathematically quantifying how each input is responsible for the target output variation.

There exist different methodologies for the computation of sensitivity indexes, most of them have been developed for the analysis of deterministic models simulated by mean of the resolution of systems of ordinary differential equations (a kind of modeling particularly widespread in science). In general, it is possible to distinguish between methods based on the evaluation of derivatives or different quotients, named also elementary effects [27] (described in 2.3.1), the ones based on the analysis of output variance [28] (described in 2.3.2), and the ones based on the generation of meta-models or surrogate models representing the relation between inputs and the target output [29] (described in 2.3.3).

The different techniques differ in the computational cost required for the analysis (i.e., number of model simulations required), and in the level of the information that could be retrieved, usually there is a trade-off between the quality of the retrieved knowledge and the computational time required for the investigation [30].

### 2.3.1 Elementary effects

A seminal work about this class of SA techniques is the screening SA method proposed by Morris [31]. The Morris method is based on the evaluation of different quotients (defined also elementary effects in the context of SA). Although not necessarily, elementary effect methods ($EE_s$) are often applied for screening SA studies, that is, they are commonly designed to retrieve coarse information on the input factors influence over the output values using a relatively low number of model simulations.

The application of a screening $EE$ approach is usually aimed at the computation of unrefined sensitivity analysis measures, and the obtained indexes are in general used to filter the non-influential input factors or to partition the inputs into ranking sets according to the overall influences (e.g., with negligible, medium or pivotal role). In an $EE$ method the input factor space is explored following One At Time procedures (OAT), which means that only one input at a time is changed and the computation of the sensitivity indexes is based on the evaluation of the different quotients between couples of model configurations [32].

Given a model with $k$ independent input factors and an output $Y$ of interest:

$$Y = f(x_1, \ldots, x_k) \tag{2.1}$$

and considering two model configurations that differ only for the $i$-th input value, the elementary effect associated with the $i$-th input ($EE_i$) is defined as:

$$EE_i = \frac{[Y(x_1, x_2, \ldots, x_{i-1}, x_i + \Delta_i, \ldots, x_k) - Y(x_1, x_2, ..., x_k)]}{\Delta_i} \tag{2.2}$$

where $\Delta_i$ represents the relative variation of the $i$-th input within its constraints of variation (that is its dimension in the input factor space).

Once a set of elementary effects for each input factor are computed in different regions of the input factor space, it is possible to estimate the overall influence of an input considering the average and the standard deviation of the absolute elementary effects associated (usually referred as $\mu^*$ and $\sigma$). The use of absolute value rather than the actual elementary effect allows for random sampling obtained by either increasing or decreasing the input value (otherwise it would be necessary to arbitrarily fix the direction of variation). Moreover by taking the absolute values it is possible to avoid potential cancellation effects that may occur when an input differently affects the output (i.e., leads to its increase or its decrease) in different regions of the input factor space.
The $\mu^*$ of $EE_i$ represents the main effect of an input while the $\mu^*$ represents the effects due the its interactions with other input factors [33]. The ultimate goal of $\mu^*$ and $\sigma$ evaluation is to assess if an input effect is negligible, linear and additive, or nonlinear and/or involved in interactions with other factors.

Different approaches to sample the input factor space can be exploited and importantly the exploration of the input factor space may affect the computation of elementary effects. The original Morris method is based on the discretization of the input factor space $X^k$ (where $k$ is the number of inputs) into a p-level grid $\Omega$. Each input factor $x_i$ can assume a set of different values within $\Omega$, $p$ is conventionally chosen even and $\Delta$ is set to $p/2(p-1)$ [19], as such, the number of potential values of $x_i$ is $p^{k-1}[p - \Delta(p-1)]$ (Figure 2.2).

While the Morris method has been proved to provide good estimation of screening sensitivity indexes, the main drawback is indeed associated with the way in which the input factor space is sampled [34]. In fact, as the number of input factors increases the number of model configurations to consider grows in an exponential way, eventually leading to a prohibitive computational cost when a large number of input factors need to be evaluated.

As a consequence, a number of different approaches have been developed to tackle the issue of the computational costs. A refined OAT sampling to reduce the number of model configurations required to compute the sensitivity indexes was proposed by Campolongo et al. [35]. The sampling strategy is based on the generations of trajectories within the

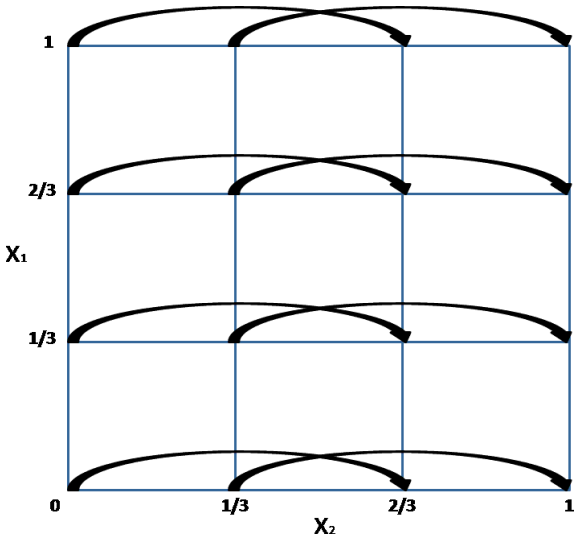


Figure 2.2: **Morris sampling**

Representation of a four-level grid (p=4) in the two-dimensional input space (k = 2) with $\Delta$ of 2/3. The arrows identify the eight points needed to estimate the elementary effects relative to factor $x_1$.

input factor space $X^k$ and the following selection of the trajectories that better explore the space.

The sampling starts from a base vector $\vec{x^*}$ randomly selected in $X^k$, then moving by $k$ steps in each one of which the value of an $x_i$ input, which has not been already modified, is either increased or decreased by $\Delta$. This leads to the generation of a trajectory with $(k+1)$ sampling points $\vec{x^1}$,$\vec{x^2}$,$\ldots$,$\vec{x^{k+1}}$. A trajectory has the key property that two consecutive points differ in only one component (and thus it is suited for the computation of the elementary effects), and moreover since any value of the base vector $\vec{x^*}$ has been selected once to be modified by $\Delta$, all the input space $\Omega$ is covered (Figure 2.3).
Once a pool of $N$ trajectories are generated, it is possible to select the subset of $r$ trajectories among the $\binom{N}{r}$ with the maximum spread within $X^k$. The spread among a subset of $r$ trajectories is given by the squared sum of all the $d_{ml}$ distances generated by all the possible couples within the set, while the distance $D$ between two different trajectories $m$ and $l$ in $X^k$ is given by their square distance:

$$d_{ml} = \sum_{i=1}^{k+1} \sum_{j=1}^{k+1} \sqrt{\sum_{z=1}^{k} [X_z^{(i)}(m) - X_z^{(j)}(l)]} \tag{2.3}$$

In effect, the distance $d_{ml}$ represents the sum of the geometric distance between all the pairs of points of the two trajectories under analysis With respect to the original Morris sampling, the generations of trajectories is able to reduce the number of model evaluations required [35]. However, for large input factor space, the computation of all the possible trajectories combinations can lead to a combinatorial explosion, thus making unfeasible

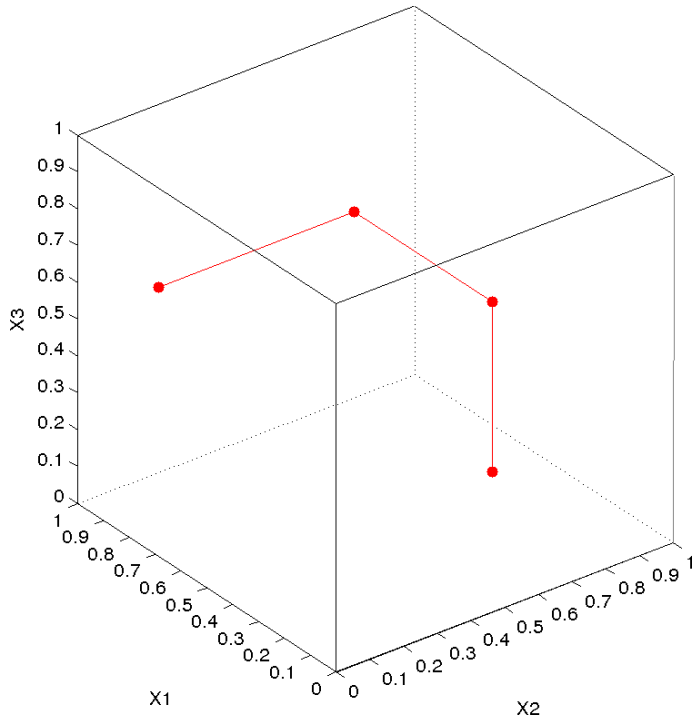


Figure 2.3: **Trajectory sampling**

Example of a trajectory in a case with a three-dimensional input factor space.

the application of the method.

A possible OAT sampling approach avoiding the problem of the combinatorial explosion is the radial like exploration. A radial sampling is based on the generation of two random vectors whose values can me mapped in the input factor space [36]. Differently from the trajectory generation, in which after each step one keeps moving away from the original point, in radial design one goes back to the first point (named reference or base point) after each movement (Figure 2.4). Since the number of changes correspond to the input number $k$ plus the reference point, the total cost is the same of trajectory sampling, that is $k+1$ model configurations need to be tested per each reference model configuration.

Noteworthy, the limit of an elementary effect (i.e., different quotient) as $\Delta i$ becomes infinitesimally small correspond to the computation of the derivative:

$$\lim_{\Delta x_i \to 0} \frac{\Delta Y}{\Delta x_i} = \frac{dY}{dx_i} = f'(x_i) \tag{2.4}$$

therefore, the evaluation of elementary effects considering a significant low value of $\Delta i$ is a numerical way to consider the partial derivative of a given input factor with respect to the target output. As a consequence, the computation of a set of derivatives within the input factor space is another way to compute the sensitivity indexes, and similarly to elementary effects, the overall influence can be computed averaging the local sensitivity values.

The exploitation of derivatives based approach can be extremely powerful in all the

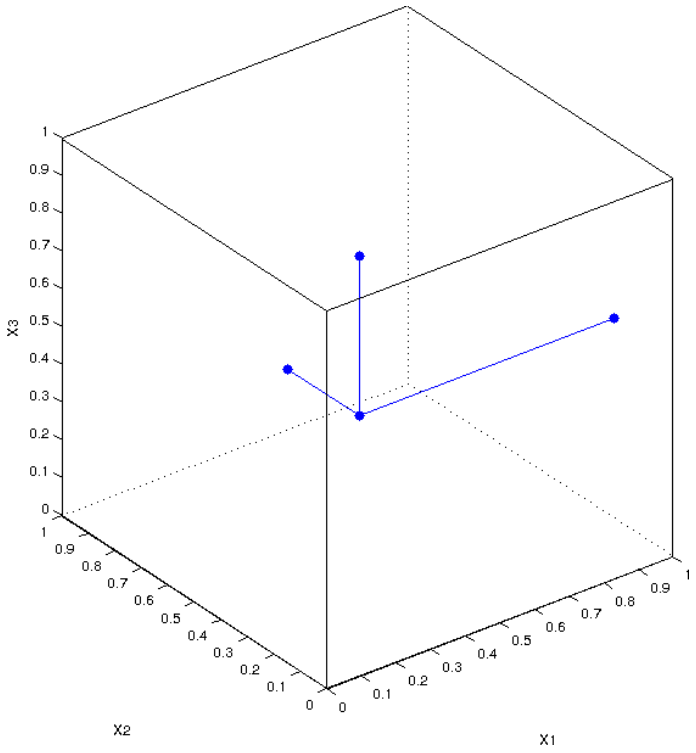


Figure 2.4: **Radial sampling**
Example of a radial-like sampling in a case with three-dimensional input factor space.

cases in which a model can be considered a differentiable function, which is indeed the condition of several systems represented by mean of system of ordinary differential equations. In such cases, a model is seen as a differentiable function $f(X_i)$ where $X_i = x_1, \ldots, x_k$ is the vector of input factors free to move across the input factor space, defined as unit hypercube $X^k$ $(0 \leq x_i \leq 1, i = 1, \ldots k)$.
The local sensitivity measures are based on partial derivatives in the form:

$$EE_i(X^*) = \frac{\partial f}{\partial x_i} \tag{2.5}$$

where $X^*$ represents a nominal point in the input factor space. Since the local sensitivities may change as different nominal points are considered, by averaging $EE_i(X^*)$ over the input space $X^k$ global based sensitivity measures can be retrieved, as proposed by Kucherenko et al. [27].

As for the finite elementary effects, it is possible to consider the average and the standard deviation of the derivatives values over the input space, this lead to the computation of the following quantities:

$$M_i^* = \int_{X^k} = |EE_i| dx_i \qquad \sigma_i = \sqrt{\int_{X^k} (EE_i - M_i^*)^2 dx} \tag{2.6}$$

which can be used as overall sensitivity indexes. Likewise in screening $EE$, $M_i^*$ represents the main effect while $\sigma_i$ depends from interactions among inputs.

Both the computation of finite elementary effects and of derivative based global sensitivity measures have been proved to provide important information on how input factors affect the model output [37, 38].

### 2.3.2 Output variance decomposition

As the name states, variance SA methods are based on the analysis of how the variance of a target output is affected by the input factor values [35, 39]. Variance SA methods are well suited for quantitative SA, and importantly they can deal with non additive and non linear models [40].

Given a model $Y = f\{x_1, \ldots, x_k\}$ with $k$ uncertain input factors, the output variance $V(Y)$ can be decomposed (by mean of the so called ANOVA decomposition) into terms depending on single factors and on their interactions. The idea behind variance-based measures is that the total output variance $V(Y)$ for a model with $k$ input factors can be decomposed as:

$$V(Y) = \sum_{i=1}^{k} V_i + \sum_{i=1}^{k} \sum_{j>i}^{k} V_{ij} + \cdots + V_{1,\ldots,k} \tag{2.7}$$

where:

$$V_i = V(E(Y|X_i)) \tag{2.8}$$

while,

$$V_{ij} = V(E(Y|X_{ij})) - V_i - V_j \tag{2.9}$$

and so on.

The number of terms of the ANOVA decomposition grows exponentially with respect to the input factors number (i.e., $2^k$). As a consequence, only a limited number of input factors can be considered, otherwise the analysis becomes unfeasible (the limit is adjustable depending on the simulation time, however in general the analysis is limited to less than ten inputs).

Moreover, to further reduce the computational cost, usually not all the terms of ANOVA decomposition are evaluated, instead only two sets of $k$ indexes are commonly computed: the first order and the total order effect [41].

The first order effect of a generic input factor $x_i$ is defined as:

$$V_{X_i}(E_{X_{\sim i}}(Y|X_i)) \tag{2.10}$$

where $X_i$ is the $i$-th input and $X_{\sim i}$ denotes the matrix representing all the input factors but $i$-th. The inner expectation operator is due to the fact that average of $Y$ is taken over

all the possible values of $X_{\sim i}$ (within the associated constrained defined for each input factor) while keeping $x_i$ fixed; the outer variance is taken over all possible values of $x_i$ ($X_i$).
The associated sensitivity measure, named first order sensitivity coefficient (acronym $S_i$) is defined as:

$$S_i = \frac{V_{X_i}(E_{X_{\sim i}}(Y|X_i))}{V(Y)} \tag{2.11}$$

The first order sensitivity index states how much the output variance $V(Y)$ decreases as a consequence of setting of $x_i$ to a specific value, in other words, $S_i$ indicates how much $V(Y)$ could be reduced if we were able to learn the true value of $x_i$.
Since $V(Y)$ can have values in between 0 and 1, $S_i$ may range from 0 when $x_i$ does not affect at all on the output variance, to 1 when all the input variance rely on the $x_i$ value (i.e., changing the other input factor values does not affect $V(Y)$).

The total order effect of a generic input factor $x_i$ is defined as:

$$E_{X_{\sim i}}(V_{X_i}(Y|X_{\sim i})) = V(Y) - V_{X_i}(E_{X_i}(Y|X_{\sim i})) \tag{2.12}$$

and it measures the expected variance of $V(Y)$ that would be left, on average, when beside from $x_i$, which is let free to vary over its allowed range of variation, all the other input factors are fixed. The associated sensitivity measure, named total sensitivity index (acronym $S_{T_i}$), is defined as:

$$S_{T_i} = \frac{E_{X_{\sim i}}(V_{X_i}(Y|X_{\sim i}))}{V(Y)} = 1 - \frac{V_{X_{\sim i}}(E_{X_i}(Y|X_{\sim i}))}{V(Y)} \tag{2.13}$$

and it presents the expected reduction of variance $V(Y)$ that would be obtained if all inputs but $x_i$ could be fixed to their true value. $S_{T_i}$ is a measure of the overall effect of an input factor on the output variance (i.e., inclusive of interactions).
$S_{T_i}$ as $S_i$ has a bottom value of 0, instead, differently from $S_i$, $S_{T_i}$ may exceeds 1 due to the presence of interaction effects.

Regarding the interpretation of variance based sensitivity indexes, a high $S_i$ is a necessary and sufficient condition to state that $x_i$ has a big influence on the model output, therefore it is a good coefficient for ranking the most influent input factors over the output variance. Instead, a low $S_i$ is a necessary but insufficient condition to state that $x_i$ is not influential, in fact it could be involved in interactions of high orders affecting the output variance. Conversely, a low $S_{T_i}$ is a necessary and sufficient condition to state that $x_i$ is not influential and therefore it is a good coefficient to look at for factor fixing (identify the factors whose values could be fixed without affecting the model output). Instead, a high $S_{T_i}$ proves that an input greatly impact the output, but it does not permit to discern if the influence depends specifically on its value or arises from interactions with other input

factors.

From the mathematical definition, $S_{T_i}$ can be equal or greater than $S_i$, and the difference between the two indexes is a marker of how much $x_i$ is involved in interactions. In fact, for a complete additive model the sum of all the first order effects must sum to one $\sum_{i=1}^{k} S_i = 1$, while the value of the difference $1 - \sum_{i=1}^{k} S_i$ indicates how much the model output variance depends from interactions among input factors rather than the values of specific inputs.
The computation of $S_i$ and $S_{T_i}$ allows model users to have a fairly good description of the effects of the analyzed inputs at a reasonable computational cost [41].

A number of approaches to compute both $S_i$ and $S_{T_i}$ have been developed, in general, the computation of indexes is obtained by a combination between specific sampling design and application of proper formulas (theoretically validated) evaluating output variance. The evolution of the different approaches has been guided by the aim to reduce the number of model simulations required to estimate the true indexes [42].

At first sight, the straightforward approach to estimate the conditional variance $V_{X_i}(E_{X_i}(Y|X_{\sim i}))$ and $E_{X_{\sim i}}(V_{X_i}(Y|X_{\sim i}))$ would be the computation of the multidimensional integrals over the input space. In other words, this implies the use of a double Monte Carlo loop, one in which $x_i$ is fixed to different values and one in which $X_{\sim i}$ is set to different configurations. However, for orthogonal input values (i.e., independent input factors), which is a typical condition of mechanistic models of biochemical systems, the computation can be greatly accelerated, and different sampling methods aimed at the reduction of computational cost have been proposed [41].

Among the sampling techniques developed so far, particularly significant is the one proposed by Saltelli et al. in [43]. The method exploits the same model configurations for the simultaneous computation of $S_i$ and $S_{T_i}$, thus further reducing the computational cost.

The sampling is based on the generation of two independent sampling matrices $A$ and $B$, defined respectively sampling and re-sampling matrix, with $a_{ij}$ and $b_{ij}$ as generic elements, the index $j$ run from 1 to the number of inputs factor $k$, while the index $i$ run from 1 to $n$, where $n$ is the configuration number. Therefore, matrix rows correspond to model configurations, while the columns correspond to the set of values assumed by each specific input factor within all the model configurations.
Subsequently, a set of $k$ matrices (one per input factor) are generated so that the $i$-th matrix, named $A_B^i$ (associated with $i$-th input factor), is formed by taking all the columns from A with the exception of the $i$-th column, which is instead taken from B. As a consequence, for a given input $i$-th, the model configurations represented by $A$ and $A_B^i$ matrices differ only for the $i$-th values, this setting is consistent with the definition of $S_i$ which aims at the evaluation of the output variance variation deriving from the setting of the analyzed input factor to a specific value (not yet determined).

On the other hand, the model configurations represented by $B$ and $A_B^i$ matrices have in common only $i$-th input values, this setting is consistent with the definition of $S_{T_i}$ which aims at the evaluation of the remained output variance when all the input factors but $i$-th, are fixed.

The best practice to derive the initial $A$ and $B$ matrices is from the generation of a quasi-random sequences, which are indeed low discrepancy sequences providing equidistibuted values in each dimensions, this in turn assures the maximum exploration of the space [44]. Importantly, the quasi-random sequence are formed by numbers in the interval $[0, 1]$, therefore each value has to be properly mapped to the constraint associated with each input. For example, in case of narrow constraints of variation allowed (i.e., within the same order of magnitude) a uniform mapping can be suggested, while for larger constraints (i.e., few to several order of magnitudes) a logarithmic mapping can be the proper choice (this is to assure the testing all over the range rather than oversampling higher values and subsampling lower values).

To this purpose, it is possible to generate a quasi-random matrix $Q$ of size $(n, 2k)$, then derive the $A$ matrix from the left half of $Q$, and the $B$ matrix from the right half of $Q$. As such, the sampling matrix $A$ is so defined:

$$A = \begin{bmatrix} x_1^1 & \dots & x_i^1 & \dots & x_k^1 \\ \vdots & & \vdots & & \vdots \\ x_1^j & \dots & x_i^j & \dots & x_k^i \\ \vdots & & \vdots & & \vdots \\ x_1^n & \dots & x_i^n & \dots & x_k^n \end{bmatrix}$$

instead, the resampling matrix is defined as:

$$B = \begin{bmatrix} x_{k+1}^1 & \dots & x_{k+i}^1 & \dots & x_{2k}^1 \\ \vdots & & \vdots & & \dots \\ x_{k+1}^j & \dots & x_{k+i}^j & \dots & x_{2k}^i \\ \vdots & & \vdots & & \vdots \\ x_{k+1}^n & \dots & x_{k+i}^n & \dots & x_{k2}^n \end{bmatrix}$$

while, a generic matrix $A_B^i$ is so derived:

$$A_B^i = \begin{bmatrix} x_1^1 & \dots & x_{k+i}^1 & \dots & x_k^1 \\ \vdots & & \vdots & & \vdots \\ x_1^j & \dots & x_{k+i}^j & \dots & x_k^i \\ \vdots & & \vdots & & \vdots \\ x_1^n & \dots & x_{k+i}^n & \dots & x_k^n \end{bmatrix}$$

where, $x_i^j$ represents the value of the $i$-th input in the $j$-th model configuration.

Once a model have been simulated, and the target output have been computed in all the model configurations, it is possible to estimate the output variance using the retrieved output vectors. A number of different formulates have been proposed to compute the sensitivity indexes from $Y(A)$, $Y(B)$ and $Y(A_B^i)$ [43], thus for a total cost of $k(n+2)$ rather than the $k^n$ required by a Monte Carlo approach.
Among others, the Jansen formulas have been proved to be the more efficient estimators of variance based sensitivity indexes [45].
According to the Jansen estimators, the first order sensitivity index $S_i$ is computed as:

$$V_{X_i}(E_{X_i}(Y|X_{\sim i})) = V(Y) - \frac{1}{2N}\sum_{j=1}^{n}(Y(B)_j - Y(A_B^i)_j)^2 \tag{2.14}$$

while, the total order sensitivity index is computed as:

$$E_{X_{\sim i}}(V_{X_i}(Y|X_{\sim i})) = \frac{1}{2N}\sum_{j=1}^{n}(Y(A)_j - Y(A_B^i)_j)^2 \tag{2.15}$$

Since usually is not possible to compute the true values of $S_i$ and $S_{T_i}$ (i.e., they can not be theoretically validated or the computation time would be excessive), the value of $n$ is commonly selected to ensure the convergence of the sensitivity indexes [21]. Because of that, in practice for large models the computation of variance based sensitivity indexes is often based on the careful evaluation of the indexes trend as the number of considered model configurations increases.

**Which method use for the computation of sensitivity indexes?**

A sensitivity index is an index to assess the relevance of an input on determining the model output value, and since its value depends on the property that is evaluated (e.g., variance, derivatives) there is no a unique correct answer. Instead, it is more proper to say that there exist different ways to study the input-output relation of a model, and the more valuable

depends by the specific case study.

In fact, as different methodologies are looking at different properties, it is impossible to assess that one method theoretically outperforms another one (i.e., provides more realistic indexes). As such, the choice of the SA technique to apply in a given condition is often motivated by the computational cost. In turn, different aspects affect the computational cost, like the size of the input factor space (i.e., the number of input factors and the associated constraints of variations), and the property to be measured (e.g., derivatives vs. variance).

Therefore, it can be suggested to integrate different methods to have a more comprehensive view of the model behavior, possibly starting from less computational expensive methods (e.g., screening elementary effect), on a higher input space, and then focusing on detailed but more expensive approaches for study the effects of the most relevant inputs (e.g., variance based methods, possibly including interactions evaluation).

### 2.3.3 Surrogate models

A different class of SA methods is constituted by development of surrogate models describing the input-output interaction in a simpler way with respect to a more complex original system [19]. The use of surrogate models is not aimed at the computation of specific sensitivity indexes but rather at the reconstruction of the relationship between some input factors and the model behavior [46].

Basically, given a model $Y = f(x_1, \ldots, x_k)$, a surrogate model is a mathematical formulation able to directly describe the input-output relation. The analysis of the surrogate model can disclose effects hidden in the overall system (e.g., a linear or quadratic relationship between a given input and a given model property).
Furthermore, since surrogate models are De-facto meta-models providing an approximation of an original system, they are usually computationally less demanding in comparison to the original ones. As a consequence, surrogate models can also be used to speed up the analysis by faster simulations and thus reducing the analytical time to study the system behavior [47].

From a theoretical point of view, the problem of the generation of a surrogate models is a problem of approximation. There is a vast literature on this topic, in general it is possible to distinguish between the local approximation methods and global approximation techniques. In the local methods the values of the partial derivatives is taken at a base point $X_0$ and a function that matches the property of $f$ at $x_0$ in the nearby region is derived by calculating the Taylor series) [48]. Instead, the global techniques are usually based on the exploitation of interpolation/regression methods representing the relation between the output and the input within a given range of input values [49].

The difference between the interpolation and the regression methods is that, in the

former ones, given a set of data $P = p_1, \ldots, p_n$, the aim is the identification of a function that match the exactly the original values, instead in the latter ones, the aim is the identification a function mimicking the trend presents in the data (Figure 2.5).

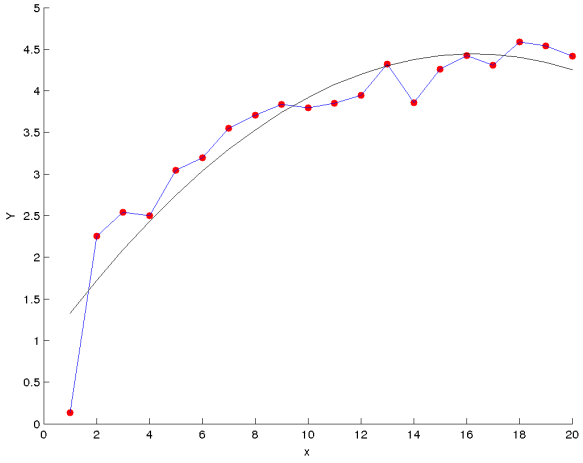


Figure 2.5: **Interpolation vs. regression**
In red dots the original data, in blue line piecewise linear interpolation, in black line a regression curve.

In the interpolation methods a piecewise approach is usually used when the input space is large or the output shape is complex. In such cases, instead of generating a specific function describing the overall trend, different functions are used to "connect" the consecutive data points, thus allowing the pairing between the data and surrogate model values. On the other hand, regression approaches are usually based on the generation of a distinct function mimicking the overall trend, as consequence the ability to depict the data is lower but the system is more robust (e.g., less prone to over-fitting, and able to deal with stochasticity).
Regression approaches are usually preferred over interpolation methods due to the capability to detect trends hidden by noise and to represent the relation in closed function form. However, there are circumstances in which interpolation is the only feasible option, for instance when the complexity of the data distribution makes impossible the description by mean of a function, or when the function would be to computationally expensive to compute (e.g., it would require a high polynomial degree).

The development of a surrogate model can elicit relationships hidden in the general system, nonetheless for a complex system the interactions can be variable according to the state of the system, that is different areas of the input factor space can be associated with different input-output relations (Figure 2.6). As such, the value of an input can significantly affect the output in a given state (i.e., model configuration), while it can be negligible in another state.

The use of surrogate models has the limit to not provide some sensitivity index for

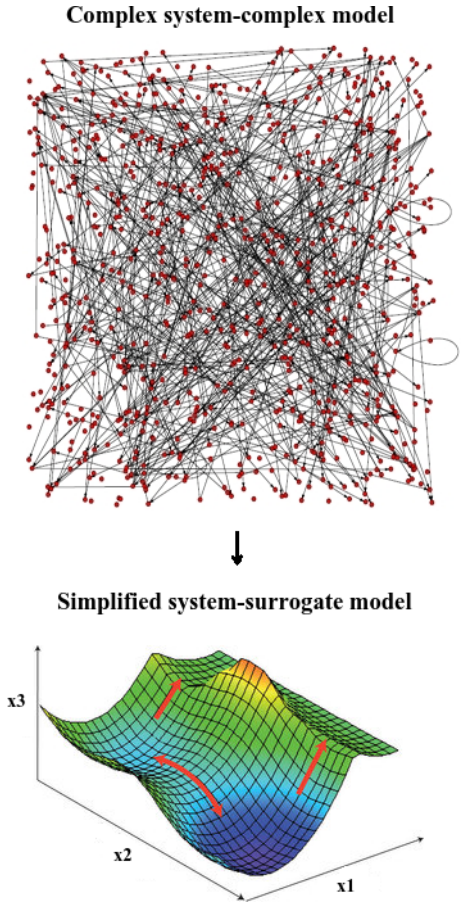


Figure 2.6: **Surrogate model**

Starting from a complex model by mean of a surrogate model is possible to study specific input-output relationship. The red curves indicate the different trends that can be observed in different regions of the input factor space.

direct comparison of inputs effects on the model output. However, differently from the classic methodologies employed in SA, a surrogate model is able to provide qualitative information on the input-output relation (e.g., answer the question if the increase of an input leads to a decrease or increase of the output). A surrogate model may be used for quantitative analysis as well, that is to estimate the expected output in response to a given input modification. The ability to provide an approximated overall picture of a system behavior with a reduced computational demand allows the analysis of systems that would be otherwise useless for practice purposes due to the excessive computational time required by models simulations. This property makes surrogate models particularly suited for experimental design of complex systems in which the high number of parameters involved makes impossible not only the computational simulation but also the experimental analysis of all the different variables. In this context, exploit surrogate models helps in the planning of experiments aimed at obtaining the maximum information on alternative hypothesis about a system behavior [49].

## 2.4 Qualitative nominal sensitivity analysis

In the thesis, with qualitative nominal SA we mean the study of how the input factor values affect the qualitative behavior of a system. In other words, we consider as output target of qualitative SA a property that is not coupled to graded values but instead that can be associated with different nominal states.

While for the classic quantitative SA different techniques have been already defined in a structured way, that is it is know their implementation and the methods have been theoretically validated (at least for deterministic model), qualitative SA is still a not well-defined research context. In the thesis, we focus in particular to the analysis of the relation between the input configuration and the dynamics of a model, that is to the study of how the input factor values lead a model to the attainment of different behavioral dynamics (e.g., oscillations vs. steady state).

In this context, some problems are analogous to quantitative SA (e.g., exploration input factor space), however, the major issue in qualitative SA is related to the behavioral identification [50]. In fact, the classification of a behavior is often an arbitrary choice, moreover some qualitative behavior may be easily classified in an automatic way (e.g., increase or decrease of a property), while other dynamics trends can be difficult to discern (e.g., random noise and stochastic oscillations).

Therefore, the output target of qualitative SA differs from quantitative SA since it is not a continuous or discrete ordinal property (i.e., with graded values that can be ranked), instead it can be classified into discrete nominal set of potential states.

As a consequence, the aim is not the computation of some sensitivity indexes ranking the relevance of input factors over the model output, but rather to draw a connection between input factor space and system behavior (e.g., dynamics trends).

The mapping between input-output encompasses two main issues, from the one hand find a numerical property which can be used to discern different model behavior, from the other hand empirically sample many input values combinations to find the connection between the input factor space and output mode.

A critical issue is to find measurable properties at low level definition (i.e., computable variable) that can be associated with different behavioral patterns at high level definition (i.e. different modes), thus allowing the classification of dynamics. In fact, according to the specific characteristics of output trends, different strategies can be adopted for the dynamics classification, such as equilibrium levels for steady states, periods and amplitudes for oscillations (Figure 2.7).

The mapping between the input space and the output dynamics can be formalized as the identification of sets of input factor values for which the system does (or does not) reach a given set of dynamics, and to the following partitioning of the input factor space into

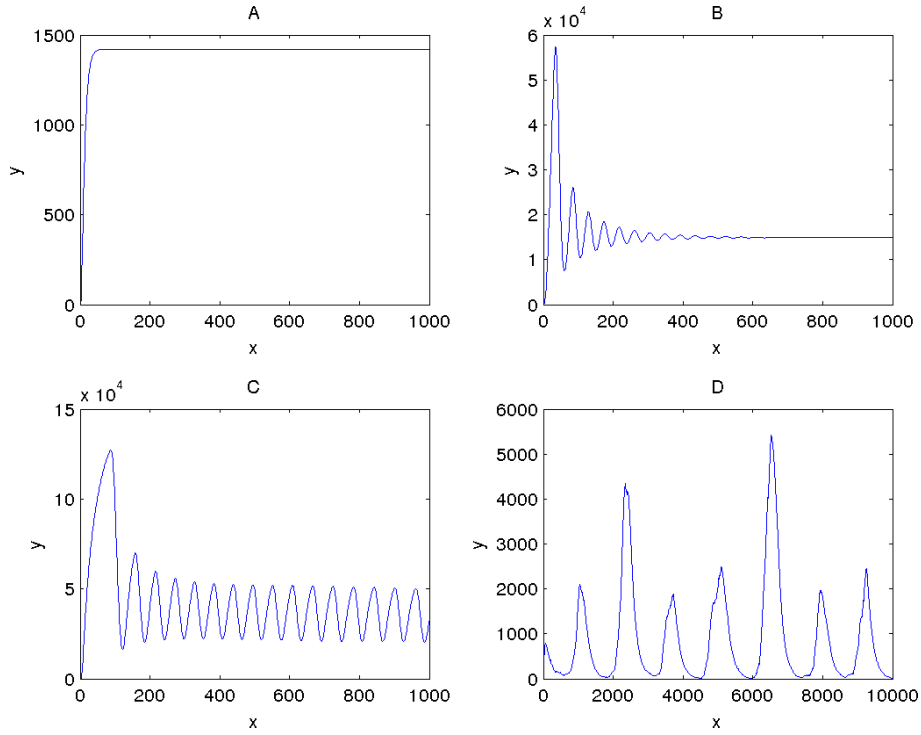


Figure 2.7: **Dynamics trends**

Some dynamic trends that a generic model can attain, steady state (A), damped oscillation (B), stable oscillation regime (C) and stochastic oscillation regime (D).

different regions according to the conditions which occur in different areas.

A common approach exploited by the algorithms for the input-output mapping employ Monte Carlo sampling strategies to identify the subsets in the input factor space which satisfy a specific property state. The strategy which is usually deployed by mapping algorithms is to iteratively find subregions in the input factor space in which a condition is verified [51, 52]. The expected result of the overall procedure is to partition the input factor space into larger regions where the satisfaction of the property is more robust (i.e., stable) and smaller regions where is more probe to violation (i.e., unstable), thus identifying the boundaries between different discrete property states. If these states are associated with different dynamics, the procedure ultimately leads to the identification of regions within the input factor space associated with different system dynamics.

# Chapter 3

# Implementation of sensitivity analysis techniques

This part of the thesis concerns the exploitation of SA techniques for the analysis of input-output relation in a model (using different biological models as case studies) .
In section 3.1 we describe how to address a common problem in the analysis of computational model behavior, that is how a given property is affected by the input values. Regarding this issue, we proposed a pipeline based on the integration of different SA methods in order to first detect the key factors involved in the regulation of the analyzed property, and then quantitatively characterize the relation between the pivotal input factors and the property values. In our test, we identified how the metabolic activity (i.e., the target property) is regulated by enzymatic activities (i.e., the input factors).
Section 3.2 is focused on the analysis of stochastic output, in particular, through the analysis of a stochastic model of bacterial chemotaxis (used as a case study), we show how it is possible to extend the common SA methodologies developed in the context of deterministic systems, so to be able to deal also with a noisy output.
In section 3.3 we propose a potential approach, based on the application of SA, to face a common issue in the development of strategies aimed at the control of a system, that is how to specifically act on the system behavior only in certain situations for determined purposes. Our proposal, by comparing condition-specific SA indexes is able to obtain a derived index representing the level of specificity of an input modification over an output target (i.e., according to the responses in different system conditions).
Section 3.4 regards the analysis of spatial effects, an aspect that has been often neglected in the development and analysis of computational models simulating biological systems. More specifically, we built a simplified model of molecules diffusion processes in a cell nucleus. Subsequently we acted modifying the initial spatial configuration (i.e., initial molecules positions), and then through the analysis of the resulting model behaviors we were able to identify a relevant connection between spatial configuration and dynamics of the system.

Finally, in section 3.5 we considered the effects of random perturbations over time on a system. The analysis has been performed exploiting a modeling framework explicitly able to consider changing environmental perturbations on a system.

## 3.1 Characterization of a property regulation: a metabolic case

Metabolism is the set of biochemical reactions that sustain the cellular activity of a living organism [53]. The study of metabolic processes encompasses several problems which are associated with the inherent complexity of metabolism and with technical issues in the experimental analysis (e.g., need to evaluate many elements, high experimental cost).

Given the importance of metabolism, metabolic pathways have been well characterized and not by chance are among the first biological processes described with mathematical models [54]. Nowadays, several comprehensive models about different metabolic processes are available in literature (i.e., can be freely used by the scientific community), and thus the development of computational techniques aimed at the analysis of the behavior of those models, can lead to practical advances in prediction and control of biochemical systems in biomedical or bioengineering research [55].

However, understanding the functioning of metabolism is a very challenging task since the different pathways are tightly connected, as such, several regulatory feedbacks influence metabolic activity [56]. In this context, the exploitation of mathematical models able to simultaneously simulate different mechanisms enable proper representations of biochemical pathways behavior, may help in unraveling the control processes underlie biochemical networks dynamics.

Metabolic pathways are regulated by activities of enzymes, which are biomolecules promoting biochemical reactions present in biochemical pathways. As a consequence, the majority of the studies in metabolic analysis are aimed at the identification of how enzymes affect the fluxes of the different biochemical transformations [57].

The other molecular actors of metabolism are metabolites, which are the bio-molecules involved in biochemical pathways and which are present in plentiful amounts (i.e., several molecules number). As such, the mainstream simulation framework for metabolic pathways is the exploitation of system of ordinary differential equations [58]. In general, the variation over time of a metabolite concentration $m_i$ is expressed by a differential equation in a form like:

$$dm_i/dt = \frac{\sum_{k=1}^{g}(S_{ik}J_k(x(t),c))}{V} \tag{3.1}$$

where, $V$ is the volume where an ensemble of $g$ reactions occur, $S_{ik}$ is the stoichiometric coefficient indicating the number of moles of metabolite $i$ that are consumed or produced

by reaction $k$, and $J$ is the flux expression of reaction $k$, which contain kinetic parameter $c$ and should be written according to the reaction kinetic mechanism or using a biochemical plausible approximation [59].

For such kinds of models, the exploitation of SA between the kinetic rates and the fluxes can be a profitable approach to study how enzyme activities influences a particular metabolic phenotype (i.e., a flux representing a specific system state). For these kinds of analysis, a common practice is the application of Metabolic Control Analysis (MCA), which is a SA technique able to quantify how the control of fluxes and metabolite concentrations in a metabolic network is distributed among the different enzymes that constitute the pathway [60]. MCA essentially computes control coefficients which are calculated on the basis of one-at-time (infinitesimally) small parameter changes in the linearized system around a stable steady state. Hence, MCA is a local SA method that does not cope with nonlinearities, and that it is not able to evaluate interactions among reaction rates, neglecting potential important distributed regulatory effects.

Instead, an overall analysis capable to deal with interaction effects could be carried on exploiting global SA methods [25]. However, the application of global SA on metabolic models is not trivial, because many methods are available, each one with its peculiar features, advantages and disadvantages. Each SA technique is aimed at studying a particular issue and to gain information related to a specific property of the model behavior. For instance, from the one hand, screening SA techniques even though are powerful computational tools to detect the pivotal factors affecting a given property, they do not provide any information about the quantitative relation between the input and the output. From the other hand, extensive methods are often unfeasible due to the large input factor space (i.e., to many input values combinations need to be considered).

To overcome the barriers in the application of global SA method on complex models (e.g., metabolic models), possibly characterized by a large input factor space and nonlinear behavior, we developed a pipeline based on the integration of different SA methods to describe the input-output relation [61]. The proposed pipeline is based on the application of Regionalized SA (RSA), which is a screening SA technique, to first identify the reaction rates affecting the property under analysis and then on the exploitation of Responses Surface Methodology (RSM), to reconstruct the relation between the pivotal factors and the analyzed property.

More in detail, RSA is a technique suited at identifying the regulatory factors of a qualitative property of a system that can be expressed as potential binary combination of states. RSA is based on the simulation of a model using different configurations (i.e., different input factor values), and on the classification of the model output according to the fact that a given condition is fulfilled or not [62]. The idea is that given a sufficient high number of model configurations, it is possible to compare the values distribution of

an input factor within the set of model configurations that lead to the fulfillment of the property and within the set of model configurations that failed in the realization of the condition (the sets are complementary). So doing, a couple of similar distribution is expected if an input factor is not relevant in determining the property state, while a dissimilar distribution is expected if the factor is unrelated to the regulation of the analyzed property. Specifically, RSA is based on the computation of an index measuring the similarity between couple of value distributions, and to the subsequent qualitative ranking of input factors relevance according to the values of the computed index. RSA has been proved to be particularly efficient in classifying the input factor influences that lead a model to a particular behavior, also in the case of complex non-linear models [63].

RSM instead is able to explore the relationship between different explanatory variables and one or more response variables. RSM has been proved to be a powerful tool for understanding and controlling the behavior of a complex system. In particular, RSM can be used to find the optimal combination of a set of influencing parameters, able to minimize or maximize a specific property of the system [64]. RSM involves the selection of ranges and distributions for each input factor, the development of an experimental design defining the combination of input factor values on which evaluate the model, the simulation of the model to retrieve the output values, and the reconstruction of a response surface approximation. RSM is able to provide a deeper knowledge on how the target output is affected by the input factors at the cost of a high computational demand. In fact, since the number of combinations increases exponentially with the number of input factors, RSM becomes de-facto not feasible for larger input factor space.

In the developed framework, in order to take advantage of the potentiality of RSM to explain the input-output relation while keeping low the computational cost, RSM has been coupled with RSA. More specifically, the proposed pipeline is based on the application of RSA to first qualitatively screening the most important input factors, and then on the exploitation of RSA to quantitatively characterize the output response to the variation of the pivotal input factors.

In our applicative context, the input consists in reaction rates, while any metabolic phenotype (i.e, measurable property) can be considered as output.
We applied the pipeline on the model of mitochondrial metabolism developed by Bazil et al. [65], which represents all the major steps in mitochondrial bio-energetics (Figure 3.1). The model consists of 73 DAEs (65 non-linear ODEs and 8 algebraic expressions) and was corroborated using several independent datasets containing information about the concentration levels. The model includes 42 adjustable parameters, of which 28 are enzymes maximum rates, the remaining parameters regard other biochemical mechanisms (e.g., permeability coefficients and binding constants).

In our study as target output has been investigated a critical aspect of cancer cells,

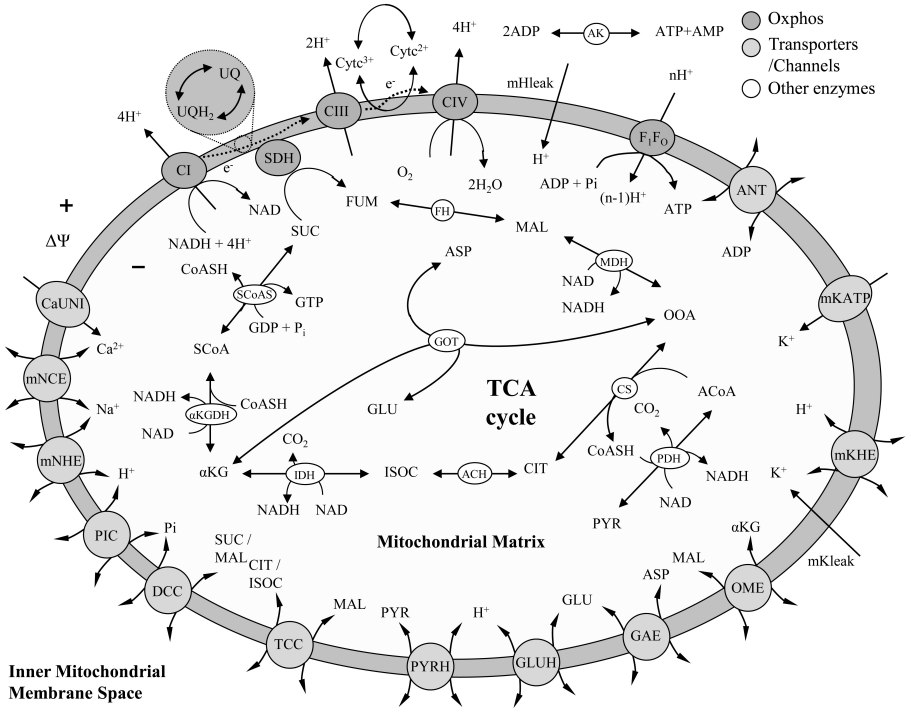


Figure 3.1: **Model of mitochondrial metabolism**

Biochemical network simulated by the model. The figure is taken from [65].

namely the flux of citrate between cytosol and mitochondrion. The citrate flux according to energetic and metabolic condition can go in either the directions (Figure 3.2). Since the flux is influenced not only by the enzymatic activities but also from the initial concentration of citrate in the cytosol ([citrate)], the analysis was performed considering three different initial state, ([citrate]=$(1e^{-3}M, 1e^{-4}M, 1e^{-5}M)$), and computing for each case the sensitivity indexes.

In our specific application, the set of input factors is constituted by the reaction rates, which in turn is related with the enzyme activity, More specifically, the maximum rate of an enzyme is proportional to enzyme concentrations and scale the reaction rates $J_i$ (or flux expression) of the corresponding enzyme $J_i = v_i f(x(t), c)$. The input factors set consists of the maximum rates values ($v$), which is a sub-vector of the model parameters vector $c$. The vector $c$ and can be varied to simulate different levels of enzyme expression, thus studying how the enzymatic activities affect the metabolic phenotypes. The model output has been classified according to the direction of the citrate flux. In particular, a flux toward the mitochondrion which provide substrate for mitochondrial bioenergetic represents the physiological condition of resting cell, while a flux toward the cytosol which provide substrate for cytosolic anabolism represents a condition of proliferating cells (e.g., cancer cells).

The constraints of the input factor space have been set one order of magnitude above and below the reference values (i.e., default values present in the model), and the sampling

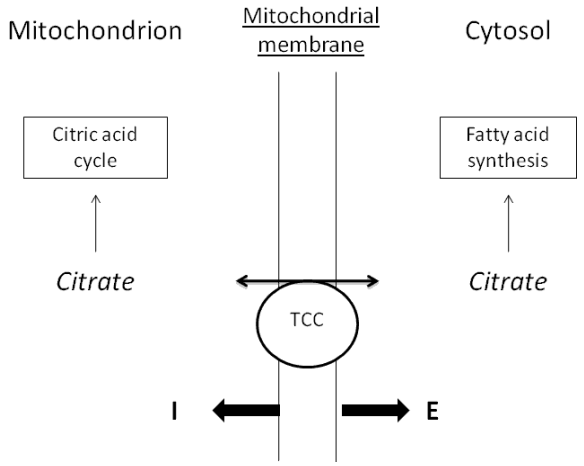


Figure 3.2: **Citrate flux**

The phenotype of citrate transport between mitochondrion and cytosol exerted by tricarboxylate carrier (TCC) has been characterized. The classification is based on the discrimination between the condition in which citric acid moves into mitochondrion (I), mainly to enter in the acid citric cycle, and the condition in which citric acid moves from mitochondrion to the cytosol (E) providing substrate for the fatty acid synthesis.

has been carried on by generating Sobol quasi-random sequence followed by a logarithmic mapping. As already seen, the use of Sobol quasi-random sequence is able to maximize the exploration of the space, while the choice of logarithmic mapping guarantees a uniform distribution in a space of more than one order of magnitude.

We used the two-sample Kolmogorov-Smirnov in order to compare the kinetic parameter distributions $c_i$ within the subset of model configurations which lead either, citrate flux toward cytosol or toward mitochondrion. The two-sample Kolmogorov-Smirnov test is a powerful non-parametric test able to quantify the distance between two empirical distribution functions to verify if they come from the same distribution [66]. More specifically, the $p$-values of Kolmogorov-Smirnov test have been evaluated, a small $p$-value implies that the distributions are highly dissimilar, that is the input factor is extremely relevant, on the contrary, a high $p$-values indicates that the same result may be obtained by a random sample, and thus it is associated with an unimportant factor.

The simulations of the model considering the simultaneous variation of the input factors makes the complexity of RSA super-linear, thus the computational burden can be a limiting factor when the number of parameters increases. However, there are heuristics proving that the number of parameters which strongly affects the behavior of a model is usually limited to a small subset of the total number of parameters [19]. In practice, through the evaluation of $p$-values trend as the number of model perturbations considered increases it is possible to understand when a sufficient number of model simulations to discriminate the parameters effects is reached.

In fact, a non variation in $p$-values has to be observed for an unimportant factors while a continuous decrease has to be observed for a relevant factor (since it becomes more

unlikely that the distributions difference occur by chance). Therefore, the presence of a number of approximatively constant trends as the number of considered model configurations increases suggests the relevant factors, if any, have been already detected. In our case study, for instance, it was observed that only three kinetic parameters, namely citrate dehydrogenase (IDH), adenilate nucleotide transferase (ANT), and pyruvate dehydrogenase (PDH) significantly affect the citrate flux (Figure 3.3).

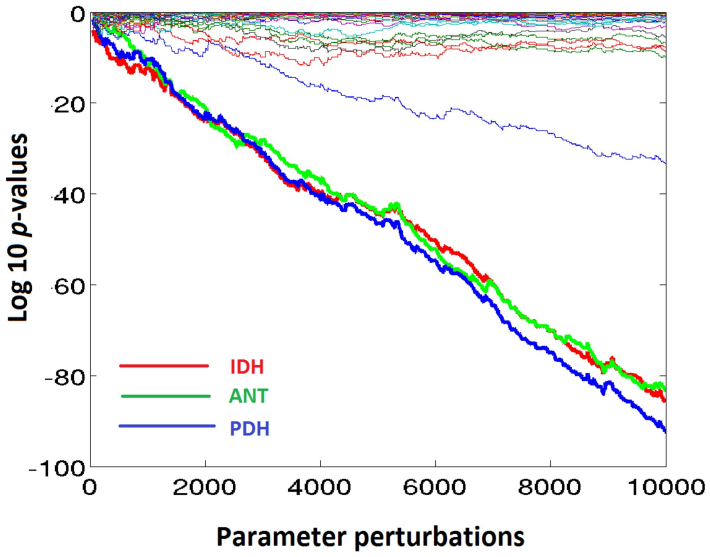


Figure 3.3: ***p* values Kolmogorov-Smirnov test**

A total of 10,000 unique parameter perturbations (i.e., model configurations) are sufficient to identify the most important P-values. Log10 of $p$-values obtained in relation to the number of parameter perturbations considering initial citric acid concentrations of 1e10-4 (similar results were obtained for the different initial conditions). The three parameters having the overall strongest impact on JCIT and JATP are reported in thick lines. Figure taken from [61].

Noteworthy, since the same comparison test has to be repeated several times, it is important to correct the computed $p$-values with statistical methods (e.g., the false discovery rate method) to decrease the occurrence of false positives. The computation of corrected sensitivity indexes also works as a confirmation of the role of the hypothetical influencing factors (Table 3.1)

Although RSA allows discriminating between significant and not significant reactions for a specific metabolic phenotype, it does not provide any information concerning the quantitative relation between reactions and phenotypes, and this is when RSM can play its role for extensive quantitative analysis.

In our specific case study, we applied RSM considering as input the three kinetic parameter (PHD, IDH, ANT) having the strongest influence and as output the citrate flux, as for RSA, and the ATP production flux which is an import trait of the cellular well-being. In fact, alongside with citrate flux we wanted to consider that ATP flux were in

| **Enzyme** | [citrate]=$1e^{-3}$ | [citrate]=$1e^{-4}$ | [citrate]=$1e^{-5}$ |
|---|---|---|---|
| IDH | $\mathbf{5.61e^{-4}}$ | $\mathbf{3.91e^{-4}}$ | $\mathbf{3.30e^{-2}}$ |
| ANT | $\mathbf{1.41e^{-2}}$ | $\mathbf{7.36e^{-4}}$ | $\mathbf{3.30e^{-6}}$ |
| PDH | $\mathbf{2.18e^{-2}}$ | $\mathbf{2.83e^{-4}}$ | $1.31e^{-1}$ |
| CS | $6.61e^{-2}$ | $\mathbf{2.71e^{-2}}$ | $3.68e^{-1}$ |
| CIII | 1 | $4.14e^{-1}$ | $\mathbf{1.68e^{-3}}$ |
| F1F0 | $3.12e^{-1}$ | $2.54e^{-1}$ | $\mathbf{4.13e^{-2}}$ |

Table 3.1: **False discovery rate**

False discovery rate calculated as part of the RSA; only the reactions associated with parameters emerged as significant in at least one analysis are shown: in bold parameters that are statistically significant ($< 0.05$).

valuable range, that is to a level allowing for a proper functioning of mitochondrion [54]. With the aim to better characterize the relation between PHD, IDH, ANT and citrate and ATP flux, all the combinations between two kinetic parameters and either the fluxes have been considered. Hence, the steady state values of citrate and ATP flux in relation to the perturbations of the three kinetic parameters have been collected. For each pair of kinetic parameters, a total of 2,500 combinations logarithmically distributed in a 2D grid were generated (over the same constraints of variations used with RSA, that are one order above and one order below the reference values). RSM has been repeated for the three different initial citrate concentrations: therefore, a total of 18 surfaces (3 pairs of reactions, 2 observed outputs, 3 values of [citrate]) were computed (Figure 3.4).

Through a detailed analysis of the quantitative relation between the kinetic parameter and the flux it is possible to identify the combinations of enzyme activity that lead to a specific phenotype. For instance, the highest efflux of citric acid for low values of IDH and high values of PDH, this condition, corresponding to biological inhibition of IDH and activation of PDH assures a negative flux of citric acid to mitochondrion (that is exit of citrate from mitochondrion to cytosol). RSM permits also to predict a possible range of kinetic parameters values to lead the system to the state E, which are approximately, IDH $< 1e^{-5}$ and $PDH > 1,000$.

The results have proven that the combined use of RSA and RSM with a proper integrated pipeline allows the identification of the pivotal factors affecting a property and for the subsequently characterization of their regulation over the target property.

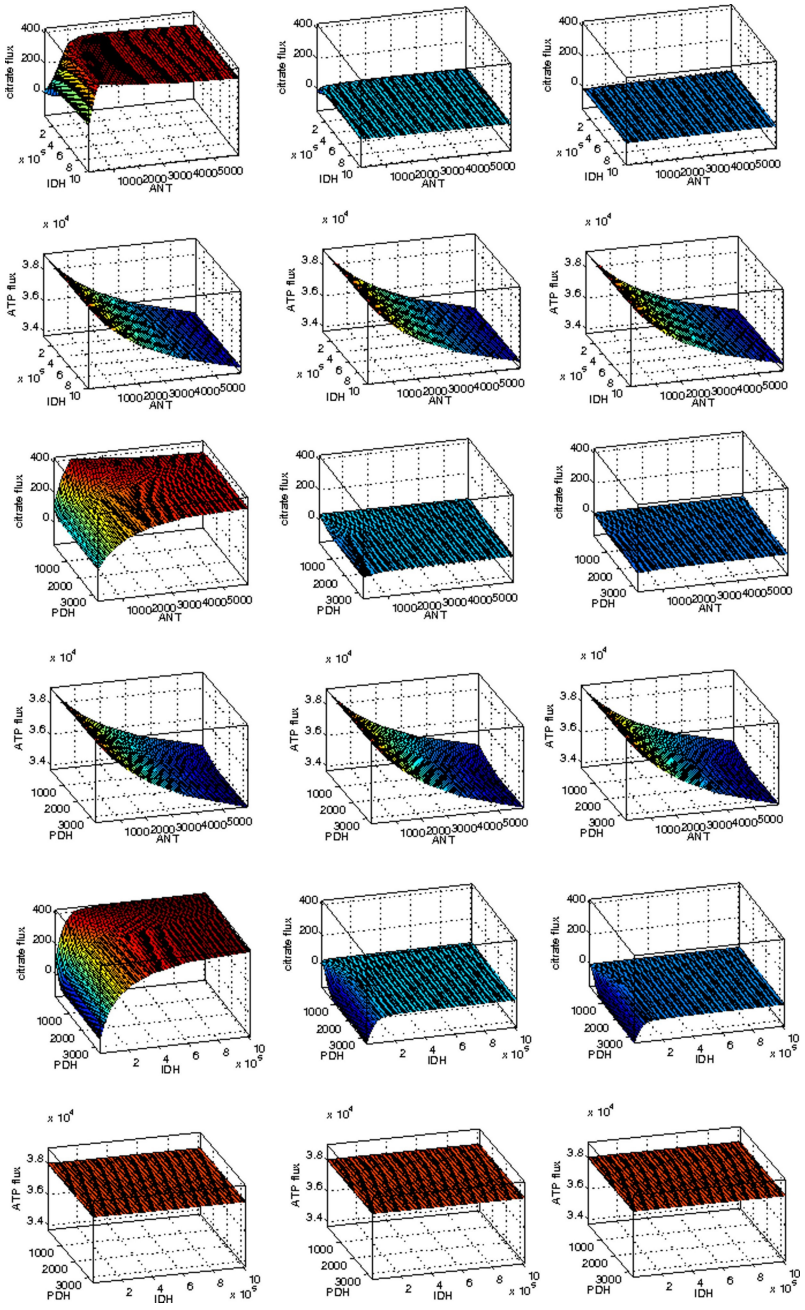


Figure 3.4: **Responses surfaces computed with the RSM**

Response surfaces obtained using the steady state values of JCIT and JATP (vertical axes, nmol/min/mg) in relation to the perturbation of IDH, ANT and PDH (horizontal axis) and using [citrate]=$5e^{-3}M$ (left), [CIT]=$5e^{-4}M$ (middle), [CIT]=$5e^{-5}M$ (right). Colors from blue to red are associated with, respectively, low and high flux values. Figure taken from [61].

## 3.2 Stochastic output analysis: the chemotaxis example

While a various number of SA techniques are defined in the analysis of deterministic models, a shortage of practical applications is present in the context of stochastic simulations. We aim at providing a strategy which takes the cue from the usual methodologies performed with deterministic models, in order to make them employable for the analysis stochastic models.

We tested SA methodologies on a stochastic model of bacterial chemotaxis [67]. Bacterial chemotaxis is a signaling process that leads a bacteria to move towards nutrients (chemoattractants) and away from harmful substances (chemorepellents) [68]. Noteworthy, biomolecules involved in signaling processes are usually present in relatively low amounts (i.e., order of tens of molecules). As such, the exploitation of stochastic modeling approaches can be suggested to properly describe such biochemical systems. In fact, while for models of metabolic pathways, in which the involved metabolites are characterized by high concentrations (i.e., expressed in molar concentrations), the leading simulation framework is represented by systems of ordinary differential equations, for the simulation of signaling pathways is instead quite common to find stochastic models [69].

Among others, the chemotaxis signaling in escherichia coli is one of the most well-understood of all sensory pathways, the structural and the biochemical details of every step of the pathway are well characterized. As a consequence of the high quality of the available information, chemotaxis has been chosen as a benchmark to develop computational models using a systems biology perspective [70]. The motor of chemotaxis is represented by flagella which are helical filaments whose movement provide bacterial motility [24]. The flagella movements depend on the chemosensory regulation of the rotatory activity; A flagella can spin clockwise or counterclockwise, the latter spin leads the bacterium to swim in a straight line, while the former spin leads to tumble in place, thus causing a change in the direction. Receptors near the surface of the cell sense molecules of interest (sugars, amino acids, dipeptides) and control the direction of flagellar rotation. Therefore, according to the frequency at which bacteria switches between clockwise and counterclockwise rotation of the flagella (i.e., chemotactic activity) , it results in moving in biased directions in response to concentration gradients of attractant or repellent ligands occurring in their surrounding environment.

The bacterial chemotaxis model that we considered describes the detailed protein-protein interactions involved in the signaling pathways, which sum up to a total of 59 reactions and 32 molecular species. The initial amounts (i.e., molecules number) have been set up according to the information retrieved in literature [71]. Essentially, the model considers all the main the main biochemical events involved in the regulation of flagella movements (Figure 3.2).

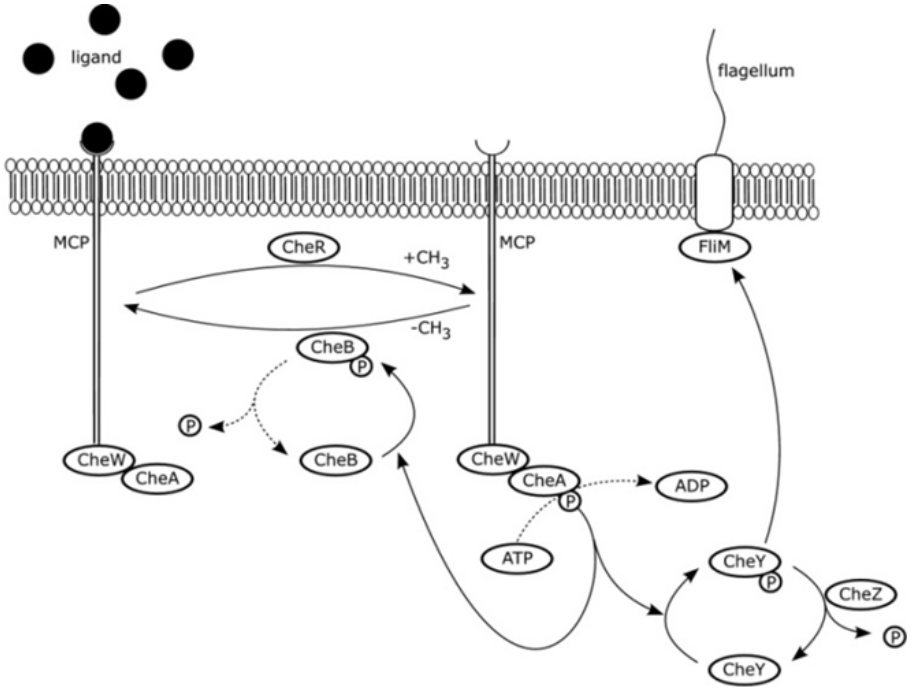


Figure 3.5: **Model of bacterial chemotaxis**

Signal transduction pathway in bacterial chemotaxis: solid arrows indicate enzyme-catalyzed reactions, dashed arrows indicate autocatalysis. The figure is taken from [72].

More specifically it takes into account both the signal sensor module and the regulator module. The sensor module is constituted by a receptor complex involving a methyl-accepting transmembrane protein ($MCP$), an adaptor protein ($CheW$) and a transmitter kinase protein ($CheA$). The regulator module is constituted by a methylesterase ($CheB$), a methyl transferase ($CheR$), the response regulator protein ($CheY$) and ($CheZ$). Each reaction in the model is given in the form "reagents $\rightarrow$ products", where the notation X + Y is used to represent a molecular interaction between species X and Y, while X::Y denotes that X and Y are chemically bound in the formation of a complex.

In order to use a $\tau$ leaping stochastic simulation method (which considers the probability of reactions according to molecular collisions), only monomolecular or bimolecular reactions are considered. The formation of complexes consisting of more than two species is performed stepwise. All the biochemical reactions (i.e., rules in the stochastic model formalization) are listed in Table 3.2.

The SA of bacterial chemotaxis model was performed in different steps. Initially we performed a screening SA, using elementary effects method, this was done considering as input all the 59 stochastic kinetic constants present in the mathematical model definition, while steady state concentration of $CheYp$ ($[CheYp]_s$) was the output. The choice of $CheYp$ is due to the fact that it is the final effector of bacterial chemotaxis signaling transduction since it causes the flagellum ripple that leads to the bacterium movements. Then, we performed a deeper analysis using a variance based method on the stochastic constants

| Index | Reactants | Products | Methylation state | Stochastic constant |
|---|---|---|---|---|
| 1 | $2MCPm + 2CheW$ | $2MCP^m :: 2CheW$ | $m = 0$ | $C_1 = 0.1$ |
| 2 | $2MCPm :: 2CheW$ | $2MCP^m + 2CheW$ | $m = 0$ | $C_2 = 0.01$ |
| 3 | $2MCPm :: 2CheW + 2CheA$ | $2MCP^m :: 2CheW :: 2CheA$ | $m = 0$ | $C_3 = 0.1$ |
| 4 | $2MCP^m :: 2CheW :: 2CheA$ | $2MCPm :: 2CheW + 2CheA$ | $m = 1$ | $C_4 = 0.02$ |
| 5 | $2MCP^m :: 2CheW :: 2CheA + CheR$ | $2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 0$ | $C_5 = 5.0e^{-7}$ |
| 6 | $2MCP^m :: 2CheW :: 2CheA + CheR$ | $2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 1$ | $C_6 = 5.0e^{-4}$ |
| 7 | $2MCP^m :: 2CheW :: 2CheA + CheR$ | $2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 2$ | $C_7 = 2.0e^{-7}$ |
| 8 | $2MCP^m :: 2CheW :: 2CheA + CheR$ | $2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 3$ | $C_8 = 0.0080$ |
| 9 | $2MCP^m :: 2CheW :: 2CheA + CheBp$ | $2MCPm - 1 :: 2CheW :: 2CheA + CheBp$ | $m = 1$ | $C_9 = 1$ |
| 10 | $2MCP^m :: 2CheW :: 2CheA + CheBp$ | $2MCPm - 1 :: 2CheW :: 2CheA + CheBp$ | $m = 2$ | $C_{10} = 0.6$ |
| 11 | $2MCP^m :: 2CheW :: 2CheA + CheBp$ | $2MCPm - 1 :: 2CheW :: 2CheA + CheBp$ | $m = 3$ | $C_{11} = 15$ |
| 12 | $2MCP^m :: 2CheW :: 2CheA + CheBp$ | $2MCPm - 1 :: 2CheW :: 2CheA + CheBp$ | $m = 4$ | $C_{12} = 0.35$ |
| 13 | $2MCP^m :: 2CheW :: 2CheA + ATP$ | $2MCP^m :: 2CheW :: 2CheAp$ | $m = 0$ | $C_{13} = 5.0e^{-7}$ |
| 14 | $2MCP^m :: 2CheW :: 2CheA + ATP$ | $2MCP^m :: 2CheW :: 2CheAp$ | $m = 1$ | $C_{14} = 5.0e^{-4}$ |
| 15 | $2MCP^m :: 2CheW :: 2CheA + ATP$ | $2MCP^m :: 2CheW :: 2CheAp$ | $m = 2$ | $C_{15} = 2.0e^{-4}$ |
| 16 | $2MCP^m :: 2CheW :: 2CheA + ATP$ | $2MCP^m :: 2CheW :: 2CheAp$ | $m = 3$ | $C_{16} = 6.0e^{-4}$ |
| 17 | $2MCP^m :: 2CheW :: 2CheA + ATP$ | $2MCP^m :: 2CheW :: 2CheAp$ | $m = 4$ | $C_{17} = 0.325$ |
| 18 | $2MCP^m :: 2CheW :: 2CheAp + CheY$ | $2MCP^m :: 2CheW :: 2CheA + CheYp$ | $m = 0$ | $C_{18} = 7.0e^{-6}$ |
| 19 | $2MCP^m :: 2CheW :: 2CheAp + CheY$ | $2MCP^m :: 2CheW :: 2CheA + CheYp$ | $m = 1$ | $C_5 = 0.0035$ |
| 20 | $2MCP^m :: 2CheW :: 2CheAp + CheY$ | $2MCP^m :: 2CheW :: 2CheA + CheYp$ | $m = 2$ | $C_{20} = 0.0014$ |
| 21 | $2MCP^m :: 2CheW :: 2CheAp + CheY$ | $2MCP^m :: 2CheW :: 2CheA + CheYp$ | $m = 3$ | $C_{21} = 0.0044$ |
| 22 | $2MCP^m :: 2CheW :: 2CheAp + CheY$ | $2MCP^m :: 2CheW :: 2CheA + CheYp$ | $m = 4$ | $C_{22} = 0.8$ |
| 23 | $2MCP^m :: 2CheW :: 2CheAp + CheB$ | $2MCP^m :: 2CheW :: 2CheA + CheBp$ | $m = 0$ | $C_{23} = 0.15$ |
| 24 | $2MCP^m :: 2CheW :: 2CheAp + CheB$ | $2MCP^m :: 2CheW :: 2CheA + CheBp$ | $m = 1$ | $C_{24} = 0.325$ |
| 25 | $2MCP^m :: 2CheW :: 2CheAp + CheB$ | $2MCP^m :: 2CheW :: 2CheA + CheBp$ | $m = 2$ | $C_{25} = 7.0e^{-6}$ |
| 26 | $2MCP^m :: 2CheW :: 2CheAp + CheB$ | $2MCP^m :: 2CheW :: 2CheA + CheBp$ | $m = 3$ | $C_{26} = 0.0035$ |
| 27 | $2MCP^m :: 2CheW :: 2CheAp + CheB$ | $2MCP^m :: 2CheW :: 2CheA + CheBp$ | $m = 4$ | $C_{27} = 0.0014$ |
| 28 | $lig + 2MCP^m :: 2CheW :: 2CheA$ | $lig :: 2MCP^m :: 2CheW :: 2CheA$ | $m = 0$ | $C_{28} = 0.0044$ |
| 29 | $lig + 2MCP^m :: 2CheW :: 2CheA$ | $lig :: 2MCP^m :: 2CheW :: 2CheA$ | $m = 1$ | $C_{29} = 0.29$ |
| 30 | $lig + 2MCP^m :: 2CheW :: 2CheA$ | $lig :: 2MCP^m :: 2CheW :: 2CheA$ | $m = 2$ | $C_{30} = 2.8e^{-5}$ |
| 31 | $lig + 2MCP^m :: 2CheW :: 2CheA$ | $lig :: 2MCP^m :: 2CheW :: 2CheA$ | $m = 3$ | $C_{31} = 0.0014$ |
| 32 | $lig + 2MCP^m :: 2CheW :: 2CheA$ | $lig :: 2MCP^m :: 2CheW :: 2CheA$ | $m = 4$ | $C_{32} = 0.0056$ |
| 33 | $lig :: 2MCPm :: 2CheW :: 2CheA$ | $lig + 2MCP^m :: 2CheW :: 2CheA$ | $m = 0$ | $C_{33} = 0.0175$ |
| 34 | $2MCP^m :: 2CheW :: 2CheA + CheR$ | $2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 1$ | $C_{34} = 1$ |

| Index | Reactants | Products | Methylation state | Stochastic constant |
|---|---|---|---|---|
| 35 | $2MCP^m :: 2CheW :: 2CheA + CheR$ | $2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 2$ | $C_{34} = 15$ |
| 36 | $2MCP^m :: 2CheW :: 2CheA + CheR$ | $2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 3$ | $C_{34} = 0.29$ |
| 37 | $2MCP^m :: 2CheW :: 2CheA + CheR$ | $2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 4$ | $C_{34} = 2.8e^{-5}$ |
| 38 | $lig :: 2MCP^m :: 2CheW :: 2CheA + CheR$ | $lig :: 2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 0$ | $C_{38} = 0.014$ |
| 39 | $lig :: 2MCPm :: 2CheW :: 2CheA + CheR$ | $lig :: 2MCPm + 1 :: 2CheW :: 2CheA + CheR$ | $m = 1$ | $C_{38} = 0.0056$ |
| 40 | $lig :: 2MCP^m :: 2CheW :: 2CheA + CheR$ | $lig :: 2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 2$ | $C_{38} = 0.01$ |
| 41 | $lig :: 2MCP^m :: 2CheW :: 2CheA + CheR$ | $lig :: 2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 3$ | $C_{38} = 0.0306$ |
| 42 | $lig :: 2MCP^m :: 2CheW :: 2CheA + CheBp$ | $lig :: 2MCPm - 1 :: 2CheW :: 2CheA + CheBp$ | $m = 1$ | $C_{42} = 5.0e^{-5}$ |
| 43 | $lig :: 2MCP^m :: 2CheW :: 2CheA + CheBp$ | $lig :: 2MCPm - 1 :: 2CheW :: 2CheA + CheBp$ | $m = 2$ | 0.025 |
| 44 | $lig :: 2MCP^m :: 2CheW :: 2CheA + CheBp$ | $lig :: 2MCPm - 1 :: 2CheW :: 2CheA + CheBp$ | $m = 3$ | 0.01 |
| 45 | $lig :: 2MCP^m :: 2CheW :: 2CheA + CheBp$ | $lig :: 2MCPm - 1 :: 2CheW :: 2CheA + CheBp$ | $m = 4$ | 0.0306 |
| 46 | $lig :: 2MCPm :: 2CheW :: 2CheA + ATP$ | $lig :: 2MCPm :: 2CheW :: 2CheAp$ | $m = 1$ | $C_{46} = 1.2$ |
| 47 | $2MCP^m :: 2CheW :: 2CheA + CheR$ | $2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 2$ | $C_{47} = 15$ |
| 48 | $2MCP^m :: 2CheW :: 2CheA + CheR$ | $2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 3$ | $C_{48} = 0.0165$ |
| 49 | $2MCP^m :: 2CheW :: 2CheA + CheR$ | $2MCP^{m+1} :: 2CheW :: 2CheA + CheR$ | $m = 4$ | $C_{49} = 5.0e^{-5}$ |
| 50 | $lig :: 2MCPm :: 2CheW :: 2CheAp + CheY$ | $lig :: 2MCPm :: 2CheW :: 2CheA + CheYp$ | $m = 0$ | $C_{50} = 0.03$ |
| 51 | $lig :: 2MCPm :: 2CheW :: 2CheAp + CheY$ | $lig :: 2MCPm :: 2CheW :: 2CheA + CheYp$ | $m = 1$ | $C_{51} = 0.0112$ |
| 52 | $lig :: 2MCPm :: 2CheW :: 2CheAp + CheY$ | $lig :: 2MCPm :: 2CheW :: 2CheA + CheYp$ | $m = 2$ | $C_{52} = 0.0343$ |
| 53 | $lig :: 2MCPm :: 2CheW :: 2CheAp + CheY$ | $lig :: 2MCPm :: 2CheW :: 2CheA + CheYp$ | $m = 3$ | $C_{53} = 0.05$ |
| 54 | $lig :: 2MCPm :: 2CheW :: 2CheAp + CheB$ | $lig :: 2MCPm :: 2CheW :: 2CheA + CheBp$ | $m = 1$ | $C_{54} = 6.8e^{-8}$ |
| 55 | $lig :: 2MCPm :: 2CheW :: 2CheAp + CheB$ | $lig :: 2MCPm :: 2CheW :: 2CheA + CheBp$ | $m = 2$ | $C_{55} = 0.0336$ |
| 56 | $lig :: 2MCPm :: 2CheW :: 2CheAp + CheB$ | $lig :: 2MCPm :: 2CheW :: 2CheA + CheBp$ | $m = 3$ | $C_{56} = 0.0135$ |
| 57 | $lig :: 2MCPm :: 2CheW :: 2CheAp + CheB$ | $lig :: 2MCPm :: 2CheW :: 2CheA + CheBp$ | $m = 4$ | $C_{57} = 0.035$ |
| 58 | $CheYp + CheZ$ | $CheY + CheZ$ | | $C_{58} = 1.4$ |
| 59 | $CheBp$ | $CheB$ | | $C_{59} = 15$ |

Table 3.2: **Stochastic model of bacterial chemotaxis**
List of rules defining the stochastic model together with the corresponding stochastic constants.

having the strongest influence on $[CheYp]_s$ according to the results of the screening analysis. Finally, we studied, using regression methodologies, the specific correlations, within the input space, between the pivotal stochastic kinetic constants and $[CheYp]_s$.

Since the simulation have been performed using the stochastic simulator algorithm $\tau$-DPP [73], the system can evolves following qualitatively similar dynamics that can be quantitatively different. Therefore, for each model configuration we obtain a distributions of values corresponding to the number of model executions (that we set to 100). The simulation time was set to 0.5 time units, a time proven to be enough to assure the attainment of steady state in the different conditions (Figure 3.6).

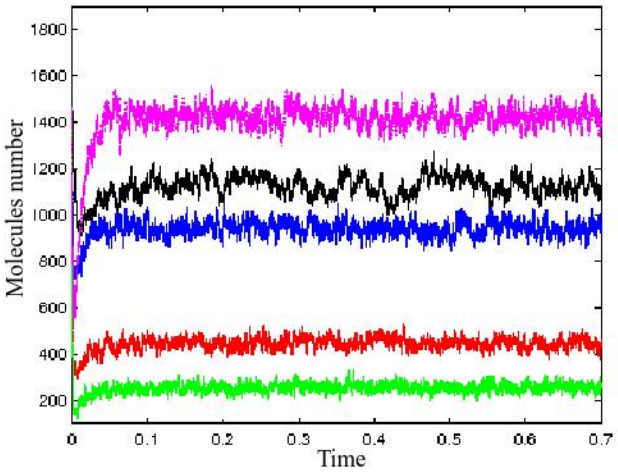


Figure 3.6: **Chemotaxis model simulations**
Dynamics of *CheYp* molecules number obtained starting from five different model configurations.

The problem in the application of elementary effects method is that it has been thought for deterministic models, thus having a specific output value per model configuration. Instead, when we deal with stochastic models, we have distributions of values per each model configuration. A possible approach to compare output distributions, as suggest by Degasperi and Gilmore [74], is the use of histogram distance.

Using histogram distance, the quantity $D = Y(X+\Delta x_i) - Y(X) = Y_1 - Y_0$, to calculate a generic elementary effect, is computed as follow:

$$\sum_{i=1}^{h} \left| \frac{\sum_{j=1}^{n(Y_1)|} \chi(Y_{1_j}, I_i)}{n(Y_1)} - \frac{\sum_{j=1}^{n(Y_0)} \chi(Y_{0_j}, I_i)}{n(Y_0)} \right| \tag{3.2}$$

where $h$ is the number of histogram bins, $n(Y_1)$ and $n(Y_0)$ are the cardinalities of the multisets $Y_1$ and $Y_0$ and correspond to the number of simulations executed, the function $\chi$ returns 1 if the element $Y_{1/0j}$ belongs to the interval $I_i$, 0 otherwise. $I_i$ is the $i$th interval in the range, which runs from $m_{min} + \frac{(i-1)L}{h}$ to $m_{min} + \frac{iL}{h}$, where $m_{min} = min\ \{Y_1 \cup Y_0\}$, $m_{max} = max\ \{Y_1 \cup Y_0\}$ and $L = m_{max} - m_{min}$.

| **Stochastic constant** | $\mu^*$ | $\sigma$ |
|---|---|---|
| $c_{13}$ | 37204.7 | 24776.7 |
| $c_5$ | 36061.1 | 27723.5 |
| $c_{18}$ | 6231.71 | 7407.27 |
| $c_{25}$ | 5758.29 | 7288.1 |
| $c_{37}$ | 3637.65 | 2897.85 |
| $c_{30}$ | 2649.92 | 2857.51 |
| $c_{42}$ | 1803.36 | 1461.62 |
| $c_{49}$ | 1175.78 | 1277.24 |
| $c_{54}$ | 792.272 | 840.66 |
| $c_{15}$ | 124.342 | 102.439 |

Table 3.3: **Elementary effects ranking**

List of the ten most influential input factors according to the average of absolute elementary effect values $\mu^*$ and the corresponding standard deviations $\sigma$.

With sufficient simulations runs, the use of the histogram distance is able to precisely describe the difference between output distributions obtained changing model configuration (Figure 3.7).

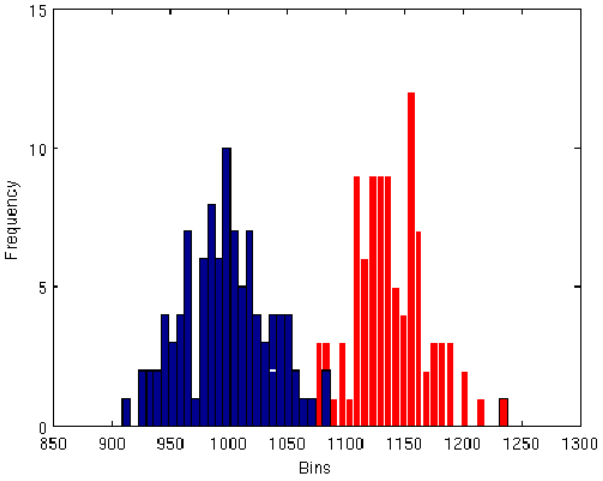


Figure 3.7: **Histogram distance:**

Histogram distributions of two outputs sets (respectively blue and red bars) obtained simulating the model in two different model configurations.

The computation of elementary effects using trajectories sampling (selecting the 10‘ trajectories with the maximum spread among 40, as described in 2.3.1) and histogram distance to compare output distributions allowed us to detect the most influencing stochastic constants over $[CheYp]_s$ (Table 3.3).

However, we verified that the histogram distance has the drawback not to be capable of distinguish between different high distributions differences. In fact, while histogram dis-

tance is a profitable approach to compare similar distributions (i.e., for non-disjoint sets of values) when the set are completely disjoint histogram distance reaches the maximum theoretical value regardless the overall distant (i.e., it is equal to 2).
The problem is that for global SA, where the exploration of input factor space is usually extensive (i.e., large constraints of variations allowed to the input values), the chance to have very dissimilar output values is a likely event. For this reason, we tested other comparison measurements, in particular we verified that the use of average linkage between couple of output distributions is a good compromise between the capability to detect small differences while gain also the possibility to compare significant dissimilar distributions. The average linkage is based on the computation of the sum of the differences between all the possible couples of output values (Figure 3.8) over the number of comparisons. This approach is basically the same exploited to compute the average linkage metric in cluster analysis [75].

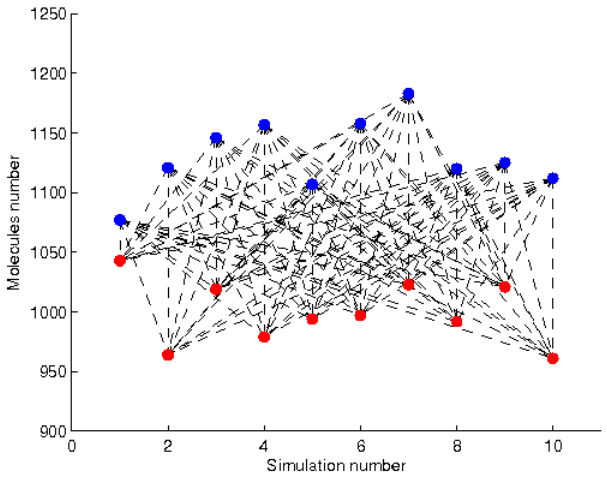


Figure 3.8: **Average linkage** Couples of output value considered for average linkage computation (example with 10 model executions for each of the two model configurations).

Using average linkage metric, the quantity $D = Y(X+\Delta x_i) - Y(X) = Y_1 - Y_0$ is computed as follow:

$$\frac{\sum_{i=1}^{n(Y_1)} \sum_{j=1}^{n(Y_0)} Y_1(i) - Y_0(j)}{n(Y_0)n(Y_1)} \tag{3.3}$$

Moreover, we studied how the exploited sampling procedure (i.e., trajectory vs. radial sampling, see 2.3.1 for details) affects the computation of sensitivity indexes. To this purpose, we derived a number of base points (i.e., model configurations) equal to the number of trajectories, and performed a radial sampling considering same $\Delta$ of trajectory sampling. The choice of the same $\Delta$ alongside to the execution of the same number of model simulations per each model configuration (i.e., 100) was to achieve comparable data

| **Base point** | $\sigma([CheY]s)$ | $\mu(EE_i)$ |
|---|---|---|
| 1 | 180.32 | 153.60 |
| 2 | 108.42 | 102.23 |
| 3 | 119.98 | 108.72 |
| 4 | 279.15 | 229.85 |
| 5 | 770.98 | 423.03 |
| 6 | 153.73 | 138.15 |
| 7 | 66.55 | 72.13 |
| 8 | 130.62 | 106.74 |
| 9 | 135.6 | 78.31 |
| 10 | 266.66 | 181.12 |

Table 3.4: **Sampling and model variability**
A strong correlation is present between intrinsic model variability expressed in term of standard deviation of model output and mean of elementary effects. The correlation coefficient between the two indexes is 0.98.

(i.e., whose difference depends only on the sampling).
The analysis of the elementary effects computed in the different base points reveals a connection between the local variabilities of the model (i.e., output standard deviation) and the values of elementary effects (Table 3.4). The analysis suggests that the use of trajectory based sampling should be preferred when the aim is to generally explore the model behavior (i.e., verify if there exists a condition in which a target variable overcome a given threshold), while the use radial approach should be preferred for ranking input factors influences. In fact, since different input space regions (i.e., base points), may be associated with different model variabilities, starting from the same reference configurations (i.e., radial sampling), rather than from variable configurations (i.e., trajectory sampling) assures to have the same bias in the computation of elementary effects. Therefore, for the classic SA studies aimed at the comparison between input factor effects, the proper choice for the evaluation of elementary effects can be radial sampling since it assures the computation of comparable indexes, which avoids random effects deriving from the fact that input influences are evaluated in regions subject to different sensitivities at perturbations.

In effect, the computation of elementary effects using radial sampling and average linkage allowed us to disclose important relations that were not observed with trajectories and histogram distance. In particular, the stochastic constants having the higher $\mu^*$ are $c_8$, $c_{37}$ , $c_{40}$,$c_{48}$, and $c_{49}$. However, the majority of the constants having negligible effect are the same (i.e., both methods provide consistent qualitative classification).
Then, we focused on the analysis of the five most important stochastic constants according to elementary effects computed exploiting radial sampling and average linkage distances.

To this end, for the computation of the first and total order variance based sensitivity index, we applied the methods proposed by Saltelli et. al in [19] (see 2.3.2 for details).

In order to take into account model stochasticity, we repeated a number $s$ of simulations for each model configuration (we set $s = 100$, as for the computation of elementary effects). As a consequence, we generated matrices of size ($s * N$, where $N$ is the sampling size), therefore, we needed to adapt the original formulas for the computation of variance based sensitivity indexes (which were defined for deterministic output), so to consider the $s$ model realizations obtained in each model configuration.
The estimation of the first order sensitivity index of the parameter $i$th is computed as follow:

$$S_i = \frac{V[E(Y|x_i)]}{V(Y)} = \frac{Y_A \cdot Y_{A_B^i}^T - f_0^2}{Y_A \cdot Y_B^T - f_0^2} = \frac{\frac{1}{S*N}\sum_{z=1}^{S}\sum_{j=1}^{N} Y_A^{zj} Y_{A_i}^{zj} - f_0^2}{\frac{1}{S*N}\sum_{z=1}^{S}\sum_{j=1}^{N} Y_A^{zj} Y_B^{zj} - f_0^2} \tag{3.4}$$

where the symbol $\cdot$ denote the scalar product of the two matrices, the quantity $f_0^2$ represents the mean of $Y$ in all the model configurations and it is calculated as follow:

$$f_0^2 = \Big(\frac{1}{S*N}\sum_{z=1}^{S}\sum_{j=1}^{N} Y_A^{zj} Y_B^{zj}\Big)^2 \tag{3.5}$$

Similarly the estimation of the total order sensitivity index of the $i$th parameter is computed as follow:

$$S_{T_i} = 1 - \frac{V[E(Y|x_{\sim i})]}{V(Y)} = \frac{Y_B \cdot Y_{A_B^i}^T - f_0^2}{Y_A \cdot Y_B^T - f_0^2} = \frac{\frac{1}{S*N}\sum_{z=1}^{S}\sum_{j=1}^{N} Y_B^{zj} Y_{A_i}^{zj} - f_0^2}{\frac{1}{S*N}\sum_{z=1}^{S}\sum_{j=1}^{N} Y_A^{zj} Y_B^{zj} - f_0^2} \tag{3.6}$$

Following this procedure we are able to compute the set of both first and total sensitivity indexes of set of the five stochastic constants $k$ with the total cost of $(N * s(k+2))$ model simulations. In our example $N$ was set to 1000, thus a total of $5 \times 10^5$ model simulations have been executed. The computed indexes are consistent with $\mu^*$, in particular the main influence of the stochastic kinetic constant $c_8$ has been confirmed, while the other four kinetic constants considered have a lower, but significant, similar quantitative influence of $[CheYp]_s$ variance (Table 3.5).

We studied how the size of the sampling matrix $N$ and the number of model repetition $s$, for each model configuration, affect the computation of variance based sensitivity indexes. In particular, it has been observed that the parameter $s$ has a negligible effect on the computation of the sensitivity indexes, probably this is due to the average effect in the relative variation of $[CheYp]_s$ values occurring when several model configurations are

| **Index** | $C_8$ | $C_{37}$ | $C_{40}$ | $C_{48}$ | $C_{49}$ |
|---|---|---|---|---|---|
| $S_i$ | 0.54 | 0.06 | 0.09 | 0.08 | 0.07 |
| $S_{T_i}$ | 0.74 | 0.11 | 0.13 | 0.10 | 0.12 |

Table 3.5: **Variance based sensitivity indexes:**
First and total variance based sensitivity index for the five stochastic constants having the strongest effects according to screening SA.

considered.
Instead, the sampling matrix $N$ significantly affects the sensitivity indexes values, and a few hundreds of model configurations are required in order to assure the convergence of the sensitivity indexes (Figure 3.9).

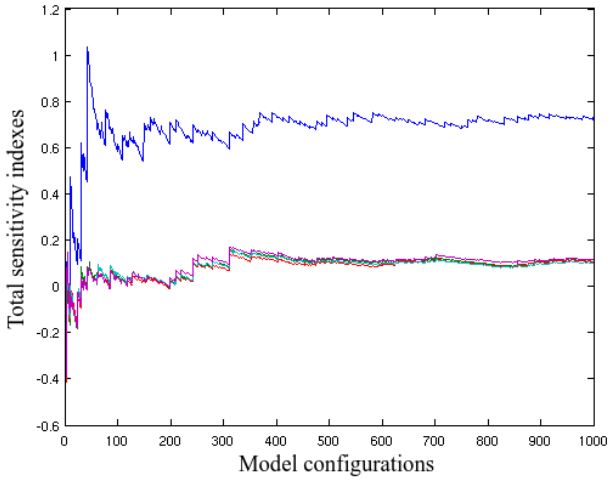


Figure 3.9: **Total sensitivity index trends**
Evolution of $S_{T_i}$ indexes as the number of model configurations considered increases.

Despite screening SA methods and variance based approaches are very powerful tools to respectively rank input effects and to deeply analyze to which extent they affect output variance, as we already seen, they do not provide information on the qualitative/quantitative input-output relation. This kind of information can be retrieved exploiting a regression based sensitivity analysis aimed at the reconstruction of the relation between input parameter and output.
We performed a regression based SA of $[CheYp]s$ considering as inputs the three stochastic constants having the highest $S_{t_i}$, which are respectively associated with, the methylation of methyl-accepting chemotaxis protein MCP ($c_8$), and with the phosphotransfer of CheYp at different levels of methylation in presence of ligand ($c_{48}$ and $c_{49}$).
To study the effects using a global SA perspective, the regression analysis has been performed starting from different model configurations and precisely we considered the ones

used for radial sampling in elementary effects analysis. Starting from each reference model configuration (i.e., base point), each of the three input has been changed within its defined constraints of variation, moving from the lowest value to the highest values in 100 uniform distributed steps. The regression curves have been obtained using polynomial interpolation in which the degree of the polynomial is increased until the regression curve has an $R^2$ value is below a given threshold $\epsilon$ (set to 0.95). Noteworthy, to build the regression curve, an accurate estimates of $[CheYp]_s$ value associated with each specific model configurations is required, thus differently by variance based analysis, the number of simulations $s$ is extremely relevant to build a correct regression curve. In fact, if only few model outcomes of stochastic simulations are considered, frequent changes of the regression curves may occur due to the presence of noise rather than from the actual output variation. It is also important to consider that a regression analysis makes sense only in presence of quantitative variations of the model output (i.e., the dynamics are similar), instead, in case a model exhibits change in the output mode, a qualitative SA analysis based on the evaluation of the different dynamic trends should be preferred (as we will see in chapter 4).

In our case study, we identified a not linear (i.e., the input significantly control the output value) and monotonic relation between $c_8$ and $[CheYp]s$. More specifically, $c_8$ reduction leads to increase of $[CheYp]s$ and the effect is more intense near the lowest constraint values. This kind of relationship has been found similar in all the considered model configurations.

The relationship between $c_{49}$ and $[CheYp]s$ was found to be strongly linear (directly proportional), and also in this case the same relation was found in all the different considered model configurations.

Instead, the analysis of the relationship between $c_{48}$ and $[CheYp]s$ revealed unexpected results, in fact the relationship is not linear and not constant, that is according the reference model configuration the same variation of $c_{48}$ leads to different qualitative variations of $[CheYp]s$ (i.e., it can either increase or decrease).

As a consequence, predict the effect of the modifications of this input is a tough task since it depends also by the values assumed by other inputs. This behavior is probably due to the presence of feedbacks in the model structure (Figure 3.2) which lead the model to reach unforeseeable dynamics depending on the combinations of different input values, which in turn should mimic biochemical process interactions.

To summarize the analysis of stochastic outputs, we described how it is possible to apply the common SA techniques in the study of stochastic models, adapting the deterministic formulations to cope with the noisy outputs. Specifically, by using as a case study a stochastic model of bacterial chemotaxis, we showed how the application of the adapted SA methods in a step based process, allowed us to get more insights about the regulation of the model behavior.

## 3.3 Description of condition-specific responses

A critical issue in a various number of fields is the application of actions on a target system so that its behavior is affected only in some specific conditions (i.e., state of the system) and in a specific way as well (i.e., acting on a given property level rather than perturbing the overall system)[76]. In fact, the aim is often to alter the system behavior only when it is strictly necessary while trying to avoid any influence when it is not required. This is particular true for systems whose behaviors is highly regulated and in which even small variations can lead to deleterious effects (e.g., biological system). In this context, the exploitation of global SA to study the system response considering both different system properties and different system states, can provide meaningful information on the overall system behavior, thus allowing to study the specificity of different system perturbations in different conditions.

We tackled the specificity response problem by analyzing a computational model on a metabolic process, developed by us, described in Mosca et al. [72]. The model, based on a system of ordinary differential equations describes the metabolic states regulated by akt kinase, a master regulator of metabolism related to cell proliferation [77]. The model provides the link between different metabolic pathways, namely, PI3K/Akt/mTOR pathway and nucleotide biosynthesis which are related to the proliferating activity, and glycolysis and lactic acid production which are related to the bio-energetic state (Figure 3.10). A typical characteristic of cellular metabolism is the relation with the cellular growth, as a matter of fact, the onset of proliferation induces important changes in cellular metabolism in both normal and cancer cells [54]. Therefore metabolic activities in proliferating cells are fundamentally different from those in non-proliferating cells, and a typical trait differentiating the two conditions is provided by Akt level which is responsible for the modification of some pivotal reaction rates [78]. In our study we wanted to analyze how the system responds to the same reactions rates variations when it is characterized by different concentration of Akt ([Akt]), representing respectively, a proliferating cell state (characterized by high [Akt], named H), and resting cell state (characterized by low [Akt], named L).

To this end, we computed derivatives based global sensitivity measures (DGSMs) considering all the reaction rates (as inputs), and fluxes (as outputs), in both L and H state. The methods is based on the evaluation of local derivatives at randomly or quasi-randomly selected points in the whole range of uncertainty [27]. More specifically, we derived two sets of model configurations, one for H state and one for L state, by mapping Sobol quasi random sequences within two input spaces representing respectively, the space around H and L state, thus allowing to study the sensitivities in the two conditions. DGSMs has proven to retain accuracy of derivatives computation while guaranteeing an intensive exploration of the input space, thus including the ability to consider interaction effects, as

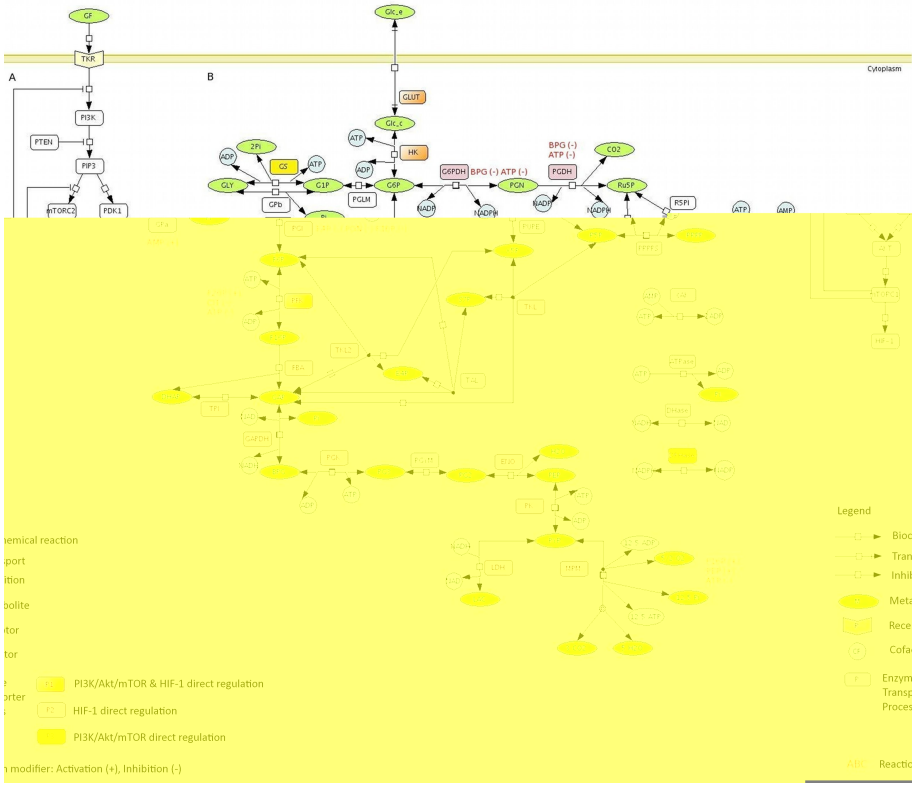


Figure 3.10: **Model of mitochondrial metabolism**

Integrated model on the regulation exerted by Akt. The figure is taken from [72].

Sobol indexes [79].

In detail, to compute DGSMs for both L and H states, we generated a set of model configurations $p_k$ in the form:

$$p_k = (p_{k1}, \ldots, p_{kn}) \tag{3.7}$$

in which $n$, representing the sampling size, was set to 500, while $k = 29$ is given by the number of reaction rates. Each $p_k = [p - \epsilon p, p + \epsilon p]$, where $\epsilon = 5 * 10^{-6}$, represents a possible perturbation of the default values representing the reference conditions (Table 3.6). Beside the computation of DGSMs, we also computed local sensitivity measures by mean of the calculation of partial derivatives at both L and H state, verifying the stability (i.e., robustness) of the model behavior (Table 3.6). The analysis of the stability is a way to measure the reliability of the model, in fact, biological system are characterized by high robustness, at least in the physiological state, and a proper modeling should reflect thus attitude (on the other hand, cancer cells are often more metabolic unstable). The model configurations $p_k$ have been used to numerically compute normalized derivatives, that is,

| **Reaction rate** | H state | | L state | |
|---|---|---|---|---|
| | $p_i$ | $s$ | $p_i$ | $s$ |
| AK | $1.412e^{2}$ | $3.449e^{-2}$ | $1.412e^{2}$ | $3.449e^{-2}$ |
| ATPase | $6.210e^{3}$ | $2.292e^{-1}$ | $6.210e^{3}$ | $2.373e^{-1}$ |
| DHase | $4.982e^{6}$ | $1.171e^{-3}$ | $4.982e{+}^{6}$ | $2.278e^{-3}$ |
| DPHase | $1.278e^{5}$ | $6.963e^{-2}$ | $7.413e^{4}$ | $3.430e^{-2}$ |
| ENO | $1.609e^{2}$ | $3.627e^{-4}$ | $9.330e^{1}$ | $4.074e^{-4}$ |
| FBA | $1.463e^{1}$ | $6.639e^{-4}$ | $8.484e^{0}$ | $1.491e^{-3}$ |
| G6PDH | $1.008e^{0}$ | $1.181e^{0}$ | $5.846e^{-1}$ | $7.600e^{-1}$ |
| GAPDH | $1.091e^{2}$ | $2.345e^{-2}$ | $6.329e^{1}$ | $2.696e^{-2}$ |
| GLUT | $2.303e^{1}$ | $4.271e^{-1}$ | $1.336e^{1}$ | $5.256e^{-1}$ |
| GPa | $3.347e^{-2}$ | $3.602e^{-2}$ | $3.347e^{-2}$ | $3.790e^{-2}$ |
| GPb | $1.049e^{-2}$ | $3.679e^{-2}$ | $1.049e^{-2}$ | $3.962e^{-2}$ |
| GS | $3.204e^{4}$ | $3.934e^{-2}$ | $1.858e^{4}$ | $6.016e^{-2}$ |
| HK | $8.685e^{1}$ | $7.173e^{-2}$ | $5.037e^{1}$ | $5.998e^{-2}$ |
| LDH | $3.403e^{2}$ | $3.331e^{-3}$ | $1.974e{+}^{2}$ | $3.647e^{-3}$ |
| MPM | $9.801e^{6}$ | $1.142e^{-1}$ | $1.137e^{7}$ | $1.423e^{-1}$ |
| PFK | $1.076e^{2}$ | $5.706e^{-2}$ | $6.243e^{1}$ | $1.138e^{-1}$ |
| PGDH | $3.102e^{1}$ | $5.136e^{-3}$ | $1.799e^{1}$ | $6.160e^{-4}$ |
| PGI | $7.778e^{3}$ | $3.709e^{-2}$ | $4.511e^{3}$ | $3.216e^{-3}$ |
| PGK | $7.341e^{1}$ | $8.065e^{-3}$ | $4.258e^{1}$ | $9.261e^{-3}$ |
| PGLM | $7.364e^{0}$ | $3.351e^{-2}$ | $7.364e^{0}$ | $1.651e^{-2}$ |
| PGYM | $1.540e^{2}$ | $2.232e^{-3}$ | $1.540e^{2}$ | $2.489e^{-3}$ |
| PK | $2.781e^{1}$ | $1.070e^{-7}$ | $1.613e^{1}$ | $4.182e^{-7}$ |
| PRPPS | $5.104e^{1}$ | $5.898e^{-1}$ | $5.104e^{1}$ | $5.439e^{-1}$ |
| R5PI | $7.646e^{1}$ | $2.643e^{-2}$ | $7.646e^{1}$ | $6.134e^{-2}$ |
| RUPE | $1.471e^{0}$ | $1.156e^{-3}$ | $1.471e^{1}$ | $2.467e^{-3}$ |
| TAL | $5.827e^{1}$ | $7.061e^{-4}$ | $5.827e^{1}$ | $1.040e^{-4}$ |
| TKL | $1.056e^{3}$ | $5.345e^{-4}$ | $6.124e^{2}$ | $1.180e^{-3}$ |
| TKL2 | $1.761e^{1}$ | $4.674e^{-1}$ | $1.021e^{1}$ | $2.420e^{-2}$ |
| TPI | $5.976e^{0}$ | $2.779e^{-4}$ | $3.466e^{0}$ | $6.055e^{-4}$ |

Table 3.6: **Reaction rates Akt model**

Reaction rates values expressed in nmol/min/mg with the corresponding local sensitivities $s$ computed as partial derivatives.

a generic derivative $k$ associated with a given reaction rate $i$ over a flux $j$ is in the form:

$$d^{ijk} = \frac{|[J_i(p_1, \ldots, p_{kj} + \delta, \ldots, p_{kn}) - J_i(p_{k1}, \ldots, p_{kj}, \ldots, p_{kn})|}{\delta} \times \frac{|J_i(p_{k1}, \ldots, p_{kj}, \ldots, p_{kn})|}{p_{kj}} \tag{3.8}$$

which measures the influences of the perturbations $\delta$ (set as $10^{-6}$) in the $j$-th element of $p_k$ over the steady state flux $J_i$. Scaling with respect to the actual flux avoids differences due to the different absolute values of the fluxes rather than flux variations (otherwise the same variation in a lower flux would be associated with a higher derivative, since the denominator is lower).

Hence, two matrices of sensitivities between reaction rates and fluxes (thus of 29X29 size) were computed, one representing sensitivities at low Akt condition (obtained around L state), and one representing sensitivities at high Akt condition (obtained around H state). Subsequently, two corresponding scaled matrices were derived as:

$$g^{ij} = \frac{m_{ij}^2 + s_{ij}^2}{\sum_{j=1}^{29} m_{ij}^2 + s_{ij}^2} \tag{3.9}$$

where, $m_{ij}$ is mean and $s_{ij}$ is the sample standard deviation of derivatives $d_{ijk}$.
Finally, we computed the normalized differential ranking of the reaction as:

$$R^{ij} = \frac{r_{ij}^L - r_{ij}^H}{r_{ij}^L + r_{ij}^H} \tag{3.10}$$

where, $r_{ij}$ is the rank of the scaled sensitivity index $g_{ij}$, when ordering the elements $[g_{i1}, \ldots, g_{i29}]$ from the highest (most influent) parameter to the lowest (superscripts H and L indicate the reference state).

The coefficient $R_{ij}$ is the ratio of the influence between the state $L$ and the state $H$ of the parameter $j$ over the flux $i$. The index $R_{ij}$ is negative if the effects is greater on $H$ state, on the contrary it is positive if the effect is greater on $L$ state. Therefore, the analysis of the ratio effects provides a general map of the different sensitivities (Figure 3.11). The results show that some steps of the metabolic network have significant sensitivities ratios in the two metabolic states. For instance G6PDH and TKL affect the proliferation pathways in the model at higher level in H condition with respect to L condition.

Therefore, the use of global SA can be used to analyze the specificity to system perturbations, providing information that can be used to assist in the selection of drug targets in anticancer treatments balancing the desired effects (i.e., modify particular fluxes in a specific condition) and the undesired effects (i.e., minimizing non specific, possibly deleterious, alterations).

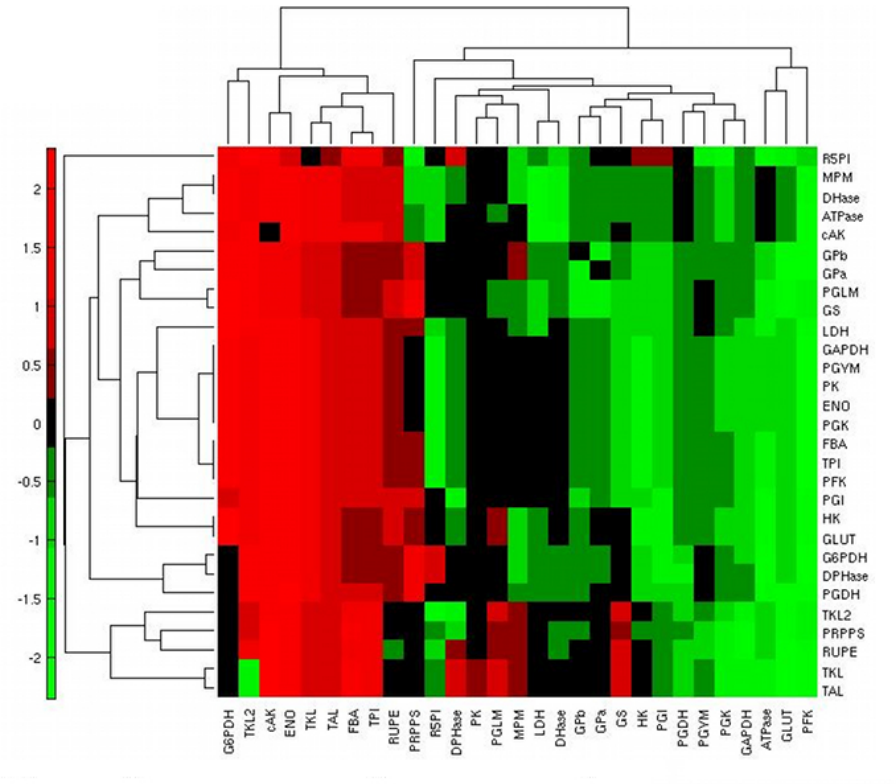


Figure 3.11: **Differential sensitivity ranking**

Differential ranking of steady state flux sensitivities (rows) to reactions (columns) when comparing conditions H and L; red: high sensitivity in condition H; green: high sensitivity in condition L.

## 3.4 Exploratory analysis of spatial effects

In this section, we consider the effect of spatiality in biochemical systems, an aspect that is commonly disregarded in the analysis of the dynamics of model simulations. In fact, include spatial effects in simulation algorithm is a complex task, thus the space is usually considered as an homogeneous system (i.e., instead of having local concentrations of elements the system is considered well stirred). Moreover, when in a biological process the time scale of biochemical transformations is higher than the one of molecular diffusion (which is true in various conditions), the well-stirred approximation if justified [80]. However, since in reality living cells are very far from the homogeneous and diluted compartment used for their modeling, there can be conditions in which the explicit considerations of diffusion processes becomes essential. This is particularly true for crowded environments, like intracellular systems, where also spatial segregation of biochemical entities slows diffusion processes [81].

To this purpose, we developed a simple model of gene expression regulation and exploited a stochastic algorithm for reaction diffusion processes. The algorithm is based on the evaluation of the chemical master equation (see 1.2), and it keeps track of the amount of substances in different locations of a space domain $\theta$, derived from the space partitioning into smaller sub-compartments $i$. Each compartment has a characteristic volume $V_i = l^d$, being $l$ an adjustable parameter and $d \in 1, 2, 3$ representing the three spatial dimensions. The choice of the granularity in which partition the space should be performed to assure the definition of sub-compartments in which the homogeneous conditions can be acceptable. The algorithm is also able to takes into account the crowded effect, by evaluating the free volume available in each compartment, this can be done with explicit consideration of each biochemical entity volume (as we will see in chapter 5.3).

Our model represents a simple genetic regulatory circuits. It is well-know that gene expression can be regulated by a different number of factors, which can either inhibit or promote gene expression under different conditions [82]. Moreover, since regulatory factors are present in low quality, noise plays a relevant role in gene expression [83]. Furthermore, the explicit modeling of nucleus space allowed us to consider the crowding effect, in fact, the presence of chromatin (the substance in which is included DNA) fill significant amount of space, hampering the movements of transcription factors in the nucleus [84].

In the model, the nucleus is represented as a two-dimensional grid composed of finite number of squared compartments of the same size (Figure 3.12). Six biochemical entities are considered in the model, two genes $G_1$ and $G_2$, whose location is fixed, and four factors $F_1$, $F_2$, $F_3$, $F_4$ regulating the gene expression, which are instead free to diffuse in the nucleus space. The compartments in which genes are placed, together with with adjacent compartments, have a lower free space in order to consider the chromatin crowding effect.

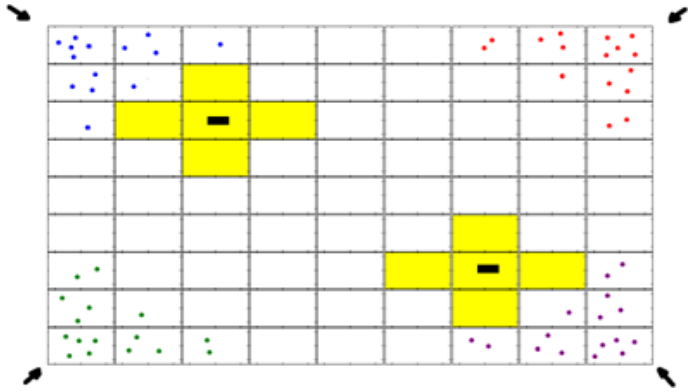

Figure 3.12: **Nucleus space definition**

Schematic representation of the space domain. The 2D-grid of compartments represents a section of the nucleus that contains the two genes G1 and G2. Regulatory factors (filled circles) diffuse within this environment. Compartments filled in yellow have a lower free space. The representation is not in scale with actual sizes used in simulations.

Each regulatory factor interact in a specific way with each gene, thus the overall system behavior depends from the regulatory network resulting from all the possible interactions (Figure 3.13).

The evolution of a system so defined is influenced by both spatial and temporal dimensions affecting the diffusion of regulatory factors.

Sixteen rules have been codified in order to represent all the possible interactions between the factors and the genes, moreover, there are eight rules describing the complexes formation, and eight rules describing the corresponding dissociations (Table 3.7). One way to study the evolution of the defined system is to analyze state of a gene over time, more specifically a gene can be in three different states:

- Active: when it is bound to an activator
- Inhibited: when it is bound to an inhibitor
- Free: when no bounds with regulatory factor are present

Simulations results considering random initial spatial configuration in both spatiality (i.e., initial factors localization) and dynamism (i.e., diffusion constants and complexes formation/degradation kinetics) show that genes fluctuate among all the three possible states (Figure 3.14).

We observed that, the higher is the stochastic constants associated with formation and dissociation of $F_i :: G_j$ complexes, the higher is the variability of gene states (i.e., more state transitions in the same interval of time). These transitions are also influenced by the number of molecules of each regulatory factor available in the system, that is, the more the factors are the more a gene fluctuates (Table 3.8).

| Description | Rule |
|---|---|
| Activation of $G_1$ mediated by $F_1$ | $F_1 + G_1 \rightarrow G_1^+$ |
| Dissociation of $F_1$ from $G_1$ | $G_1^+ \rightarrow F_1 + G_1$ |
| Activation of $G_2$ mediated by $F_1$ | $F_1 + G_2 \rightarrow G_2^+$ |
| Dissociation of $F_1$ from $G_1$ | $G_2^+ \rightarrow F_1 + G_2$ |
| Inhibition of $G_1$ mediated by $F_2$ | $F_2 + G_1 \rightarrow G_1^-$ |
| Dissociation of $F_2$ from $G_1$ | $G_1^- \rightarrow F_2 + G_1$ |
| Inhibition of $G_2$ mediated by $F_1$ | $F_2 + G_2 \rightarrow G_2^-$ |
| Dissociation of $F_2$ from $G_2$ | $G_2^- \rightarrow F_2 + G_2$ |
| Activation of $G_1$ mediated by $F_3$ | $F_3 + G_1 \rightarrow G_1^+$ |
| Dissociation of $G_1$ from $F_3$ | $G_1^+ \rightarrow F_3 + G_1$ |
| Inhibition of $G_2$ mediated by $F_3$ | $F_3 + G_2 \rightarrow G_2^-$ |
| Dissociation of $F_3$ from $G_2$ | $G_2^- \rightarrow F_3 + G_2$ |
| Inhibition of $G_1$ mediated by $F_4$ | $F_4 + G_1 \rightarrow G_1^-$ |
| Dissociation of $G_1$ from $F_4$ | $G_1^- \rightarrow F_4 + G_1$ |
| Activation of $G_2$ mediated by $F_4$ | $F_4 + G_2 \rightarrow G_2^+$ |
| Dissociation of $F_4$ from $G_2$ | $G_2^+ \rightarrow F_4 + G_2$ |
| Diffusion from $(x, y)$ to $(x + nx, y + ny)$ | $F_z^{x,y} \rightarrow F_z^{x+nx,y+ny}$ |

Table 3.7: **Reaction rules gene network model**

List of rules defined for reaction and diffusion processes. Reactions occur in compartments where $G_1$ and $G_2$ are placed, while diffusion occur in all compartments; $(x, y)$ are compartments coordinates, $n \in \{-1, 0, 1\}$ and $z \in \{1, 2, 3, 4\}$.

| Regulatory Factors | Reaction Constants | Diffusion Constants | $G_1$ state variation |
|---|---|---|---|
| 5 | $10^3$ | $10^5$ | 244 |
| 5 | $10^2$ | $10^5$ | 26 |
| 5 | $10^1$ | $10^5$ | 3 |
| 10 | $10^3$ | $10^5$ | 635 |
| 10 | $10^2$ | $10^5$ | 89 |
| 10 | $10^1$ | $10^5$ | 11 |
| 20 | $10^3$ | $10^5$ | 987 |
| 20 | $10^2$ | $10^5$ | 113 |
| 20 | $10^1$ | $10^5$ | 16 |

Table 3.8: **Dynamics of gene state variations**

Analysis of $G_1$ state variation over single simulation performed changing the initial number of regulatory factors.

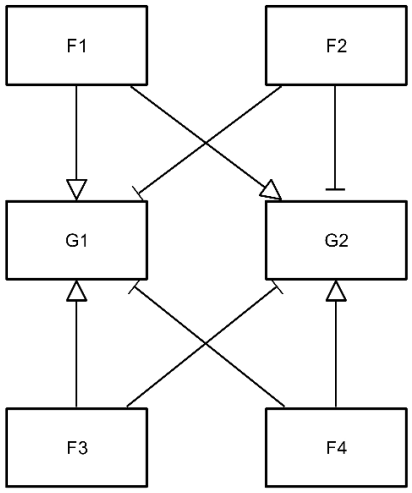


Figure 3.13: **Gene network model**
Activity flow in the gene regulatory network. Representation of the gene regulatory network using the SBGN (Systems Biology Graphical Notation).

Instead, by considering several model executions while keeping the same initial spatial configuration, it is possible to observe a convergence of the gene state probability over time (i.e., the probabilities evolution eventually reach a steady state condition). The convergent values (i.e., steady state level) are affected by the values of the stochastic constants describing the system and by the number of regulatory factors. For instance, lower amounts of regulatory factors and higher complex dissociations constants favor for activated and inhibited states, on the contrary, higher regulatory factors amounts and lower dissociation constants favor the free state.

However, it has been found that the system spatiality plays a major role not in affecting the final equilibrium level of gene state probabilities, but instead, in their evolution. In particular, a gene state probability, before reaching a steady state condition, is more likely to pass through different phases according to the initial space localization of the regulatory factors (Figure 3.15).
More specifically, a gene is prone to pass through a transient phase in which it is more like to be in a state depending by the closest regulatory factor (i.e., activate or inhibited according the presence in the nearest environment of an activator rather than an inhibitor).

This results can be extremely meaningful, in fact, the matching between the localization of a gene in the chromatin, and the pore position in the nucleus membrane (through which a transcription factor enters in the nucleus), may explain the dynamics of gene expression [85].

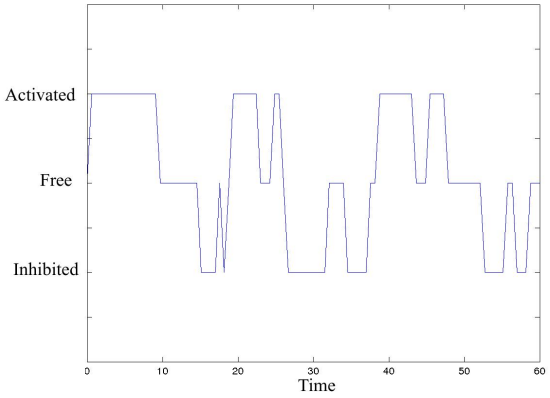


Figure 3.14: **Gene state fluctuations**

$G_1$ state over a single simulation, the gene basically fluctuates among all the three potential states, activate, inhibited and free.

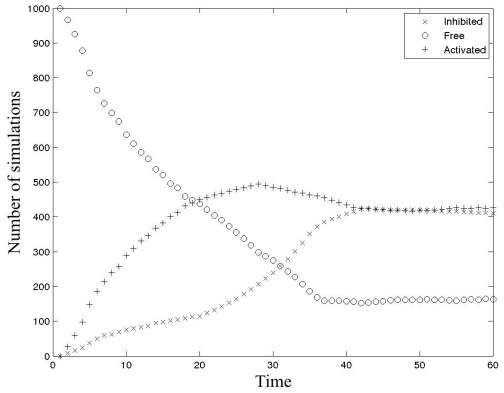


Figure 3.15: **Gene state probabilities over time**

Time evolution of G1 in 1000 simulations in which as initial condition an activator is located in the closest corner while an inhibitor is placed in the distant one. G1 starting from the free state passes through a transient phase in which the activated state becomes the more likely state before reaching a condition in which the activated and the inhibited state can equally occur.

Our study shows the sensitivity of the system dynamics with respect to the spatial configuration, thus corroborating the hypothesis that explicitly consider spatial dimension in the analysis of biochemical systems can disclose properties that otherwise would be neglected.

## 3.5 System response to random perturbations over time

The analysis of the relationship between the input and the output of a model is usually performed considering how the variation of the initial configuration, which could be the initialization of some state variables (e.g., molecular concentrations), or adjustable parameters (e.g., kinetic constants), affect the model behavior. However, there might be cases in which modelers are interested on how a system responds to perturbations occurring over a dynamic process. Regarding this issue the literature is still poor, although, an interesting approach, named *impulse parametric* SA has been prosed for the study of perturbations occurring during simulation of ordinary differential systems [86].

We studied the effect of real time perturbations on a system dynamics. To this end, we considered a model developed exploiting the reaction system framework, a simulation framework allowing for the explicit consideration of external variable perturbations on a system [87]. Reaction systems are in effect formal models based on the inhibition and facilitation of biochemical reactions underlie biological processes [88]. Noteworthy, in reaction systems framework there exists a specific model element representing an external environment with respect to the actual system, this makes reaction systems particularly suitable for the analysis of real-time perturbations.

Formally a reaction systems is an ordered pair $\mathcal{A} = (S, A)$, where $S$ is a finite set of elements, called "entities", and $A$ is a finite set of "reactions" in $S$. The set $S$ is called the "background set" of $\mathcal{A}$; its elements represent the molecular species (e.g., atoms, ions, molecules) in an environment where the current entities change over time.

A reaction in $S$ is an ordered triplet $a = (R_a, I_a, P_a)$ of non empty and finite sets, such that $R_a, I_a, P_a \subseteq S$ and $R_a \cap I_a = \emptyset$. The sets $R_a$, $I_a$ and $P_a$ are called, respectively, the "reactant set", the "inhibitor set", and the "product set". A reaction is enabled (i.e, it occurs), only if in a given state, which is represented by a set, there are all the required reactant(s) and none inhibitor(s). The reaction firing leads to the production of all its products in the successive state. Two key properties of reaction systems are permanency and absence of concurrency. Permanency property implies that if an element is not a reactant for any enabled reaction, then it ceases to exist and will not appear in the next state of the system, while absence of concurrency implies that a reactant can be used by all the reactions in which it is involved, without any limiting effect due to its availability. Furthermore, another important property is that in reaction systems there is no counting, therefore, a defined biochemical entity either exist or does not exist in a specific system state. Given a reaction system $\mathcal{A}$, it is possible to define a dynamic behavior through the notion of an interactive process. Informally, the process is a sequences of states, where each state is a subset of $S$ and it depends from the result of the reactions previously enabled plus the current context set $C \in S$ which can be used to represent the input element(s)

form the environment.

We developed a model of lac operon of *Escherichia coli*, a genetic system involved in the regulation of lactose metabolism [89]. An operon is a functioning unit of genomic DNA under the control of the same signaling process. The lac operon is constituted by three adjacent structural genes and some regulatory components (Figure 3.9).

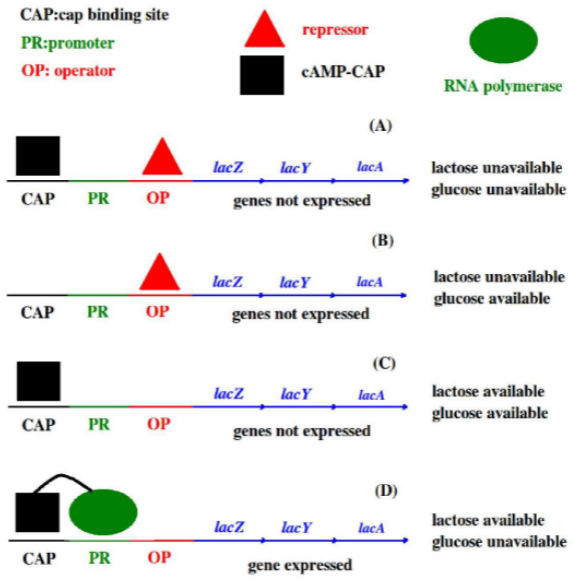


Figure 3.16: **Lac operon**

Regulation of the *lac* operon expression when the environment provides: (A) neither glucose nor lactose; (B) only glucose; (C) both lactose and glucose; (D) only lactose.

The three genes are called lacZ, lacY and lacA: they encode for two enzymes (hereby denoted Z and A) and a transporter (Y) involved in the digestion of lactose. The gene expression depends from a complex biochemical system that starts with the binding of RNA-polymerase (an enzyme) with the promoter (a sequence of DNA upstream the gene). In lac operon, an integrated regulation depending on the available nutrients affect this process. More specifically, is possible to distinguish the following key regulatory elements: (I) A gene, called *lac*I, that encodes for a repressor protein (I); (II) A segment of DNA, named *operator* (OP) upon which the inhibitor can bind forming a complex (I-OP) that blocks gene expression; (III) A short DNA sequence, called CAP-binding site, that, when bound to a specific molecular complex (formed by a protein called CAP and a signal molecule named cAMP) promotes the gene expression.

The regulation is carried out by two control mechanisms, one depending by the presence of lactose and one depending by the presence of glucose, which is a more readily and thus preferred source of energy for cellular metabolism. The first control mechanism exploits the repressor protein I, in absence of lactose the repressor I bind the operon, thus preventing gene expression. Instead, in presence of lactose, a metabolite, named allolactose is formed, and it has the property to prevent the binding between the repressor and the operon,

thus allowing gene expression to occur. The second control mechanism exploits the signal molecule cAMP (whose concentration is inversely related to the one of glucose), to greatly increase lac operon expression in absence of glucose by the interaction between cAMP and CAP, a specific DNA sequence upstream the promoter.
Therefore, four conditions may occur:

1. Neither lactose nor glucose are available: the formation of I-OP prevents gene expression

2. Lactose unavailable and glucose available: the formation of I-OP prevents gene expression

3. Both lactose and glucose are available: the absence of the cAMP-CAP complex which makes the interaction between RNA polymerase and the promoter unstable, prevents a proper gene expression

4. Lactose available and and glucose unavailable: the contemporary absence of inhibitor I and the presence of cAMP-CAP enables a proper gene expression

Two indirect elements related in the regulation of lac operon are also crp and and cya which are involved respectively with CAP and cAMP production, and therefore with the stimulation of gene expression occurring in absence of glucose.

In order to describe the lac operon we defined a reaction system $\mathcal{A}_{lac} = (S, A)$. The background set $S$ represents the main biochemical components involved in the genetic system, and it is defined as follow:

$$S = \{lac, Z, Y, A, lacI, I, I\text{-}OP, cya, cAMP, crp, CAP,$$
$$cAMP\text{-}CAP, lactose, glucose\},$$

The set $A = (a_1, \ldots, a_{10})$ represents all the reactions within the system, each reaction is associated with a specific biochemical event (Table 3.9).

To study the system sensitivity to the different nutrients (i.e., lactose and glucose) over time we define a default condition mimicking the physiological condition and at each iteration (i.e., time step), then we randomly added from the context (i.e, environment), a potential set of nutrient(s). The default condition DC=$\{lac, lacI, I, cya, cAMP, crp, CAP\}$ is composed by the elements normally present in the lac operon gene netwtork, while the context set C can be respectively: $\emptyset$, $\{glucose\}$, $\{lactose\}$, or $\{glucose, lactose\}$. As a target output we observed the system state and more specifically we focused on the evaluation of lac operon expression.

The analysis of system dynamics show that the expression of lac operon occurs only two time steps afterward a context, in which only lactose is present (Table 3.10).

| Reaction | Definition | Process |
|---|---|---|
| $a_1$ | $(\{lac\},\{\dots\},\{lac\})$ | *lac* operon duplication |
| $a_2$ | $(\{lacI\},\{\dots\},\{lacI\})$ | repressor gene duplication |
| $a_3$ | $(\{lacI\},\{\dots\},\{I\})$ | repressor gene expression |
| $a_4$ | $(\{I\},\{lactose\},\{I\text{-}OP\})$ | regulation mediated by lactose |
| $a_5$ | $(\{cya\},\{\dots\},\{cya\})$ | *cya* duplication |
| $a_6$ | $(\{cya\},\{\dots\},\{cAMP\})$ | *cya* expression |
| $a_7$ | $(\{crp\},\{\dots\},\{crp\})$ | *crp* duplication |
| $a_8$ | $\{crp\},\{\dots\},\{CAP\})$ | *crp* expression |
| $a_9$ | $(\{cAMP,CAP\},\{glucose\},\{cAMP\ mbox{-}CAP\})$ | regulation mediated by glucose |
| $a_{10}$ | $(\{lac,cAMP\text{-}CAP\},\{I\text{-}OP\},\{Z,Y,A\})$ | *lac* operon expression |

Table 3.9: **Lac operon reaction system**

List of the reactions. With dots is indicated the $\emptyset$ set.

This behavior depends by the fact that reaction $a_{10}$, which is the one generating *lac* operon expression, is involved in a negative feedback mechanism with reaction $a_4$ (activated by glucose) and in a positive feedback mechanism with reaction $a_9$ (activated by lactose). In fact, since the enabling of reaction $a_{10}$ requires the simultaneous inactivation of $a_4$, reaction $a_{10}$ will be enabled only one step ahead of the context state that provided only lactose, leading to the production of the final protein products (i.e., $Z, Y, A$), two steps ahead of that state.

The delay between the presence of the only lactose, as energy source in the environment and the lac operon expression is consistent with the natural condition in which a cell reacts to a modified environment only after a proper adaptation is occurred. Another interesting property regarding the input-output relation (i.e., context set and lac operon expression), is the presence of hysteresis in the dynamics, that is the dependence of the system state not only from the current condition but also the past environmental condition. In fact, when lactose is the only element in the context, two steps ahead lac operon will be expressed even if in the in between step lactose disappears and glucose becomes available.

Therefore, according to our study, reaction systems, by explicit considering real time perturbations in the simulation framework, can be a profitable solution to study the evolution of system sensitivities over dynamic processes.

| **Index** | **Context set $C_n$** | **Result set $D_n$** | **Lac expression** |
|---|---|---|---|
| 0 | DC | $\emptyset$ | no |
| 1 | $\{glucose\}$ | DC $\cup \{I\text{-}OP, cAMP\text{-}CAP\}$ | no |
| 2 | $\{glucose\}$ | DC $\cup \{I\text{-}OP\}$ | no |
| 3 | $\{glucose\}$ | DC $\cup \{I\text{-}OP\}$ | no |
| 4 | $\{glucose\}$ | DC $\cup \{I\text{-}OP\}$ | no |
| 5 | $\{glucose\}$ | DC $\cup \{I\text{-}OP\}$ | no |
| 6 | $\{glucose, lactose\}$ | DC $\cup \{I\text{-}OP\}$ | no |
| 7 | $\{glucose, lactose\}$ | DC | no |
| 8 | $\{glucose, lactose\}$ | DC | no |
| 9 | $\{glucose, lactose\}$ | DC | no |
| 10 | $\{glucose, lactose\}$ | DC | no |
| 11 | $\{lactose\}$ | DC | no |
| 12 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP\}$ | no |
| 13 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 14 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 15 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 16 | $\emptyset$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 17 | $\emptyset$ | DC $\cup \{I\text{-}OP, cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 18 | $\emptyset$ | DC $\cup \{I\text{-}OP, cAMP\text{-}CAP\}$ | no |
| 19 | $\emptyset$ | DC $\cup \{I\text{-}OP, cAMP\text{-}CAP\}$ | no |
| 20 | $\emptyset$ | DC $\cup \{I\text{-}OP, cAMP\text{-}CAP\}$ | no |
| 21 | $\{lactose\}$ | DC $\cup \{I\text{-}OP, cAMP\text{-}CAP\}$ | no |
| 22 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP\}$ | no |
| 23 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 24 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 25 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 26 | $\{glucose, lactose\}$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 27 | $\{glucose, lactose\}$ | DC $\cup \{Z, Y, A\}$ | yes |
| 28 | $\{glucose, lactose\}$ | DC | no |
| 39 | $\{glucose, lactose\}$ | DC | no |
| 30 | $\{glucose, lactose\}$ | DC | no |
| 31 | $\{lactose\}$ | DC | no |
| 32 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP\}$ | no |
| 33 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 34 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 35 | $\{lactose\}$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 36 | $\{glucose\}$ | DC $\cup \{cAMP\text{-}CAP, Z, Y, A\}$ | yes |
| 37 | $\{glucose\}$ | DC $\cup \{Z, Y, A, I\text{-}OP\}$ | yes |
| 38 | $\{glucose\}$ | DC $\cup \{I\text{-}OP\}$ | no |
| 39 | $\{glucose\}$ | DC $\cup \{I\text{-}OP\}$ | no |
| 40 | $\{glucose\}$ | DC $\cup \{I\text{-}OP\}$ | no |

Table 3.10: **Dynamics of lac operon reactions system**

Example of a random dynamic of lac operon reaction system. The context represent the potential nutrient environment and its non predictable modification over time.

# Chapter 4

# Input space and output dynamics mapping

This part of the thesis concerns the analysis of an innovative strategy for the mapping between the input factor space and the different qualitative dynamics that a model can reach. Traditionally, SA has been focused in the computation of indexes able to quantitatively analyze a property representing an output variability (e.g., variance, derivatives), while few investigations have been carried on regarding the qualitative analysis of the model behavior. A major issue in the development of qualitative SA is the coupling, between a "high level" issue, as the discrimination between completely different output modes, with a "low level" issue, as the computation of an automatic index discerning among different output trends. The analysis of the relationship of between system parameters, environmental variable, and phenotypic traits, is indeed a common problem in the study of biological systems.

Find the boundaries within the input space representing the transition between different dynamic behaviors, can help in understanding the system regulatory mechanisms and may support the developing of control strategies to modify the system behavior. The problem of mapping the input space with the output dynamics encompasses two sub-problems, which are on the one hand the definition of a quantitative measure discerning distinct qualitative dynamics, and from the other hand, the identification of the boundary along the identified behavioral regions within the input space.

In our study, we developed a pipeline for the discrimination of output dynamics based on cluster analysis of model outputs retrieved changing model configurations (i.e., exploring the input space), integrated with a sampling strategy aimed at the identification of dynamics transitions sectors.

In particular, we focused on two main types of dynamics which are widespread in biochemical systems, the steady state condition and the oscillatory behavior. Taken together these two dynamic modes represent the majority of the known biological processes. In effect, the

proper functioning of biological systems usually depends from the attainment of a biochemical equilibrium, which in turn is associated either to the preservation (i.e., steady state) or the periodic change (i.e., oscillation) of the system properties. Noteworthy, there exist systems which can reach either steady state dynamics and oscillatory regimes according to the activity of some regulatory mechanisms. More specifically the establishment of oscillation is often under the control of feedback loops present in the system network. We applied our methodology to two well-known biochemical models, displaying this kind of behavior (i.e, characterized by the attainment of either steady state or oscillatory regimes). The exploitation of the proposed methodology produced a robust mapping of the input space with the system dynamics, even in the presence of limited amount of noise (introduced to test the classification robustness to noisy data).

Hereinafter, in section 4.1 we describe the method for dynamic classification, while in section 4.2 we explain the algorithm for the boundary detection, finally in section 4.3 we show the results ontained in the analysis of the two computational models used as a case studies.

## 4.1 The case of discrimination between steady state and oscillatory dynamics

A number of different approaches have been proposed to detect the presence of oscillations in vector data (e.g., time courses). However, the majority of the detection techniques have been developed in context in which it is present an a priori-knowledge about the existence of oscillations. For instance, several applications are normally exploited everyday in all the fields dealing with signaling processing analysis. These techniques, on the one hand are structured to be robust also in presence of noise, but on the other hand they have the drawback not be able to provide a general "high level" classification of the output trend. In fact, the majority of methodologies for oscillatory analysis are based on the exploitation of frequency component analysis (e.g., Fourier analysis) [90, 91, 92, 93]. As a consequence, they are extremely powerful in provide even detailed information (e.g., maximum or minimum frequency observed) but do not respond to general question, such as if the output is increasing or decreasing, or if a specific frequency component is indeed due to an actual oscillation rather than it is simple noise. However, it is possible to argue that for a deterministic system, the discrimination between steady state and oscillatory dynamics is in effect a trivial problem, for instance, once a model has reach the stability condition, the variance will be equal to zero in the first case, while will be different from zero in the second case. Nevertheless, as soon as some noise is introduced in the system the deployment of a binary classification for qualitative discrimination becomes a tricky problem, in fact noise hampers the presence of regular oscillation and alter the stability of

a steady state, thus making the classification arbitrary.

In our pipeline, we propose the use of autocorrelation function in order to estimate a discriminatory index to distinguish between steady states and oscillatory conditions also in presence of noise, without introducing an a-priori threshold on some frequency component. The method is instead based on the analysis of the available time courses and therefore it has a widespread applicability, from the study of model simulated data to the analysis of experimental retrieved data.

More specifically, given a time course $X_i = x_1, \ldots, x_n$, we derive the autocorrelation sequence $C$ as follow:

$$c_k = \frac{1}{n \times S^2} \sum_{t=1}^{n-k} (x_t - \bar{x})(x_{t+k} - \bar{x}) \quad \text{with} \quad k = 0, \ldots, n-1 \tag{4.1}$$

where $\bar{x}$ and $S^2$ denote respectively the mean and the standard deviation of the time course data.

Then, the quantity $F$ is defined as the interior maxima of the autocorrelation sequence $C$, that is:

$$F_{X_i} = max\{c_k\}_2^{n-1} : c_{k-1} < c_k > c_{k+1} \tag{4.2}$$

Noteworthy, there can be conditions in which no maxima are present (e.g., if $X_i$ is a constant vector), or also a negative maxima can be found (e.g., $X_i$ is a fast damped oscillation), in such cases $F_{X_i}$ is set equal to 0.

Since the defined autocorrelation sequence spans between $-1$ and 1, the value of the interior maxima $F$ is expected to be significantly higher than 0 in case of a oscillatory regime, while it is expected to be close to 0 in case of a steady state. Hence, by comparing $F_{X_i}$ with a sensitive threshold $T$ in between 0 and 1 a discrimination between an oscillatory regime and steady state can be computed. To this end, we defined an index $S = F - T$ whose sign permits a binary classification, that is:

- $S_{X_i} => 0$ the time course $X_i$ is classified as oscillatory
- $S_{X_i} < 0$ the time course $X_i$ is classified as steady state

The value of $S$ can be seen as a threshold separating steady states affected by noisy fluctuations from noisy oscillation regimes. As such, the more is $|S|$ the more is the reliability that the time course has been properly classified. In effect, the more noise is present in the system the more critical became the discrimination between the two dynamic trends. In fact, on the one hand the irregularities of the oscillations lead to a decrease of the autocorrelation, while on the other hand, the noisy fluctuations that affect the steady state condition lead to a random increase the autocorrelation values.

Since noise levels alongside with the presence potential different frequencies can affect

autocorrelation, we proposed a method that basing on the available information (i.e., time courses data), is able to perform a classification suited for the specific case study under analysis. The method, in effect, computes a classificatory autocorrelation threshold according to the available time courses. More specifically, given a set of time courses $X = X_1, \ldots, X_N$, characterized by the presence of either steady states and oscillatory dynamics, it possible to perform a cluster analysis to partition the different dynamics (i.e., time courses), according to the corresponding set $F = F_1, \ldots, F_N$ (in turn derived by the analysis of the autocorrelation sequences). A possible solution is the exploitation of $k$-means clustering algorithm (in which the number of cluster is set to two), the expected result will be the partition of $F$ into two sets $F_o \subset F$ and $F_s \subset F$, where $\bar{F_o} > \bar{F_s}$. The $F_o$ cluster (higher autocorrelation), is associated with oscillatory dynamics, while the $F_o$ cluster (lower autocorrelation), is associated with steady state dynamics. Then, it is possible to define the extreme quantities $F_{max} = min(F_o)$ and $F_{min} = max(F_s)$, representing the lower $F$ value of the time course classified as oscillatory, and the higher $F$ value of the time course classified as steady state. Finally, the threshold for the classification of dynamic $T$ is defined as:

$$T = \frac{F_{max} + F_{min}}{2} \tag{4.3}$$

that is, it is defined as the middle value in between the extreme quantities of the two clusters classifying the output modes.

## 4.2 Partition of the input space

The identification of boundary between different regions in the input space associated with different qualitative behaviors of the model output, is usually performed by employing Monte Carlo procedures for evaluating of a given property state. The rational behind these approaches is to iteratively find subsets (i.e., subregions) in the input space in which a property $\phi$ is satisfied or violated [52]. In practice, this kinds of algorithms partition the input space in larger regions where the satisfaction of violation of the $\phi$ is robust and smaller regions alongside the borders between satisfaction and violation [51]. A major issue with the application of such kinds of techniques is that the refinement process implies that the number of input space partitions increases exponentially with the number of input factors, thus it becomes unfeasible for large models.

In our pipeline we instead propose the deployment of an approach that directly parameterizes the, yet unknown, boundary by the zero level-set of a polynomial function, using statistical inference on available data to identify the coefficients of the polynomial [94]. In our framework, the boundary is assumed to be a smooth function $\Gamma(x) = 0$, which can be well approximated by polynomials. The objective is to compute the coefficients of a multivariable polynomial approximating $\Gamma$.

The approximation is of the form:

$$f(c,x) = \sum_{k=1}^{d} c_k \mathcal{L}_k(x) \tag{4.4}$$

where each $c_k$ is an input value in the input space $\mathcal{C}$, while $\mathcal{L}_k$ is the Legendre sparse grid bases. The use of Legendre sparse grid as basis functions rather than classic monomial bases is because it has proven to produce more stable result [95]. In other words, the boundary of interest are modeled as the set $\{x \in \mathcal{X} : f(c,x) = 0\}$, where the input vector $c$ needs to be estimated from available data.

The idea is to start with some sampling schemes to collect data for inference. At each point in a sequence $xi$, representing different input vectors, a binary classification is retrieved according to $y_i = \text{sign}(\Gamma(x_i))$ (i.e, polynomial sign).
According to the binary data, a probability distribution $\pi$ over $C$ is derived as:

$$\pi_n(c) \sim \exp\left[-\sum_{i=0}^{n} (y_i - \text{sgn}[f(c,x_i)])^2\right] \tag{4.5}$$

where $\pi_n(c)$ corresponds to the likelihood that the zero level-set of the polynomial $f(c,\cdot)$ is the boundary $\partial\Gamma$.

The expected coefficient $\bar{c}_n$ with respect to $\pi_N$ is computed using a Monte-Carlo Markov Chain method and $f(\bar{c}_n,\cdot)$ is used as the approximation of $\Gamma$. When the collected data $x_i$, $1 \leq i \leq m$ satisfies certain patterns, the zero level set of $f(\bar{c}_n,\cdot)$ converges to the true boundary $\partial\Gamma$. The algorithm can be run using two different settings:

1. Sparse grid sampling method: $x_i$, $1 \leq i \leq n$ is the grid points of a multi-level multidiensional sparse grid.

2. Sequential sampling method: data is collected sequentially, where the next data point $x_{n+1}$ is taken at the point where the maximum of response variance with respect to $\pi_n$ is achieved.

$$x_{n+1} = \arg\max_{x \in \mathcal{X}} Var_{\pi_n}[\text{sign}(f(c,x))] \tag{4.6}$$

In general, the sparse sampling method can quickly identify the general structure of the boundary. Instead, the sequential sampling after a burn-in period which can be high computational demanding, by selecting points only near the boundary of interest can significantly reduce the cost to refine the boundary. Therefore, a proper solution could be an integration of both methods of sampling, first identifying the general behavioral regions using sparse grid and then refining the borders using the sequential method.

## 4.3 Behavioral classification: two real cases

We applied our pipeline on two well know biological models, the repressilator system and a model of a signaling process in yeast. The former is a model of gene network, while the latter is a model of a biochemical pathway in *Saccharomyces cerevisiae*, in both models, the presence of negative regulatory feedbacks may lead to the occurrence of oscillations.

### 4.3.1 The repressilator system

The repressilator model is a milestone in synthetic biology, a discipline that aims at the engineering of biological systems displaying useful purposes [96]. More specifically, the repressilator system was designed in order to build a novel gene network exhibiting a stable oscillatory behavior [97]. The gene network is composed by three genes which inhibit the expression of each other by mean of the respective gene products (Figure 4.1).

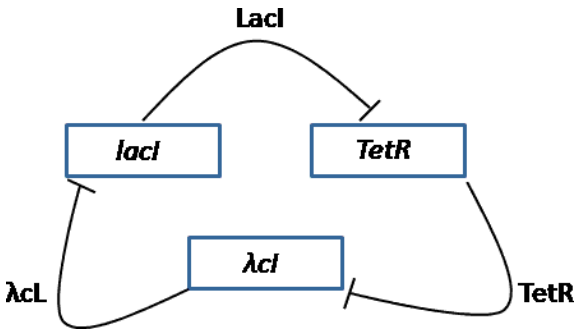


Figure 4.1: **Repressilator model**
The repressilator system consists of a set of three genes ($lacI, tetR, \lambda cI$) whose protein products (LacI,TetR,λCI) act as inhibitors of the successive gene in the set.

The dynamic of the system is described by six coupled first order differential equations, a couple for each gene, one representing the transcription process and one one representing the protein product degradation. Different model parameters (e.g., transcription rate, initial concentrations), affect the model dynamic trends. In particular, several studies have proved that according to the model configuration, two kind of solutions are possible: the system may converge toward a stable steady state, or it can reach a sustained limit-cycle oscillations [98, 99, 100, 101].

The analysis has been performed considering as a target dynamic the trend of *LacI* concentration (*[LacI]*) over simulation time of 5000 time unit (t.u.). In reality, since the gene network structure leads to the attainment of coordinate dynamics among all the the different elements, each gene product may be chosen as representative of the system dynamic. Instead, the selection of the simulation time was arbitrary chose so to allows the system to have enough time to reach the stability condition, thus enabling the classification of output dynamics. In our test case we considered as input two key system parameters, one

associated with the strength (i.e., speed of transcription) of the promoters fully induced, and the other associated strength of the promoters repressed (by the protein product of the previous gene in the set). Tuning these two parameters corresponds to act on the intensity of the negative feedback regulatory mechanism, that is, the more is the ratio between the induced and repressed promoter, the more is the intensity of the feedback activity. Therefore, the boundary between regions within the input space associated with oscillatory and the steady state dynamics has been detected considering the two dimensional parameter space, given by the strengths of the repressed and induced promoter. Aside the promoters strengths which haven been allowed to vary one order of magnitude above and below the initial values (set to 0.003 for the induced state and 0.08 for the repressed state), the other parameters values have been kept fixed at the default values (i.e., the ones of the original published model).

The analysis of the autocorrelation was indeed able to discern between clear cluster behavior, associated with higher and lower autocorrelation. The application of our pipeline, based on the threshold identification by mean of the cluster analysis of the autocorrelation sequences coupled with the boundary detection algorithm provided a good estimate the boundary between input factor space determining different output modes (Figure 4.2).

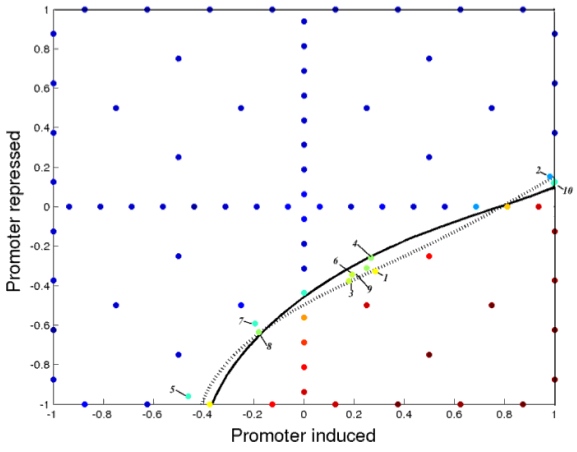


Figure 4.2: **Repressilator boundaries**

Boundary detection Repressilator model: with dashed line the contour identified using a sparse grid sampling consisting of 80 model parameterizations; in bold line the contour identified after the introduction to the previous parameterizations of 10 new model simulations selected according to the sequential sampling method (the numbers indicate the order in which the points have been selected). The color of the points correspond to the normalized value of the classifier index $S$ (red higher, blue lower).

The results confirm that the well-constructed nature of sparse grid points allows the identification of the general structure of the boundary, while the sequential sample quickly refines the boundary contours with high accuracy. The boundary shows that the establishment of the oscillatory regime is favorite for combinations associated with lower activity of the repressed promoter and higher activity of the induced promoter, that is it is supported by

higher feedback activity.

In order to test the robustness of behavioral discrimination with respect to noise, thus testing the reliability of the input space-output dynamics that would be obtained in common noisy analytical contexts (e.g., stochastic simulations, data retrieve from experiments), different levels of Gaussian noise have been introduced in the evaluation of the time courses retrieved by model simulations. The approach we used to consider noise in the system was based on the addiction of a noise vector to the analyzed state variable time course (i.e, *[LacI]*). More specifically, from the retrieved time course vector $Y_i = y_1, \ldots, y_n$, a corresponding noisy vector $Z_i = z_1, \ldots, z_n$ is derived. Different noise amounts have been introduced in the system in a way such that the different elements of the noisy vector are in the form $z_i \sim (0, (x_i g)^2)$, where $g$ represents the noise level, in this way the noise is proportional to the actual values (i.e., simulating experimental errors).

Finally, a noisy time course $Y_i^{noisy}$ is obtained by the combination of the deterministic time course and the noise vector in the following way: $Y_i^{noisy} = Y_i + Z_i$. The introduction of additive Gaussian noise that mimics the experimental noise found in wet-experiment data [102], can be a profitable approach to test the sensitivity of the boundary detection in different analytical contexts.

The results show that for non excessive amount of noise the classification still works properly (Figure 4.3).

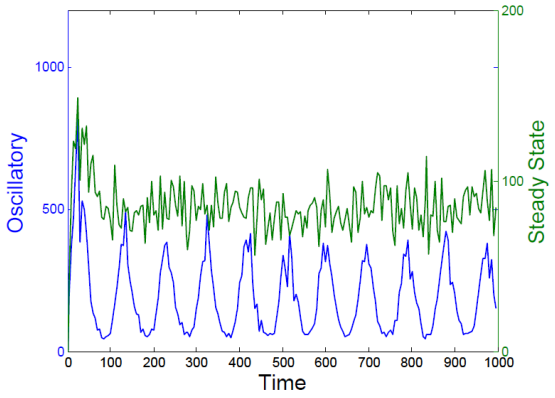


Figure 4.3: **Classification of noisy vectors**

Example of a Time course with S = −0.2, classified as steady state (green), and a time course with S = 0.2, classified as oscillatory (blue).

However, as the level of noise increases the reliability of the boundary detection decreases, this leads to the identification of unstable boundaries (Figure 4.4) The instability of the boundaries has been further proved by the analysis of the variability of the polynomial coefficients (Figure 4.5).

In other words, as the noise increase, according to the given instance under analysis, you can retrieve completely different polynomials separating the different input regions, thus

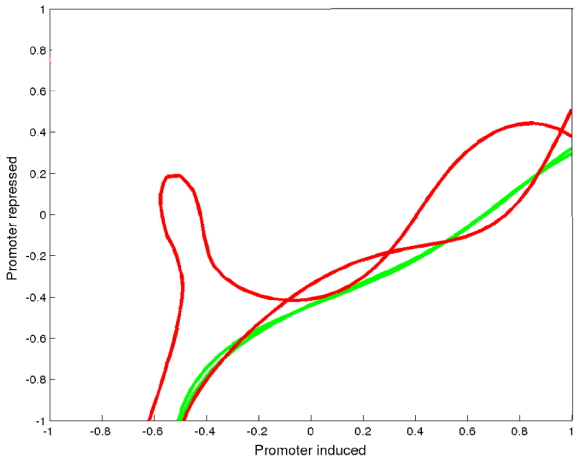


Figure 4.4: **Repressilator boundaries with noise**

Example of two retrieved behavioral boundaries for $g = 0.06$ (green), and for $g = 0.12$ (red).

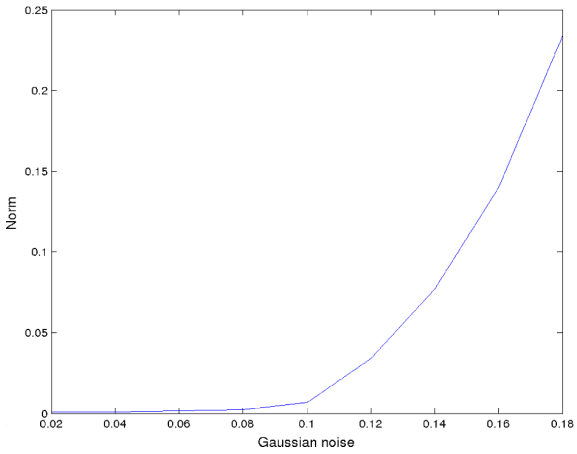


Figure 4.5: **Polynomial coefficients variability**

The relation between the added Gaussian noise g and the variability of the boundary expressed in term of euclidean norm of the variance of the polynomial coefficients (over 100 samples) is non-linear.

the behavioral regions are unstable, therefore, it is no longer possible to perform a clear mapping between the input values and the model dynamics.

### 4.3.2 The Ras model

The model simulates the Ras/cAMP/PKA pathway in *Saccaromyces cerevisiae* [103]. This pathway plays a leading role for the proper cellular functioning, among others, it regulates cell metabolism and cell cycle [104]. The control is mainly exerted by the level of cAMP, whose concentration ([cAMP]) may present an oscillatory trend or hovering around a steady state [105]. Since [cAMP] dynamics are related to the current cellular conditions, the monitoring of its level can help in the comprehension of cellular activity [106]. The model, originally developed for stochastic simulation, can be also simulated using a generalized mass action approach in which the stochastic constants are derived from the conversion of

the stochastic rules to reaction rates. The result of the conversion is the definition of a a system of ordinary differential equations, in which each one describes the concentration of a specific species over time. The model considers the main regulatory factors of the Ras/cAMP/PKA pathway (Figure 4.6), including the negative feedback loop exerted by Ira2, which is believed to be the one responsible of the establishment of oscillatory behavior [107].

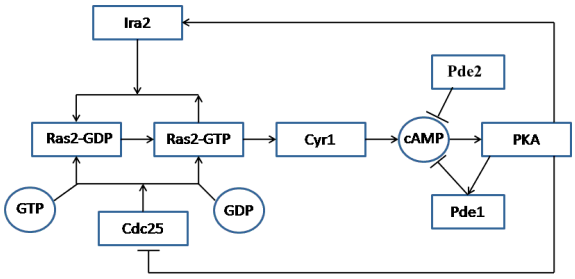


Figure 4.6: **Ras model**
Ras/cAMP/PKA pathway with the corresponding negative and positive regulations.

More specifically, the dynamic of 11 species concentrations is simulated, and previous studies have proven that [cAMP] dynamics (that we chose as output target), according to the regulatory activities, can indeed attain either, a steady state condition or an oscillatory regime [108].

Similarly to repressilator system analysis, an arbitrary interval of time of the selected state variable (i.e., [cAMP]) has been chosen so to ensure the availability of a number of values in the retrieved time courses that would be suffice for recognition of model dynamics (we set a simulation time of 5000 time unit). Regarding the input space we specifically selected three kinetic parameters involved in the regulation of [cAMP], so to study the presence of possible cooperative effects. More specifically, we considered as input factors the kinetic constant associated with:

- Upstream reaction of cAMP (i.e., its production)
- Downstream reaction of cAMP (i.e., its degradation)
- Strength of the feedback loop exerted by Ira2

The initial configuration correspond the values of $2.1e-6$ for upstream reaction, 1 for downstream reaction and 0.25 for Ira2 feedback, the constraints have been defined in one order of magnitude above and the below the initial values, while the other parameters have been fixed to the default values defined in the original model.

The analysis of the autocorrelation was indeed able to discern between clear cluster behavior, associated with higher and lower autocorrelation (Figure 4.7).

The exploitation of our pipeline allowed the identification of an embedded oscillatory region within an input space characterized by steady state output dynamics (Figure 4.8).

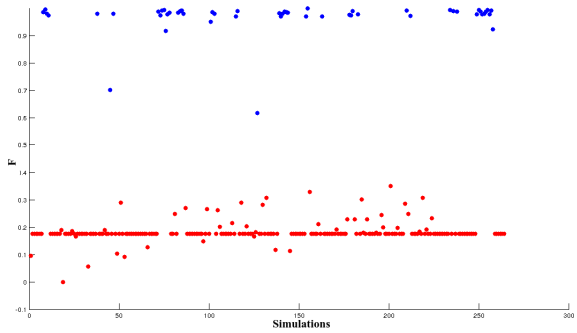


Figure 4.7: **Cluster analysis of autocorrelation coefficients**

The clustering of the $F$ values retrieved from the model simulation launched considering 264 sparse grid points within the input space permits the subdivision of the model outcome in two cluster, one with higher autocorrelation (red points) and one with lower autocorrelation (blue points). Interestingly, some points associated with the steady state condition do not correspond with zero maxima of the autocorrelation, this can be explained by the fact that the model may pass through a transient phase before eventually reaching the steady state condition (i.e., damped oscillations).

The identified boundaries show that the attainment of an oscillatory state is a critical condition which requires a proper coordination levels between the kinetics associated with formation and degradation of cAMP and with the intensity of the feedback activity exerted by Ira2.

Likewise repressilator system, also for the ras model we considered the addiction of different levels noise to the system. Interestingly, we verified that regardless from the amount of added noise, the computed threshold $T$ is not significantly affected, but instead what happen is that, the more is the noise introduced in the system the more the clusters are compressed again each other (but the separation level remain similar). This fact suggests that Ras model is more robust to noise than the repressilator system (at least for the considered input spaces). In fact, the identified borders, considering different levels of added gaussian noise $g$ produces qualitatively similar shapes (Figure 4.9).

### Discussion on pipeline for input space and output dynamic mapping

The proposed framework has been proved to be able to identify regions within the input space associated with different output dynamics. However, different issues emerged throughout the testing of the method on real cases, like the necessity to constrain the analysis according to some a-priori knowledge of the systems (i.e., interval of time to discern model dynamics) and the sensitivity to noise. Despite the limitations, the methods by coupling an autocorrelation analysis to detect the presence of periodicity in the time courses and an algorithm for the boundary identification able to estimate polynomials alongside the dynamics transitions allows the partitioning of the input space according to the associated dynamics modes. Therefore by carefully evaluating the boundary classification when modifying the experimental conditions (e.g., noise levels, simulation time) can be

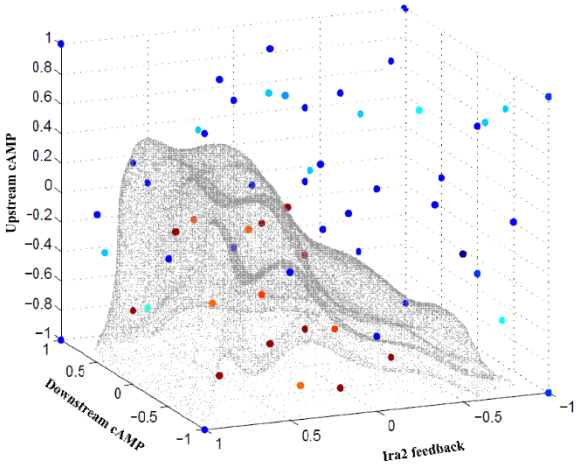


Figure 4.8: **Ras boundaries**

Separation obtained using a grid sampling method of 264 model simulations followed by 30 new realizations selected exploiting the sequential sampling method. The surface divides the region in which an oscillatory behavior has been detected (inside) to the region in which a steady state behavior has been detected (outside). The color of the points is related to the actual value of $S$, from red (higher autocorrelation) to blue (lower autocorrelation).

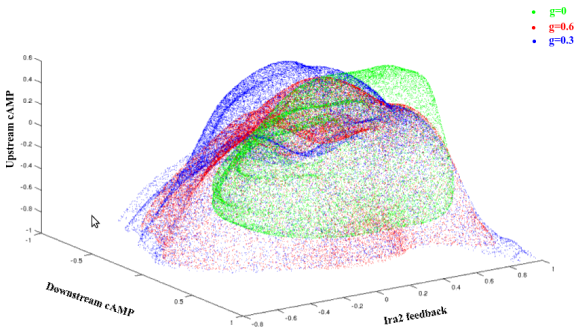


Figure 4.9: **Ras boundaries with noise**

Separations obtained considering different levels of noise. Both the boundaries have been retrieved using a grid sampling method of 264 model simulations.

a profitable approach to study the connection between input values of a model and its tendency to reach either periodic behavior (i.e., oscillations) or convergent behavior (i.e., steady state).

# Chapter 5

# Computational issues

This part of the thesis concerns different computational issues related to the implementation and the simulation of mathematical models of biochemical systems. In particular, we focused our attention to the analysis of stochastic algorithms for simulating the dynamics of biochemical processes. In fact, performing SA of a computational model requires the execution of several model simulations varying the parameter configuration. Moreover, in case of a stochastic model, several instances of the same simulation are generally needed, in order to achieve statistically consistence of the output and therefore of the model behavior. Therefore, the time required by model simulations is often the key parameter in deciding of whether or not a given SA method is applicable. In fact, since biochemical systems models are often characterized by several input factors, and since stochastic simulation algorithms are usually high computational demanding, in many cases the application of SA is de-facto unfeasible.

To tackle this issue, we proposed different methodologies based on the exploitation of adaptive/distributed/parallel approaches, in order to reduce the time required by model executions, so to make models suitable for the application of more extensive SA methods. In section 5.1, we propose a simple methodology for the simulation time in the analysis of a noisy steady state condition. The method exploits an adaptive mechanism, based on the evaluation of the output variance trend to estimate a proper time for the evaluation of the different steady state conditions.
Section 5.2 concerns the employment of distribute computing, to run independent stochastic simulations in an embarrassingly parallel way, in order to reduce the time required to perform SA. Regarding this topic, we show how in the analysis of a stochastic model of bacterial chemotaxis, the use of the European Grid Infrastructure (EGI), allowed us to substantially reduce the time to retrieve the model outputs to compute sensitivity indexes. In section 5.3 we describe revised implementations of a stochastic algorithm for reaction diffusion processes suited for the parallelization on two parallel infrastructures, which are, computer cluster and graphic processing unit. The algorithm encompasses operations

that can be independently computed and synchronization phases requiring communication among the processes: therefore, an efficient implementations implies a proper management of the different algorithmic steps. We show, how by using of our MPI implementation we were able to effectively exploit a cluster to run in parallel the simulations of a simple abstract model of a genetic circuit. Then, using the same case study, we tested our CUDA code obtaining an excellent speed up of model simulations.

## 5.1 Adaptive simulation time for steady state sensitivity analysis

Here, we propose the exploitation of an adaptive simulation time set according to model dynamics, in order to reduce the computation time for steady state condition SA of stochastic simulations. More specifically, we introduced an adaptation mechanism, based on the evaluation of model dynamics, to set the lowest simulation time assuring the attainment of the steady state condition, thus minimizing the time frame in which the model has to be analyzed to retrieve the steady state levels. In general, the introduction of adaptive mechanism can augment the capability to address the complexity present in several application fields and/or develop strategies to improve the computational performance of the system [109]. A proper adaptive mechanism should adapt itself to a variable context while avoiding system failure and achieving the expected performance [110]. Adaptive mechanisms are characterized by four core functionalities:

1. monitor the current state of the system and-or its environment
2. analyze the gathered information from the system's behavior and performance point of view
3. decide which is the proper action to performance
4. possibly perform a change

this operational sequence must be repeated during the software execution. Noteworthy, the correct implementation of adaptivity is strictly related to the characteristics of each case study, thus, a deeper knowledge of the analyzed system is required [111]. Usually, the development of an adaptive mechanism encompasses: (I) definition of the aims that should be achieved by software; (II) device a procedure able to test the current state of the system; (III) Possibly perform an adaptation to manage the changed condition.

The implementation of SA methods require the selection of a proper target variable, which is able to characterize the model behavior, this imply the presence a-priori knowledge of the general model behavior (i.e., model dynamics). However, especially for models of complex systems it may not always be possible to predict the system behavior. As

a consequence, considering the exploitation of adaptive mechanisms could enables the automatic management of variable conditions. The idea is to leave to the analytical the task of monitoring the current state and performing, if needed, an adaptation procedure with the aim to analyze the new modified model behavior in a more efficient way.

To this end, we focused on a typical issue encountered in SA, which is how long simulate a model, to retrieve output values assessing the state condition, when a stochastic simulation framework is exploited. In this work, we carried on a steady state SA of a computational model of the hetero-trimeric G protein cycle in yeast [112]. The model considers the signaling pathways responding to a peptide pheromone in preparation for mating in haploid yeast cell (Figure 5.1).

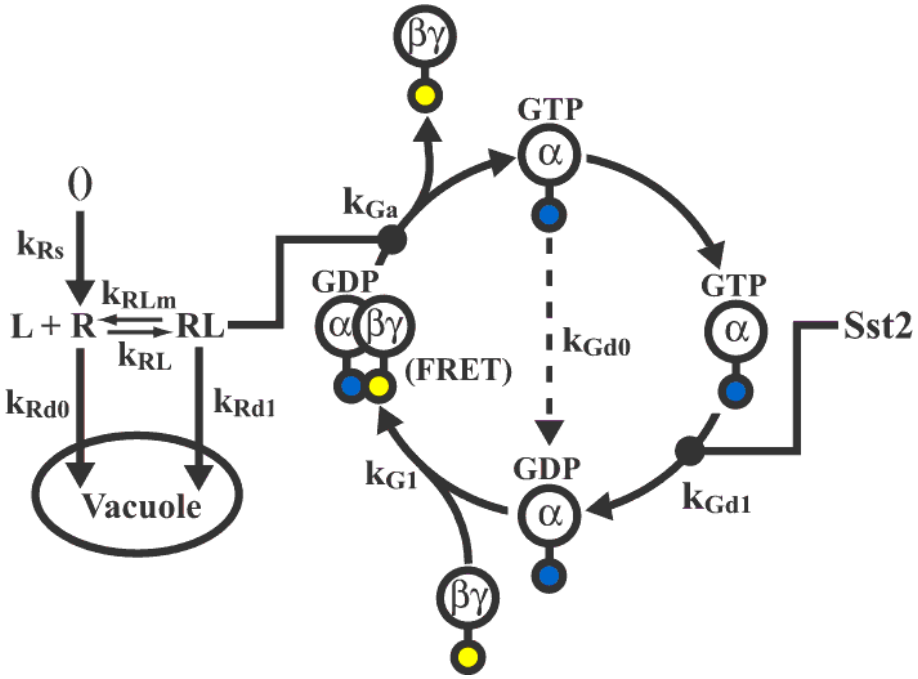


Figure 5.1: **A Model of eterotrimeric G protein cycle**
Reaction diagram of heterotrimeric G protein cycle. The figure is taken from [112].

The model behavior is characterized by the propensity to eventually reach a steady state condition, that is after a transient period of time, the concentrations of some molecular species tend to stabilize around specific values. The presence of this feature is quite common in model of biochemical systems, in fact, it reflects the the attainment of dynamics equilibria, which are typical of several biological phenomena. In effect, when a model is characterized by steady state dynamics, SA is normally aimed at the identification of how input values affect the steady state level.

In such cases, a common question that arises when writing the code for SA is "for how long the model need to be simulated to reach the steady state condition?". In fact, the identification of a proper simulation time allows the setting of a target time point, at which evaluate the output to assure a proper analysis of the model behavior. However,

the simulations of the model using various model configuration mays lead to very different dynamics. For instance, in our test case, the time to reach the steady state condition of receptor-ligand complex (RL) (that we set as target output of SA), is significantly variable in different model configurations (Figure 5.2).

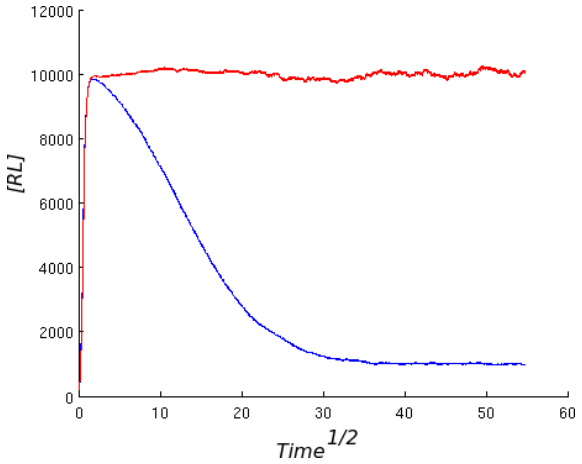


Figure 5.2: **Simulation time for steady state:**
Dynamics of RL molecules number in two model simulations using different values of the kinetic constant associated with the receptor-ligand complex. *the use of the square of time values is to provide a better readability of the different time ranges required to reach the steady state in the two conditions.

As a consequence of the uncertainties related to model dynamics, it is a common practice to consider a large simulation time, which presumably assures the computation of the steady state condition, the main drawback is that a simulation so defined yields to high computation times.

We developed our adaptive approach considering stochastic simulations of G protein model (performed using $\tau$-leaping on MATLAB), which are indeed characterized by similar but not equal dynamics. t Therefore, as common when a model is stochastically simulate it is necessary to repeat a set of model simulations per each model configuration to correctly estimate the model output.

In our test case, we performed a regression based SA between the kinetic constant of the biochemical reaction associated with RL formation (hereinafter named $k$) and RL concentration ([RL]) at steady state (hereinafter named $[RL]_s$). In the case of regression SA considering stochastic simulations, the most straightforward approach to build the regression curve is to launch a batch of model simulations per each considered model configuration, and then keeping the average values as reference values.

Our proposal is to simulate the model for the time that is required by he model to reach the steady state in a specific condition, rather than using a fixed simulation time. The assumptions is that when the model is simulated considering the same model configuration (i.e., same $k$), similar evolutions of the system are expected, while when the configuration is changed also qualitative variations of the system's dynamics are expected. Therefore,

by analyzing the dynamic of the system in a given model configuration it is possible to estimate the simulation time that is required to reach the steady state in that particular condition.

A strategy so defined works properly if the model follows qualitatively similar dynamics, as in our case, otherwise, in case of variable output modes (e.g., bistability), other approaches of SA should be adopted, like a qualitative SA based on clustering of model dynamics (as described in 4).

Our idea is to run an exploratory simulation considering a "large" simulation time whenever a new model configuration has to be tested, then estimate the simulation time required to evaluate the current steady state, and finally launch a simulation batch with the optimized simulation time.

Therefore in terms of adaptivity, our algorithm makes the use of the following procedure,
whenever the parameter value $k$ changes:

1. launch an exploratory simulation to monitor model dynamics
2. analyze the standard deviation of [RL] values
3. estimate and decide the time required for the attainment of the steady state condition
4. possibly change the simulation time and run the simulations batch

To recognize whether the steady state has been reached or not, the standard deviation of the last values assumed by [RL] is analyzed. The problem is that the intrinsic noise presents in the system may be not constant, therefore it is not an easy task set a proper standard deviation threshold evaluating if the model has reached a noisy steady state rather than it is still in a noisy transient phase. As a consequence, we propose to compute the time required by the model to reach an "intrinsic standard deviation" of the target output (i.e., [RL]), and then to use the associated simulation time $t$ to run the simulation batch.

The value of $k$ is modified by the mean of the computation of the product between its default value (which is 4) and a factor $f = 1, ..., 20$. Acting with an increase of the constant associated with a biochemical transformation, is a common practice to computationally simulate a condition of enhanced activity of the enzyme associated with the respective biochemical reaction. To consider the stochasticity of the model's behavior, the estimation [RL] at steady state ($[RL]_s$) was performed considering the average values over a set of $n = 10$ model simulations for each $k$ value.

The analysis has been performed using both, a standard approach with a fixed simulation time for the whole analysis, and a self-adaptive approach using self-adaptive simulation times according to the actual dynamics followed by the system.

| Input k | Adaptive Simulation time (t.u.) | $[RL]_s$ nMolar | | Computation time ratio |
|---|---|---|---|---|
| 4 | 1200 | [a]1105 | [b]991 | 0.54 |
| 8 | 1300 | [a]2091 | [b]2007 | 0.61 |
| 12 | 1800 | [a]3035 | [b]3009 | 0.74 |
| 16 | 1800 | [a]4047 | [b]4010 | 0.73 |
| 20 | 1700 | [a]5061 | [b]4989 | 0.70 |
| 24 | 1400 | [a]6031 | [b]6006 | 0.65 |
| 28 | 1400 | [a]7058 | [b]7021 | 0.67 |
| 32 | 900 | [a]8055 | [b]7990 | 0.46 |
| 36 | 800 | [a]9115 | [b]8989 | 0.43 |
| 40 | 700 | [a]10000 | [b]10014 | 0.39 |
| 44 | 700 | [a]10912 | [b]11003 | 0.43 |
| 48 | 1700 | [a]11982 | [b]11973 | 0.78 |
| 52 | 1300 | [a]12971 | [b]12989 | 0.75 |
| 56 | 800 | [a]13795 | [b]13991 | 0.51 |
| 60 | 1300 | [a]14944 | [b]14970 | 0.62 |
| 64 | 1400 | [a]15985 | [b]16965 | 0.67 |
| 68 | 1700 | [a]16967 | [b]16965 | 0.76 |
| 72 | 1600 | [a]17973 | [b]17978 | 0.72 |
| 76 | 1600 | [a]18976 | [b]19005 | 0.70 |
| 80 | [a]2100 | [a]20039 | [b]20050 | 0.84 |

Table 5.1: **Adaptive analysis data**

Input $k$ and retrieved $[RL]_s$ for: (a) self-adaptive mechanism; (b) non-adaptive approach. For the non-adaptive experiment the simulation time has been fixed at 3000 t.u.. The ratio represents the computation time of the adaptive analysis over the computation time of the non adaptive analysis.

For the implementation of the self-adaptive mechanism, the exploratory simulation is continuously relaunched with the simulation time gradually increasing of 100 t.u. at a time (starting with a value of 100 t.u.), this until the standard deviation of [RL] values in the last ranges of time does not change by more that 5% by having increased the simulation time (i.e., the standard deviation of [RL] is approximatively stable around a given value). Once the simulation time $t$ has been retrieved, the simulation batch is launched.

The comparison of the results obtained using the standard non-adaptive approach and the results retrieved introducing self-adaptive mechanism shows that as expected the introduction of self-adaptive simulation time leads to a significant decrease of the computation time required to run the model, and very important, the estimated output $[RL]_s$ is essentially the same in both of the cases (Table 5.1). In other words, the self-adaptive mechanism allowed the reduction of the computation time while preserving the accuracy of the analysis.

In fact, the regression performed in the two conditions leads substantially to the same outcome, but it requires less computation time in the case in which the self-adaptive

mechanism is included. More specifically, the regression analysis shows a strong linear relation between $x$ and $[\mathrm{RL}]_s$ (Figure 5.3).

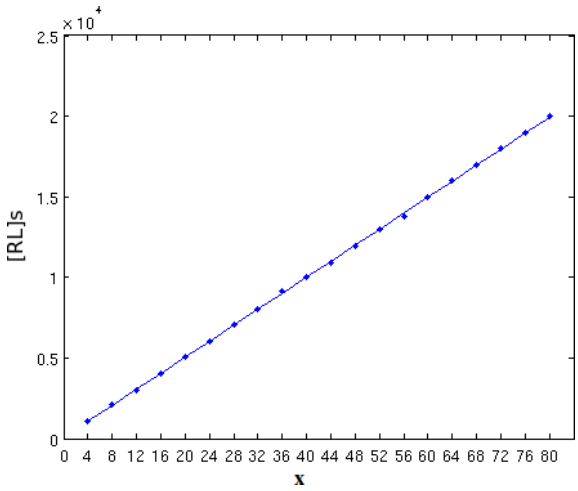


Figure 5.3: **Regression analysis:**
In circles are indicated the interpolation points, while the curve is the regression line. The interpolated curve is in the form: $[RL]_s = \beta_0 + \beta_1(x)$, with $\beta_0 = 244.12$ and $\beta_0 = 77.80$, and with an $R^2 = 0.985$.

It is important to consider that since in SA the modelers act directly on the source of variability (i.e., is the input), it is possible to foresee in which steps of the analysis the modifications in the model behavior may occur. As a consequence, the implementation of the self-adaptive mechanism is facilitated by the fact that it is possible to accurately analyze the model behavior only at specific computational steps of the analysis, that is, when qualitative variations of model dynamics are expected.

In fact, the introduction of the self-adaptive mechanism in itself introduces a boost of the computation time due to the necessity to control the [RL] evolution. However, this effect in our analysis is completely overcome by the benefits deriving by the reduction of the computation times required to perform the simulations batches, when optimized simulation times are used (at least when several simulation need to be performed).

In our case study, the ratio between the additional cost deriving by the introduction of the control mechanism, and the improvements as a consequence of the adaptive behavior, depends on the number of model runs in each simulation batch. More specifically, we verified that when $n = 10$ an important improvement corresponding to a shrink of 40% of the time required to run the simulations is observed. Instead, if only one simulation ($n = 1$) for input factor value would be considered (that would be the situation of deterministic simulation), the self-adaptive system does not lead to any reduction of the computation time. Moreover, the additional cost in terms of computational time deriving from the monitoring of [RL] evolution exceeds the effect deriving from the reduction of the simulation time, therefore yielding to a deterioration of the performance. The relation between the number of model simulation in each batch $n$ and the computation time of the analysis

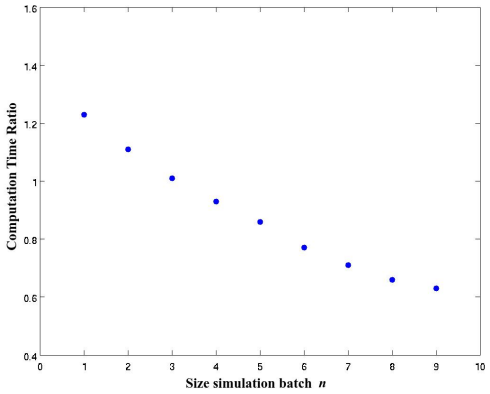


Figure 5.4: **Computation time ratio**

Ratio of the computation time between adaptive simulation time approach and fixed time solution as the number of model simulations per model configuration increases.

shows that the advantage of the exploitation of the adaptive mechanism starts for $n = 4$ and gradually increases as $n$ increases (Figure 5.4). As a consequence, the implementation of adaptive mechanisms should be carefully evaluated according to the specific analytical context.

## 5.2 Grid computing to distribute model simulations

As already discussed, one of the major issue in the analysis of stochastic models is the high computational demand that is usually associated with the stochastic simulation algorithms. The necessity of run several instances of the same model simulation, that is launching several independent processes, is indeed well suited for the application of embarrassingly parallel procedures. In this context, the deployment of distribute computing over grid infrastructure can be a profitable approach to run in parallel model simulations. In practice, grid computing consists on the exploitation of a group of networked computers to accomplish a common goal (Figure 5.5).

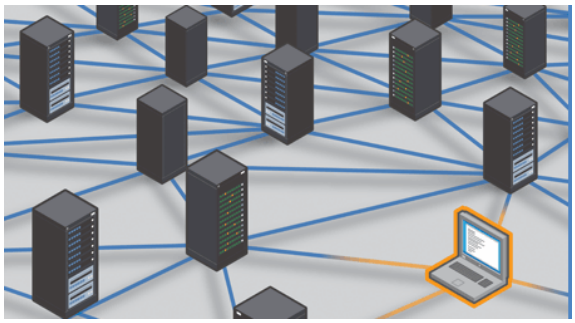

Figure 5.5: **Grid computing**
A grid infrastructure shares many different kinds of resources which are usually transparent to the end user. These computational resources may be used to solve number of problems occurred in science.

In our work, we exploited the European Grid Infrastructure (EGI [113]), a wide area grid platform for scientific applications for launching the model simulations to perform the elementary effects analysis of a stochastic model of bacterial chemotaxis (see section 3.2). EGI is a wide area grid platform for scientific applications composed of thousands of CPUs, which implements the Virtual Organisation (VO) paradigm [114]. At the time of our study (the organization is still in development and expansion), EGI was characterized by:

- 338 partners in 32 countries, organised in 13 Federations
- 239,000CPU-Cores($\approx$20000Biomed)available to users 24 hours a day,7 days a week
- 102 Petabytes (102millionGigabytes) of storage

EGI uses the gLite middleware [115], which is an integrated set of tools designed to permit the sharing of computational resources. gLite middleware must be installed on a local server, defined as User Interface (UI), and allow the management of computations on the EGI. More specifically, by mean of gLite middleware it is possible to submit grid jobs, monitoring the current state and retrieve the output if a computation is successful, or resubmit it in case of failure. The main scope of gLite is to allow the users to cope, in a relatively easy way, with the continuous dynamic reshape of the available resources, which

is typical of loosely coupled distributed platforms. EGI has been developed to be highly scalable and thus allowing in theory, the computation of the intensive challenges present in different field of science, such as bioinformatics problems.

To enable a secure connection to the remote resources, the grid middleware offers a well-established security system based on public key cryptography to recognise users. The access to remote clusters is granted by a personal certificate encoded in the X.509 format, which accompanies each job to authenticate the user. Moreover, users must be authorised to job submission by a virtual organization (VO), in our study we join the Biomed VO [116].

In our study we focused on a common issue presents when exploiting distribute computing over a grid infrastructure, which is the computational demand associated with each grid job. In our specific test case, this means in how many grid jobs split model simulations. In fact, on the one hand, a higher number of jobs leads to a higher parallelization, but on the other hand, it may lead to a higher jobs failure and to clog the batch queues. To this purpose, we launched the same number of model simulations (i.e., $6 \times 10^4$) considering different partitioning of model simulations over grid jobs. Then, to check the performance on the different grid instances we considered two indexes, the crunching factor and the overhead ratio. The crunching factor is defined as the ratio between the total expected CPU time over a single CPU and the overall duration of the grid computation, i.e., the time needed to accomplish the longest job:

$$c = \frac{nt}{\max(\tau_j)}, \quad j = 1, 2, \ldots, n \tag{5.1}$$

where $t$ is the expected time required for the computation of a single job using a single CPU, $\tau_j$ is the grid job time for job $j$ and $n$ is the number of grid jobs. In other words, the crunching factor $c$ represents the average number of CPUs used simultaneously along the whole computation, taking into account the longest job.

The overhead ratio is defined for a job $j$ as the ratio of the difference between the grid job time and the grid CPU time $\tau_j^{\text{cpu}}$ with the grid CPU time:

$$o_j = \frac{\tau_j - \tau_j^{\text{cpu}}}{\tau_j^{\text{cpu}}} \tag{5.2}$$

The quantity $o_j$ is an indicator of the time spent "on the grid" with respect to the actual $\tau_j^{\text{cpu}}$.

More specifically, we distributed the model simulations considering the following four grid runs:

1. 1000 grid jobs with 60 simulations/job
2. 100 grid jobs with 600 simulations/job

| Run | n | Simulations/Job | Success rate [%] | max ($\tau_j$) [min] | c |
|---|---|---|---|---|---|
| 1 | 1000 | 60 | 76 | 1171.0 | 17.1 |
| 2 | 100 | 600 | 84 | 994.6 | 20.1 |
| 3 | 20 | 3000 | 90 | 1742.5 | 11.5 |
| 4 | 10 | 6000 | 100 | 2646.5 | 7.6 |

Table 5.2: **Grid runs performance**

Setting and performances of the four runs of SA distributed over the EGI; $n$ is the number of jobs, success rate is the percentage of the jobs successfully finished at the first submission, and $c$ is the crunching factor.

3. 20 grid jobs with 3000 simulations/job

4. 10 grid jobs with 6000 simulations/job

The performance about the computation of the grid runs are reported in Table 5.2. The best crunching factor, corresponding with the higher reduction in term of computational cost was of 20.1, and it was obtained in run 2, that is neither in the run with more simulation nor in the one with less simulations per grid job. With respect to the run 2, a further increase of the number of jobs (run 1) yielded a lower percentage of jobs successfully completed at the first submission and a higher overhead ratio, while the reduction of the job number (runs 3 and 4) determined better job success rate and better overhead ratio, but, due to the lower parallelism, the overall performance is worse (Figure 5.6).

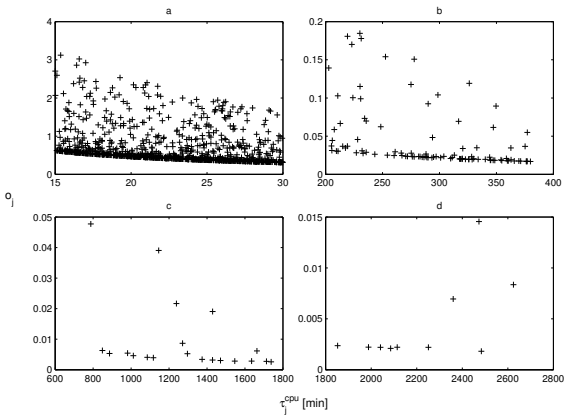


Figure 5.6: **Overhead grid runs**

Scatter plots of grid job times $\tau_j$ (vertical axes) and grid job cpu times (horizontal axes) $\tau_j^{\text{cpu}}$ for (a) run 1, (b) run 2, (c) run 3 and (d) run 4. Figure taken from [117].

To summarize, we verified that the use of grid computing can be a profitable solution to distribute independent model simulations by mean of an embarrassingly parallel paradigm. In fact, the exploitation of grid infrastructure may lead to significant reduction of the time required to retrieve the model outputs. For instance in our specific test case, we were able

to retrieve the output, in the best configuration, in less than 17 h, while using a single CPU about 13 days of computation would be required.
However, in order to get the best performance it is important to properly split the model simulations into grid jobs, in fact, the computation of long jobs on the grid may cause significant data loss in case of system failure or problems related to data transfer, while the execution of a large number of short jobs raises the total latency time in the batch queues, thus affecting the global performance of the system (i.e., higher crunching factor).

## 5.3 Accelerating a stochastic algorithm for reaction diffusion processes

As already discussed (see chapter 3.4), the development of stochastic algorithms for the simulation of diffusion processes is a particular active research field, mainly because of the importance of spatial effects in providing new insights about the regulation of biochemical processes. In fact, the use of these kind of algorithms allow the monitoring of properties whose experimental evaluation would be extremely complex (e.g., impossibility to monitor low time scale pehnomena or requirement of expensive procedures). However, a limiting factor in the exploitation of stochastic algorithms is the high computational cost required to simulate the system dynamics, which in practice, make their application often unfeasible for the the analysis of systems having a realistic complexity.

In order to face the high computational cost issue we simultaneously move in two directions, on the one hand we restated the original algorithm (S$\tau$-DPP), with an MPI version so to allow the exploitation of several CPUs to compute the system dynamics, on the other hand we codified a CUDA version of the algorithm to exploit GPU to further speed up model simulations.

Formally, S$\tau$-DPP considers a system composed by $n$ membranes, which is defined by the following tuple:

$$\Pi = (\Sigma, G_\mu, G_c, C, W, R, V_\mu, V_\Sigma) \tag{5.3}$$

- $\Sigma = \{s_1, \ldots, s_m\}$ is a finite set of symbols representing the molecular species
- $G_\mu$ and $G_c$ are graphs representing the topology of membranes and the channels of communication among them
- $R = \{R_1, \ldots, R_n\}$, where $R_i$ is the set of internal and communication rules occurring inside the $i^{\text{th}}$ membrane.
  An *internal* rule is of the form

$$\alpha_1 s_1 + \cdots + \alpha_m s_m \xrightarrow{c} \beta_1 s_1 + \ldots + \beta_m s_m \tag{5.4}$$

  a *communication* rule is of the form

$$\begin{aligned} \alpha_1 s_1 + \ldots + \alpha_m s_m \xrightarrow{c} (\beta_{1,1} s_1 + \ldots + \beta_{m,1} s_m, \text{in}_1) \\ + \ldots + (\beta_{1,n} s_1 + \ldots + \beta_{m,n} s_m, \text{in}_n) \end{aligned} \tag{5.5}$$

  where the quantities $\alpha_i$ and $\beta_j$ are natural numbers, $c$ is a stochastic constant and $in_1, \ldots, in_n$ indicate the target membrane to which the resulting quantities of molecular species are sent.

- $W = \{w_1, \ldots, w_n\}$, where $w_i = \alpha_{1,i} \ldots \alpha_{m,i}$ is the multisets containing, for the $i^{\text{th}}$ membrane, the amounts of each molecular specie;
- $\mathcal{C} = \{C_1, \ldots, C_n\}$, where $C_i$ is the set of stochastic constants $c_{i,j} \in \mathbb{R}^+$ associated with the rules ($\xrightarrow{c}$) occurring inside the $i^{\text{th}}$ membrane ;
- $V_\mu = \{v_1, \ldots, v_n\}$, where $v_i \in \mathbb{R}^+$ is the volume of the $i^{\text{th}}$ membrane;
- $V_\Sigma = \{v_{s_1}, \ldots, v_{s_m}\}$, where $v_{s_j} \in \mathbb{R}^+$, is the volume of the molecular specie $s_j$.

The system evolves by applying the rules in a set of states $W(t_a) \ldots W(t_z)$. Each rule represents a specific chemical reaction (according the standard convention, we have reactants on the left and products on the right).

A rule is applicable if the probability of a reaction to occur in the next interval of time is sufficiently high, that is reactants are present in sufficient amount. To solve a system so defined it is possible to compute for reach rule, the probability function $p(\tau, j|x, t)d\tau$, representing the likelihood that the next reaction in the system will occur in the infinitesimal time $[t + \tau, t + \tau + d\tau)$ and will be a reaction $r_j$ (as proposed by Gillespie in the original SSA algorithm [9]).

S$\tau$-DPP system exploits a $\tau$ leaping method, which instead of firing one reaction at a time, it may fires more than one reaction events after a pre-selected time step $\tau$ [118]. The concurrently evaluation of more reactions leads to a speed up the computation of the system evolution, however, to guarantee an accurate estimation of the system dynamic it is required that in each step no propensity function has a macroscopically significant change in its value [119]. In case in which more than one rule is applicable, the rule to fire is non-deterministically chosen among them. As a consequence, there may exist condition in which a reaction has at least one reactant presents in low copy number, thus if the reaction occurs it may prevent the firing of others reaction in which the reactant may be involved (these reactions are usually called "critical rules"). The presence of critical rules together with the fact that $\tau$ is computed using random values lead the system to behave stochastically (i.e., run the same simulation different times may result in different outcomes).

S$\tau$-DPP also considers the free space $F_i$ for the membrane $i - th$ at time $t$ as follows:

$$F_i(t) = v_i - \left( \sum_{j=1}^{m} (w_i(s_j, t) \cdot v_{s_j}) + \sum_{l=1}^{n} \alpha_l \cdot v_l \right) \tag{5.6}$$

where $\alpha_l \in \{0, 1\}$ takes the value 1 if the membrane labelled $l$ is a son of membrane $i$ in the membrane hierarchy, otherwise is 0, $w_i(s_j)$ denotes the number of occurrences of the symbol $s_j$ in the multiset $w_i$, and $v_x$ the volume of membranes and molecules.

This add a further constraint in the evolution of the system: a rule is applicable if $F_i \geq 0$ for $1 \leq i \leq n$ calculated in the configuration potentially reached after its application considering both the internal and communication rules.

Therefore, the resulting algorithm, as described in Mosca et al. in [120], is the following:

1. load the description of the S$\tau$-DPP system;
2. for each membrane $i \in [1..n]$ calculate $F_i(t_0)$;
3. for each membrane $i \in [1..n]$
   (a) $\forall$ rule $r_k$, $(k \in \{1, \ldots, l\})$: compute $a_k$;
   (b) evaluate the sum of all the propensity functions $a_0$ in the compartment;
   (c) IF $a_0 \neq 0$: $\tilde{\tau}_i = \infty$;
   ELSE generate the step size $\tilde{\tau}_i$ according to the internal state, and select the way to proceed in the current iteration (i.e. $\tau$ leaping evolution with or without critical reactions);
4. select $\tau_{min} = \tau_i = \min\{\tilde{\tau}_i, \ldots, \tilde{\tau}_n\}$;
5. for each membrane $i \in [1..n]$
   (a) IF $\tilde{\tau}_i = \infty$ goto e) ;
   (b) `switch` the evolution strategy type:
      - `case` "$\tau$ leaping with one critical reaction":
        `if(`$\tilde{\tau}_i > \tau_i$`)`: `goto` d) `else`: `goto` c);
      - else `goto` d);
   (c) extract the critical and the non-critical rules that will be applied in the current iteration;
   (d) IF the execution of the selected rules in all the volumes leads to an unfeasible state $\tau_{min} = \frac{\tau_i}{2}$;
6. IF a new value of $\tau_{min}$ was computed: $\tau_i = \tau_{min}$ and goto 4;
7. for each membrane $i \in [1..n]$
   (a) update the internal state by applying the internal rules
   (b) update the state of other membranes by applying the communication rules;
8. for each membrane $i \in [1..n]$ update the value of the free space $F_i$;

9. IF the termination criteria is satisfied, namely (i) the current time exceeds the end time OR (ii) there is not enough free space in any membrane: finish;
   ELSE: `goto` *3*.

In the following, in section 5.3.1 we described the MPI implementation of S$\tau$-DPP together with its performance on a specific case study, while in section 5.3.2 we described the results obtained using GPU to simulate the same model, exploiting a CUDA version of S$\tau$-DPP.

### 5.3.1 MPI to run simulations in parallel

The S$\tau$-DPP algorithm presents three possible levels of parallelization. The most straightforward is the simulation of many independent instances executed in an embarrassingly parallel way. As we already seen, the embarrassingly parallel is usually exploited in distributed platforms, such as grid computing [117]. A second parallelization approach is related the execution of each single simulation, which can exploit several parallel processes by assigning each of them with one or more membranes. In this case, it is necessary to distinguish between the independent algorithmic operations, that can be executed in parallel and the non-independent algorithmic operations, which instead, require communication processes. In S$\tau$-DPP the steps 2, 3, 5, 7$a$ and 8 can be executed in parallel since they consider the dynamic evolution within each membrane, while the steps 4, 6, 7$b$ and 9 requires communications among the different parallel processes since they are related to the transfer of objects between compartment and the synchronization of the overall system. A third potential parallelization paradigm would be the evaluation of the propensity functions in parallel (step 3$a$) which could be performed having one thread for rule and thus it is related to the number of rules per membrane.

Considering that the number of rules is usually not particularly high (order of tens at most), we focused on the second type of parallel implementation. The goal was to allow simulating systems of arbitrary size, in other words, the algorithm should be able to properly scale according to the available computational resources as the number of membranes change. To this purpose we exploited a traditional clusters to test the performance derived by the possible different usage of the resources. As case study we considered the gene network model described in section 3.4. In the model, the diffusion processes of some regulatory factors of gene expression is simulated, the diffusion happens in a nucleus space in which are located two target genes. We considered different levels of granularities in which partition the nucleus space, more specifically, we described the nucleus space considering square lattices of size: $6 \times 6$, $9 \times 9$, $12 \times 12$, $30 \times 30$, and $90 \times 90$. The more is number of membranes considered the more is the detail in which the space is represented (Figure 5.7).

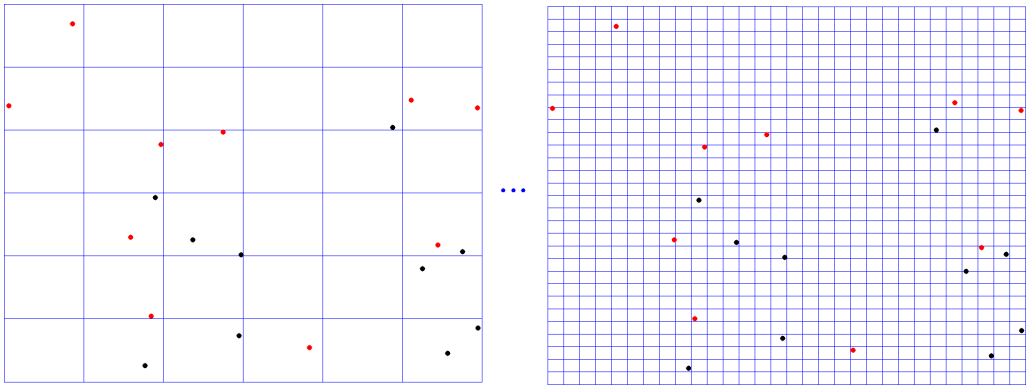

Figure 5.7: **Nucleus space granularity**

The same space representing the room within the nucleus have been partitioned considering different dense lattice, on the left a case with a square lattice of size 6, on the right with size 30. Points represent the localization of regulatory factors (different colors for different elements).

| compartment | Sequential | Num. of cores | | | | | |
|---|---|---|---|---|---|---|---|
| | | 2 | 4 | 8 | 16 | 32 | 64 |
| 6x6 | 0.5 msec. | 1.75 | 2.06 | 2.00 | 1.88 | 1.70 | 0.61 |
| 9x9 | 1.1 msec. | 1.76 | 2.60 | 2.80 | 2.26 | 2.24 | 2.23 |
| 12x12 | 2.1 msec. | 1.86 | 2.80 | 3.50 | 3.20 | 2.08 | 2.12 |
| 30x30 | 14.6 msec. | 1.81 | 3.90 | 5.30 | 6.47 | 7.03 | 7.54 |
| 90x90 | 138 msec. | 1.87 | 3.70 | 6.10 | 10.40 | 15.70 | 19.80 |

Table 5.3: **MPI performance**

Execution times, in milliseconds for the sequential version and speedup values using up to four cluster nodes.

We run the model simulations considering different number of compartments, on a cluster composed by 18 nodes equipped with two 4-cores Intel(R) Xeon(R) CPU E5345 @ 2.33GHz, 16 GB of Ram, linked together via an Infiniband QDR network. The actual simulation time of a model simulation depends by the number of steps required to simulate the system evolution which in turns depends by both the initial configuration and the chain of random numbers to compute the propensity functions. As a consequence, we considered the averaged time for each iteration of the model simulation for the comparison of the computational performance of S$\tau$-DPP (Table 5.3).

We observed that compartments number to represent the nucleus space significantly affects the scalability of the algorithm. With less than 100 regions (9x9) the use of the 4 cores of a single CPU is the most reasonable parallelism degree, instead, the usage of two nodes is suitable only with at least 900 regions (30x30), while the maximum parallelism degree is achieved for a simulation with 8,100 regions (90x90), that result in a speedup of about 20 using 4 nodes (Figure 5.8).

The scalability is not particularly good, probability due to the fact that increasing the

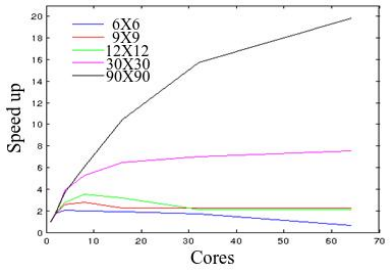


Figure 5.8: **MPI scalability**

Scalability graph of MPI implementation of S$\tau$-DPP.

number of processes and exploiting more than one node has the consequence that the communication times overcome the single computation times. However, exploit the cluster allowed us to execute a single 90x90 simulation in about 18 minutes, while the use of the sequential version of the algorithm required 6 hours.

As a consequence, our implementation permits to reduce the computation time for the simulation of S$\tau$-DPP by using a cluster infrastructure, therefore, allowing the increase the detail in which a diffusion process can be described.

### 5.3.2 Exploit graphics processing unit

In order to overcome the communication overhead of the MPI implementation of S$\tau$-DPP, we developed a CUDA-based implementation. In fact, exploiting the shared memory architecture of graphics processor may improve the algorithm scalability. Despite some works are present in literature about the use of Graphic Processing Unit (GPU) for stochastic simulation of chemical diffusion processes, they neglect the molecule volumes and the free space [121]. Therefore, S$\tau$-DPP CUDA-based simulator is a novelty in the field of stochastic simulations since it is the first implementation (at our knowledge), of a stochastic algorithm, for GPU computing, considering the effect of molecular crowding in reaction diffusion processes.

The reason to use GPU for computation rather than CPU is due to the fact, that GPU is designed has a highly parallel structure, allowing large blocks of data to be processed at one time, this can be an advantage in all the case in which massive parallel operations are required. In fact, while a CPU is composed of a only few cores with lots of cache memory which can handle a few software threads at a time, a GPU is composed of hundreds of cores that can handle thousands of threads simultaneously (Figure 5.9 ).

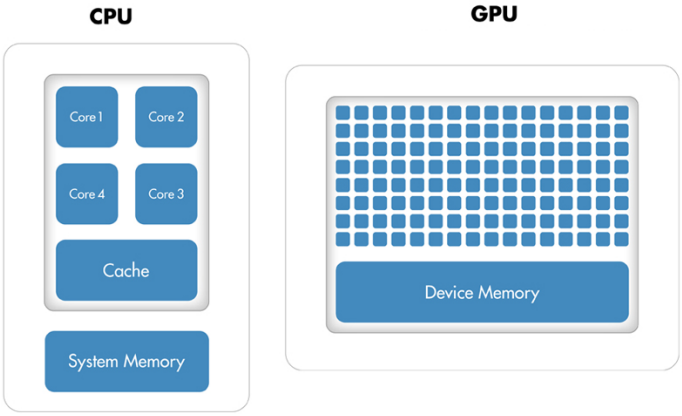


Figure 5.9: **GPU vs CPU architecture**
A GPU (on the right) comprises significant more processing units than a CPU (on the left).

Given the intrinsic nature of S$\tau$-DPP algorithm, which is based on the concurrent computation of the dynamic in different compartments, the exploitation of GPU is in effect a good solution to speed up model simulations.

We developed a CUDA kernel in which the thread blocks carry out independent simulations of the model, given an initial condition of the system and a seed for the generation of the chain of random numbers used by the algorithm throughout the computation of the system dynamics. Inside each block, the workload is balanced among the threads assigning to each of them the computation of the evolution of the most possible equal number of compartments. In this way, for any given number of membranes in the system, it is possible to choose the number of threads per block which best fits the GPU architecture.

Since usually a model has to be simulated several times, it is possible to launch an arbitrary number of blocks according to the number of simulations to be performed. Within each thread block, at each algorithm iteration, which correspond to a time frame, three communication/synchronization processes are required:

- determine the smallest time increment
- check the correctness of the overall system evolution with respect to the resulting free space in the compartments
- transfer objects between the membranes

A fundamental problem in developing parallel applications on GPU is the memory management. We verified that keeping in the the device shared memory those variables which must be visible to all threads, such as the time increment $\tau$ or the free and in the device constant memory is not a good solution, if we want to have a simulator dealing with an arbitrary number of membranes. In fact, to keep the record of every membrane location imposes too strict limits to the size of the systems which is possible to simulate. Therefore, we opted to place all the variables required for the simulation of the system dynamic in the device global memory.

We find out that the best solution for the memory management is the allocation through a unique host-side CUDA API call such as *cudaMalloc* all the one-dimensional arrays used by all threads in all blocks (and inside the computational kernel we declare pointers to the memory regions reserved to the thread blocks to reduce address arithmetics). The arrangement of the variables in memory should be done in order to allow the different threads to access memory locations in a unitary stride. For instance, given a system with h $M$ membranes, $S$ species per membrane, and $T$ threads per block, a good solution is to partition the global one-dimensional array allocated (which will be of size $M \times S$) into $S$ subarrays of $M$ elements each, and to further divide the subarrays into groups of $T$ adjacent elements. Then, with an external loop over the $S$ species, and an internal loop over $M$ membranes it is possible to access the variables in a fully coalesced way. Therefore, we avoid any communication between host and device during the simulation, thus enabling an efficient computation.

As for MPI implementation, we tested CUDA implementation of S$\tau$-DPP on the model of diffusion processes happening in the nucleus space (for detail see section 3.4). In this case, we considered three different square lattices to describe the nucleus space (i.e., $16^2$, $32^2$, $64^2$). Regarding the hardware specification, the simulation have been launched on an NVIDIA GeForce GTX 580 device.

The results show that S$\tau$-DPP CUDA implementation scales well when augmenting the number of compartments and for a fixed number of simulation instances, increasing

| Nmem | gridDIM | Time per step | Speed up |
|---|---|---|---|
| 256 | 1 | 6,52 | 0.3 |
| 256 | 32 | 0.36 | 5.8 |
| 256 | 64 | 0.17 | 12.4 |
| 256 | 128 | 0.09 | 22.6 |
| 256 | 256 | 0.09 | 24.5 |
| 1024 | 1 | 22.10 | 0.3 |
| 1024 | 32 | 1.38 | 5.5 |
| 1024 | 64 | 0.64 | 12.0 |
| 1024 | 128 | 0.38 | 20.4 |
| 4096 | 1 | 80.40 | 0.4 |
| 4096 | 32 | 4.88 | 6.1 |
| 4096 | 64 | 2.43 | 12.3 |

Table 5.4: **CUDA performance**
Execution times in milliseconds of the sequential and CUDA implementations of the STAUCC simulator with related speedup values. Nmem is the number of compartments while gridim is the number of blocks.

the system dimension does not affect the speed up achievable (Table 5.4). To conclude, our CUDA implementation provides good performance in term scalability and the more is the number of processing unit available the more it is possible to simulate a larger number of compartments. Therefore, considering the constant advance in the market of GPU we expect that our algorithm could be a good solution for the simulation of stochastic reaction diffusion processes.

# Chapter 6

# Conclusion

Hereinafter, we recall the main results that have been obtained throughout the PhD work, followed by the open issues that we deem present the major challenges in term of scientific research. In section 6.1 we summarize how according to our studies, the application of SA methods to study biochemical systems has an important role for testing the reliability of a model during its development, and for understanding its behavior when used for predictive purposes. Aftermath, in section 6.2 we illustrate the most critical issues that we believe deserve a deeper attention in order to increase the applicability of SA techniques for the analysis of the behaviors of computation models of biochemical systems.

## 6.1 Summary

The thesis focused on different aspects related to the development and application of SA on computational models for the simulation of biochemical systems. Throughout the thesis we addressed different issues involved in the exploitation of SA methods for the study of model behavior, ranging from theoretical aspects, such as the exploration of the input factor space, to strictly computational questions, like the reduction of computational time required by model simulations.

In effect, our analyses show that there exist a variety of reasons for which the application of SA for the study of biochemical systems models can greatly support the definition of reliable models and improve the comprehension of model behavior. For instance, SA can suggest new experiments aimed at reducing the uncertainties associated with critical parameters. In fact, given the complexity of biochemical systems many parameters may be needed for a proper representation bu mean of mechanist models, and the setting of these parameters may require time/money consuming experiments. As a consequence, according to some assumptions, their values is usually only roughly estimated. However, by using SA it is often possible to verify that the behavior of a model is significantly affected only by a minority of the input factors while the majority of them have small

or even negligible effects. Therefore, by computing sensitivity indexes we can identify the most influencing input factors in determining the model behavior, and subsequently perform targeted measurements aimed to precisely set their values or at least reduce their uncertainties.

Moreover, with SA it is possible to test the robustness of a model. In fact, usually biochemical systems are characterized by high level of robustness, which reflects the ability of such systems to respond to environmental modifications maintaining the proper functioning. Robustness is in effect particularly relevant for living systems which have naturally deal with high unstable conditions. However, models are often developed to represent particular set of data, and despite they may have a good fitting with the experimental observations, it is not so uncommon, that as soon as some parameters are changed the model behaves in an unrealistic way (e.g., change of dynamics modes). With SA we can indeed test if a model preserves a realistic behavior when parameter values are changed, thus testing the reliability of the model during its development.

The application of SA can be also extremely helpful also after that a model has been already validated, in this case the aim is to get new insights about the regulation of the system behavior, or foresee potential effects when it is know in advance that a given perturbation will affect the system, or even to develop strategies in order to affect the model behavior in a specific way.

In fact, since biochemical networks can be characterized by the presence of many feedbacks, it may not be easy to predict the effect of system perturbations, which could also be non-linear. In this context, the application of global SA can allows the discovering of "hidden" control points determining the system state.

## 6.2 Open issues

In our research activity on SA, we have encountered different issues, and regarding some of them we consider that more extensive studies can be planned in order to increase the applicability of SA, and enhance its capability to study model behavior.
For instance, it could be significant the application of qualitative SA not only to discriminate between oscillatory and steady state conditions, but also in general, to distinguish among whatever condition that can be associated with different discrete dynamic states. The idea is that the clustering of a given output property in conjunction with boundary detection algorithm may in theory enables the analysis of how the transitions between different dynamic trends is regulated.
Moreover, it would be more relevant the application only to noisy conditions derived from deterministic simulations, as we perform, but also to pure stochastic simulations.

Considering stochastic simulations makes more difficult the boundary detection since it

produces probabilistic contours (i.e., in a given point different probabilities to reach every potential state have to be defined). However, a profitable advantage would is that by using clustering analysis it could be possible to identify bistable points (i.e., identification of clear distinguishable dynamic trends) that would be otherwise difficult to detect.

Another topic that we believe deserve greater attention is the analysis of spatial effects in diffusion processes. In fact, spatiality, mainly due to the shortage of algorithms for efficient simulations, has been often disregarded in the modeling of biochemical processes, as a consequence, there is also a gap that in the application of SA that could be filled. In effect, we already verified that by keep tracking spatial localization of biochemical entities enables the consideration of system states probability state over time, however, none specific computation of sensitivity indexes have been proposed. It would be useful the definition of a procedure for the sampling of different spatial configurations aimed at the computation of indexes representing the influence of the spatial location over the model behavior.

Finally, another issue that hamper the application of SA to stochastic models is the high computational demand that is associated with stochastic simulation algorithms. We already proved, that our new implementations of an algorithm for the stochastic simulation of reaction diffusion processes may take advantage of parallel infrastructures (i.e., CPU clusters, GPU). However, the speed up depends from model under analysis, for this reason, it would be important to consider the integration of different approaches in order to reach the best performance as possible according to the specific case study and to the available hardware.

# List of Figures

# List of Tables

# Bibliography


[1] Gregory Stephanopoulos, Hal Alper, and Joel Moxley. Exploiting biological complexity for strain improvement through systems biology. *Nature biotechnology*, 22(10):1261–1267, 2004.

[2] Leroy Hood. Systems biology: integrating technology, biology, and computation. *Mechanisms of ageing and development*, 124(1):9–16, 2003.

[3] Zoltan Szallasi, Jorg Stelling, and V Periwal. *System modelling in cellular biology*. MIT Press, Cambridge, MA, 2006.

[4] Onn Brandman and Tobias Meyer. Feedback loops shape cellular signals in space and time. *Science*, 322(5900):390–395, 2008.

[5] K Mark Ansel, Vu N Ngo, Paul L Hyman, Sanjiv A Luther, Reinhold Förster, Jonathon D Sedgwick, Jeffrey L Browning, Martin Lipp, and Jason G Cyster. A chemokine-driven positive feedback loop organizes lymphoid follicles. *Nature*, 406(6793):309–314, 2000.

[6] Hidde De Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of computational biology*, 9(1):67–103, 2002.

[7] Tan Chee Meng, Sandeep Somani, and Pawan Dhar. Modeling and simulation of biological systems with stochasticity. *In silico biology*, 4(3):293–309, 2004.

[8] Yang Cao and David C Samuels. Discrete stochastic simulation methods for chemically reacting systems. *Methods in enzymology*, 454:115–140, 2009.

[9] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

[10] Yucheng Hu and Tiejun Li. Highly accurate tau-leaping methods with random corrections. *The Journal of chemical physics*, 130(12):124109–124109, 2009.

[11] Daniel T Gillespie. Stochastic simulation of chemical kinetics. *Annu. Rev. Phys. Chem.*, 58:35–55, 2007.

[12] Johan Elf and Måns Ehrenberg. Fast evaluation of fluctuations in biochemical networks with the linear noise approximation. *Genome research*, 13(11):2475–2484, 2003.

[13] Howard Salis and Yiannis Kaznessis. Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *The Journal of chemical physics*, 122:054103, 2005.

[14] Michael Hucka, Andrew Finney, Herbert M Sauro, Hamid Bolouri, John C Doyle, Hiroaki Kitano, Adam P Arkin, Benjamin J Bornstein, Dennis Bray, Athel Cornish-Bowden, et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[15] Michael Hucka et al. The systems biology markup language (sbml): Language specification for level 3 version 1 core. 2009.

[16] Giovanni De Matteis, Alex Graudenzi, and Marco Antoniotti. A review of spatial computational models for multi-cellular systems, with regard to intestinal crypts and colorectal cancer development. *Journal of mathematical biology*, pages 1–54, 2012.

[17] Jan Paul Medema and Louis Vermeulen. Microenvironmental regulation of stem cells in intestinal homeostasis and cancer. *Nature*, 474(7351):318–326, 2011.

[18] Daniele Ramazzotti, Carlo Maj, and Marco Antoniotti. A model of colonic crypts using sbml spatial. *arXiv preprint arXiv:1309.7692*, 2013.

[19] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. Wiley. com, 2008.

[20] H Christopher Frey and Sumeet R Patil. Identification and review of sensitivity analysis methods. *Risk analysis*, 22(3):553–578, 2002.

[21] Andrea Saltelli, Stefano Tarantola, Francesca Campolongo, and Marco Ratto. *Sensitivity analysis in practice: a guide to assessing scientific models*. John Wiley & Sons, 2004.

[22] Maria Rodriguez-Fernandez, Julio R Banga, and Francis J Doyle. Novel global sensitivity analysis methodology accounting for the crucial role of the distribution of input parameters: application to systems biology models. *International Journal of Robust and Nonlinear Control*, 22(10):1082–1102, 2012.

[23] JC Helton and FJ Davis. Illustration of sampling-based methods for uncertainty and sensitivity analysis. *Risk Analysis*, 22(3):591–622, 2002.

[24] Zoran Bosnić and Igor Kononenko. Estimation of individual prediction reliability using the local sensitivity analysis. *Applied intelligence*, 29(3):187–203, 2008.

[25] Toshimitsu Homma and Andrea Saltelli. Importance measures in global sensitivity analysis of nonlinear models. *Reliability Engineering & System Safety*, 52(1):1–17, 1996.

[26] Jon C Helton, Jay Dean Johnson, Cedric J Sallaberry, and Curt B Storlie. Survey of sampling-based methods for uncertainty and sensitivity analysis. *Reliability Engineering & System Safety*, 91(10):1175–1209, 2006.

[27] S Kucherenko, María Rodriguez-Fernandez, C Pantelides, and N Shah. Monte carlo evaluation of derivative-based global sensitivity measures. *Reliability Engineering & System Safety*, 94(7):1135–1148, 2009.

[28] Hong-Xuan Zhang and John Goutsias. A comparison of approximation techniques for variance-based sensitivity analysis of biochemical reaction systems. *BMC bioinformatics*, 11(1):246, 2010.

[29] G Gary Wang and S Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129:370, 2007.

[30] DM Hamby. A review of techniques for parameter sensitivity analysis of environmental models. *Environmental Monitoring and Assessment*, 32(2):135–154, 1994.

[31] Max D Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991.

[32] Ronald L Iman and Jon C Helton. Comparison of uncertainty and sensitivity analysis techniques for computer models. Technical report, Sandia National Labs., Albuquerque, NM (USA), 1985.

[33] Yongtai Huang and Lei Liu. Sensitivity analysis for the identification of important parameters in a water quality model. In *Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008. The 2nd International Conference on*, pages 3131–3134. IEEE, 2008.

[34] Andrea Saltelli, Marco Ratto, Stefano Tarantola, and Francesca Campolongo. Sensitivity analysis for chemical models. *Chemical reviews*, 105(7):2811–2828, 2005.

[35] 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.

[36] Francesca Campolongo, Andrea Saltelli, and Jessica Cariboni. From screening to quantitative sensitivity analysis. a unified approach. *Computer Physics Communications*, 182(4):978–988, 2011.

[37] J Cariboni, D Gatelli, R Liska, and A Saltelli. The role of sensitivity analysis in ecological modelling. *Ecological modelling*, 203(1):167–182, 2007.

[38] A Kiparissides, SS Kucherenko, A Mantalaris, and EN Pistikopoulos. Global sensitivity analysis challenges in biological systems modeling. *Industrial & Engineering Chemistry Research*, 48(15):7168–7180, 2009.

[39] Andrea Saltelli and Paola Annoni. How to avoid a perfunctory sensitivity analysis. *Environmental Modelling & Software*, 25(12):1508–1517, 2010.

[40] Andrea Saltelli, Stefano Tarantola, and Francesca Campolongo. Sensitivity analysis as an ingredient of modeling. *Statistical Science*, pages 377–395, 2000.

[41] Andrea Saltelli. Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, 145(2):280–297, 2002.

[42] Karen Chan, Andrea Saltelli, and Stefano Tarantola. Winding stairs: a sampling tool to compute sensitivity indices. *Statistics and Computing*, 10(3):187–196, 2000.

[43] Andrea Saltelli, Paola Annoni, Ivano Azzini, Francesca Campolongo, Marco Ratto, and Stefano Tarantola. Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index. *Computer Physics Communications*, 181(2):259–270, 2010.

[44] Paul Bratley, Bennett L Fox, and Harald Niederreiter. Implementation and tests of low-discrepancy sequences. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 2(3):195–213, 1992.

[45] Michiel JW Jansen. Analysis of variance designs for model output. *Computer Physics Communications*, 117(1):35–43, 1999.

[46] Dirk Gorissen, Ivo Couckuyt, Piet Demeester, Tom Dhaene, and Karel Crombecq. A surrogate modeling and adaptive sampling toolbox for computer based design. *The Journal of Machine Learning Research*, 99:2051–2055, 2010.

[47] Bruno Sudret. Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering & System Safety*, 93(7):964–979, 2008.

[48] Roberto Barrio. Sensitivity analysis of odes/daes using the taylor series method. *SIAM Journal on Scientific Computing*, 27(6):1929–1947, 2006.

[49] Zhiguang Qian, Carolyn Conner Seepersad, V Roshan Joseph, Janet K Allen, and CF Jeff Wu. Building surrogate models based on detailed and approximate simulations. *Journal of Mechanical Design*, 128:668, 2006.

[50] Mustafa Hekimoğlu and Yaman Barlas. Sensitivity analysis of system dynamics models by behavior pattern measures. 2010.

[51] Alexandre Donzé, Eric Fanchon, Lucie Martine Gattepaille, Oded Maler, and Philippe Tracqui. Robustness analysis and behavior discrimination in enzymatic reaction networks. *PLoS One*, 6(9):e24246, 2011.

[52] Alexandre Donzé, Gilles Clermont, and Christopher J Langmead. Parameter synthesis in nonlinear dynamical systems: Application to systems biology. *Journal of Computational Biology*, 17(3):325–336, 2010.

[53] Brian K McNab. On the utility of uniformity in the definition of basal rate of metabolism. *Physiological Zoology*, 70(6):718–720, 1997.

[54] Ralph J DeBerardinis, Julian J Lum, Georgia Hatzivassiliou, and Craig B Thompson. The biology of cancer: metabolic reprogramming fuels cell growth and proliferation. *Cell metabolism*, 7(1):11–20, 2008.

[55] Ettore Mosca, Matteo Barcella, Roberta Alfieri, Annamaria Bevilacqua, Gianfranco Canti, and Luciano Milanesi. Systems biology of the metabolic network regulated by the akt pathway. *Biotechnology advances*, 30(1):131–141, 2012.

[56] Jörg Stelling, Steffen Klamt, Katja Bettenbrock, Stefan Schuster, and Ernst Dieter Gilles. Metabolic network structure determines key aspects of functionality and regulation. *Nature*, 420(6912):190–193, 2002.

[57] Natalie C Duarte, Scott A Becker, Neema Jamshidi, Ines Thiele, Monica L Mo, Thuy D Vo, Rohith Srivas, and Bernhard Ø Palsson. Global reconstruction of the human metabolic network based on genomic and bibliomic data. *Proceedings of the National Academy of Sciences*, 104(6):1777–1782, 2007.

[58] Igor Goryanin, T Charles Hodgman, and Evgeni Selkov. Mathematical simulation and analysis of cellular metabolism and regulation. *Bioinformatics*, 15(9):749–758, 1999.

[59] Wolfram Liebermeister, Jannis Uhlendorf, and Edda Klipp. Modular rate laws for enzymatic reactions: thermodynamics, elasticities and implementation. *Bioinformatics*, 26(12):1528–1534, 2010.

[60] J Rankin SMALL and David A FELL. Metabolic control analysis. *European Journal of Biochemistry*, 191(2):413–420, 1990.

[61] Carlo Maj, Ettore Mosca, Ivan Merelli, Giancarlo Mauri, and Luciano Milanesi. Sensitivity analysis for studying the relation between biochemical reactions and metabolic phenotypes. *Journal of bioinformatics and computational biology*, 11(01), 2013.

[62] Robert C Spear, Thomas M Grieb, and Nong Shang. Parameter uncertainty and interaction in complex environmental models. *Water Resources Research*, 30(11):3159–3169, 1994.

[63] André I Khuri and Siuli Mukhopadhyay. Response surface methodology. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(2):128–149, 2010.

[64] Gatut Sudarjanto, Beatrice Keller-Lehmann, and Jurg Keller. Optimization of integrated chemical–biological degradation of a reactive azo dye using response surface methodology. *Journal of hazardous materials*, 138(1):160–168, 2006.

[65] Jason N Bazil, Gregery T Buzzard, and Ann E Rundell. Modeling mitochondrial bioenergetics with integrated volume dynamics. *PLoS computational biology*, 6(1):e1000632, 2010.

[66] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[67] Daniela Besozzi, Paolo Cazzaniga, Matteo Dugo, Dario Pescini, and Giancarlo Mauri. A study on the combined interplay between stochastic fluctuations and the number of flagella in bacterial chemotaxis. *arXiv preprint arXiv:0910.1415*, 2009.

[68] George H Wadhams and Judith P Armitage. Making sense of it all: bacterial chemotaxis. *Nature Reviews Molecular Cell Biology*, 5(12):1024–1037, 2004.

[69] Marcus J Tindall, Philip K Maini, Steven L Porter, and Judith P Armitage. Overview of mathematical approaches used to model bacterial chemotaxis ii: bacterial populations. *Bulletin of mathematical biology*, 70(6):1570–1607, 2008.

[70] Orkun S Soyer and Richard A Goldstein. Evolution of response dynamics underlying bacterial chemotaxis. *BMC evolutionary biology*, 11(1):240, 2011.

[71] Nazli Maki, Jason E Gestwicki, Ellen M Lake, Laura L Kiessling, and Julius Adler. Motility and chemotaxis of filamentous cells ofescherichia coli. *Journal of bacteriology*, 182(15):4337–4342, 2000.

[72] Ettore Mosca, Roberta Alfieri, Carlo Maj, Annamaria Bevilacqua, Gianfranco Canti, and Luciano Milanesi. Computational modeling of the metabolic states regulated by the kinase akt. *Frontiers in physiology*, 3, 2012.

[73] Paolo Cazzaniga, Dario Pescini, Daniela Besozzi, and Giancarlo Mauri. Tau leaping stochastic simulation method in p systems. In *Membrane Computing*, pages 298–313. Springer, 2006.

[74] Andrea Degasperi and Stephen Gilmore. Sensitivity analysis of stochastic models of bistable biochemical reactions. In *Formal Methods for Computational Systems Biology*, pages 1–20. Springer, 2008.

[75] Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[76] William L Oberkampf, Jon C Helton, Cliff A Joslyn, Steven F Wojtkiewicz, and Scott Ferson. Challenge problems: uncertainty in system response given uncertain parameters. *Reliability Engineering & System Safety*, 85(1):11–19, 2004.

[77] Jiyong Liang and Joyce M Slingerland. Multiple roles of the pi3k/pkb (akt) pathway in cell cycle progression. *Cell cycle*, 2(4):336–342, 2003.

[78] Heather R Christofk, Matthew G Vander Heiden, Marian H Harris, Arvind Ramanathan, Robert E Gerszten, Ru Wei, Mark D Fleming, Stuart L Schreiber, and Lewis C Cantley. The m2 splice isoform of pyruvate kinase is important for cancer metabolism and tumour growth. *Nature*, 452(7184):230–233, 2008.

[79] Ilya M Sobol and Sergei Kucherenko. Derivative based global sensitivity measures and their link with global sensitivity indices. *Mathematics and Computers in Simulation*, 79(10):3009–3017, 2009.

[80] Francisco C Alcaraz, Michel Droz, Malte Henkel, and Vladimir Rittenberg. Reaction-diffusion processes, critical dynamics and quantum chains. *arXiv preprint hep-th/9302112*, 1993.

[81] R John Ellis and Allen P Minton. Cell biology: join the crowd. *Nature*, 425(6953):27–28, 2003.

[82] Vijay Chickarmane, Carl Troein, Ulrike A Nuber, Herbert M Sauro, and Carsten Peterson. Transcriptional dynamics of the embryonic stem cell switch. *PLoS computational biology*, 2(9):e123, 2006.

[83] Mads Kærn, Timothy C Elston, William J Blake, and James J Collins. Stochasticity in gene expression: from theories to phenotypes. *Nature Reviews Genetics*, 6(6):451–464, 2005.

[84] Jeroen S van Zon, Marco J Morelli, Sorin Tănase-Nicola, and Pieter Rein ten Wolde. Diffusion of transcription factors can drastically enhance the noise in gene expression. *Biophysical journal*, 91(12):4350–4367, 2006.

[85] Christian Lanctôt, Thierry Cheutin, Marion Cremer, Giacomo Cavalli, and Thomas Cremer. Dynamic genome architecture in the nuclear space: regulation of gene expression in three dimensions. *Nature Reviews Genetics*, 8(2):104–115, 2007.

[86] Thanner M Perumal and Rudiyanto Gunawan. Impulse parametric sensitivity analysis. In *World Congress*, volume 18, pages 9686–9690, 2011.

[87] Luca Corolli, Carlo Maj, Fabrizio Marini, Daniela Besozzi, and Giancarlo Mauri. An excursion in reaction systems: From computer science to biology. *Theor. Comput. Sci.*, 454:95–108, 2012.

[88] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Reaction systems. *Fundamenta Informaticae*, 75(1):263–280, 2007.

[89] CJ Wilson, H Zhan, L Swint-Kruse, and KS Matthews. The lactose repressor system: paradigms for regulation, allosteric behavior and protein folding. *Cellular and molecular life sciences*, 64(1):3–16, 2007.

[90] Carole J Proctor and Douglas A Gray. Explaining oscillations and variability in the p53-mdm2 system. *BMC systems biology*, 2(1):75, 2008.

[91] Richard Hardstone, Simon-Shlomo Poil, Giuseppina Schiavone, Rick Jansen, Vadim V Nikulin, Huibert D Mansvelder, and Klaus Linkenkaer-Hansen. Detrended fluctuation analysis: a scale-free view on neuronal oscillations. *Frontiers in physiology*, 3, 2012.

[92] Tara A Whitten, Adam M Hughes, Clayton T Dickson, and Jeremy B Caplan. A better oscillation detection method robustly extracts eeg rhythms across brain state changes: The human alpha rhythm as a test case. *NeuroImage*, 54(2):860–874, 2011.

[93] Xinong Li, Jiandong Wang, Biao Huang, and Sien Lu. The dct-based oscillation detection method for a single time series. *Journal of Process Control*, 20(5):609–617, 2010.

[94] Gregery T Buzzard. Global sensitivity analysis using sparse grid interpolation and polynomial chaos. *Reliability Engineering & System Safety*, 2011.

[95] Dongbin Xiu. Efficient collocational approach for parametric uncertainty analysis. *Communications in computational physics*, 2(2):293–309, 2007.

[96] Matthias Heinemann and Sven Panke. Synthetic biology—putting engineering into biology. *Bioinformatics*, 22(22):2790–2799, 2006.

[97] Michael B Elowitz and Stanislas Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.

[98] Stefan Müller, Josef Hofbauer, Lukas Endler, Christoph Flamm, Stefanie Widder, and Peter Schuster. A generalized model of the repressilator. *Journal of Mathematical Biology*, 53(6):905–937, 2006.

[99] Natalja Strelkowa and Mauricio Barahona. Transient dynamics around unstable periodic orbits in the generalized repressilator model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 21(2):023104–023104, 2011.

[100] Keun-Young Kim, David Lepzelter, and Jin Wang. Single molecule dynamics and statistical fluctuations of gene regulatory networks: A repressilator. *The Journal of chemical physics*, 126:034702, 2007.

[101] Olguta Buse, Rodrigo Pérez, and Alexey Kuznetsov. Dynamical properties of the repressilator model. *Physical Review E*, 81(6):066206, 2010.

[102] V Ibarra-Junquera, LA Torres, HC Rosu, G Argüello, and J Collado-Vides. Nonlinear software sensor for monitoring genetic regulation processes with noise and modeling errors. *Physical Review E*, 72(1):011919, 2005.

[103] Dario Pescini, Paolo Cazzaniga, Daniela Besozzi, Giancarlo Mauri, Loredana Amigoni, Sonia Colombo, and Enzo Martegani. Simulation of the ras/camp/pka pathway in budding yeast highlights the establishment of stable oscillatory states. *Biotechnology advances*, 30(1):99–107, 2012.

[104] Yu Jiang, Corey Davis, and James R Broach. Efficient transition to growth on fermentable carbon sources in saccharomyces cerevisiae requires signaling through the ras pathway. *The EMBO journal*, 17(23):6942–6951, 1998.

[105] Cecilia Garmendia-Torres, Albert Goldbeter, and Michel Jacquet. Nucleocytoplasmic oscillations of the yeast transcription factor msn2: evidence for periodic pka activation. *Current biology*, 17(12):1044–1049, 2007.

[106] Kevin Gonzales, Ömür Kayikçi, David G Schaeffer, and Paul M Magwene. Modeling mutant phenotypes and oscillatory dynamics in the saccharomyces cerevisiae camp-pka pathway. *BMC systems biology*, 7(1):40, 2013.

[107] Sonia Colombo, Daniela Ronchetti, Johan M Thevelein, Joris Winderickx, and Enzo Martegani. Activation state of the ras2 protein and glucose-induced signaling in saccharomyces cerevisiae. *Journal of Biological Chemistry*, 279(45):46715–46722, 2004.

[108] Daniela Besozzi, Paolo Cazzaniga, Dario Pescini, Giancarlo Mauri, Sonia Colombo, and Enzo Martegani. The role of feedback control mechanisms on the establishment of oscillatory regimes in the ras/camp/pka pathway in s. cerevisiae. *EURASIP Journal on Bioinformatics and Systems Biology*, 2012(1):1–17, 2012.

[109] Betty HC Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, et al. *Software engineering for self-adaptive systems: A research roadmap*. Springer, 2009.

[110] Peter Sawyer, Nelly Bencomo, Jon Whittle, Emmanuel Letier, and Anthony Finkelstein. Requirements-aware systems: A research agenda for re for self-adaptive systems. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 95–103. IEEE, 2010.

[111] Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(2):14, 2009.

[112] Tau-Mu Yi, Hiroaki Kitano, and Melvin I Simon. A quantitative characterization of the yeast heterotrimeric g protein cycle. *Proceedings of the National Academy of Sciences*, 100(19):10764–10769, 2003.

[113] Fabrizio Gagliardi, Bob Jones, François Grey, Marc-Elian Bégin, and Matti Heikkurinen. Building an infrastructure for scientific grid computing: status and goals of the egee project. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 363(1833):1729–1742, 2005.

[114] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International journal of high performance computing applications*, 15(3):200–222, 2001.

[115] Erwin Laure, A Edlund, F Pacini, P Buncic, M Barroso, A Di Meglio, F Prelz, A Frohner, O Mulmo, A Krenek, et al. Programming the grid with glite. Technical report, 2006.

[116] D Kranzlmüller, J Marco de Lucas, and P Öster. The european grid initiative (egi). In *Remote Instrumentation and Virtual Laboratories*, pages 61–66. Springer, 2010.

[117] Ivan Merelli, Dario Pescini, Ettore Mosca, Paolo Cazzaniga, Carlo Maj, Giancarlo Mauri, and Luciano Milanesi. Grid computing for sensitivity analysis of stochastic biological models. In *Parallel Computing Technologies*, pages 62–73. Springer, 2011.

[118] Yang Cao, Daniel T Gillespie, and Linda R Petzold. Efficient step size selection for the tau-leaping simulation method. *The Journal of chemical physics*, 124:044109, 2006.

[119] Daniel T Gillespie and Linda R Petzold. Improved leap-size selection for accelerated stochastic simulation. *The Journal of Chemical Physics*, 119:8229, 2003.

[120] *21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2013, Belfast, United Kingdom, February 27 - March 1, 2013*. IEEE Computer Society, 2013.

[121] 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, 2010.