Design tools for the engineering of biological systems

Elise Rosati

UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE MATHEMATIQUES, SCIENCES DE L'INFORMATION ET DE L'INGENIEUR (ED 269)

Laboratoire des Sciences de l'Ingénieur, de l'Informatique et de l'Imagerie (ICube, UMR 7357)

THÈSE présentée par :

Elise ROSATI

soutenue le : **5 avril 2018**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline : Sciences de l'ingénieur

Outils d'aide à la conception pour l'ingénierie de systèmes biologiques

THÈSE dirigée par :

Mr. LALLEMENT Christophe	Professeur, université de Strasbourg

RAPPORTEURS :

Mr. O'CONNOR Ian	Professeur, école Centrale Lyon
Mr. AMAR Patrick	Maître de conférences HDR, université Paris-Sud

---

AUTRES MEMBRES DU JURY :

Mr. PÊCHEUX François	Professeur, Sorbonne Universités
Mme. DEJAEGERE Annick	Professeur, université de Strasbourg
Mr. MADEC Morgan	Maître de conférences, université de Strasbourg

Remerciements

Tout d'abord, j'aimerais remercier mon directeur de thèse Christophe Lallement, pour avoir fait le pari risqué d'accueillir une biologiste au sein de son équipe. J'aimerais aussi remercier Michel de Mathelin pour m'avoir accueillie au sein de son laboratoire.

Je voudrais remercier Patrick Amar, Ian O'Connor, Annick Dejaegere et François Pêcheux d'avoir accepté de faire partie de mon jury.

Je remercie Jacques Haiech qui a été à l'origine de discussions enrichissantes sur la biologie ainsi que Pierre Collet pour m'avoir initiée au monde des algorithmes évolutionnaires.

Merci aussi et surtout à mon encadrant de thèse, Morgan Madec, qui a été mon interlocuteur privilégié au cours de ces 3 ans. Les discussions scientifiques sont toujours riches et le quotidien de ma thèse s'est trouvé nettement amélioré par sa constante disponibilité et gentillesse. Merci aussi de m'avoir fait découvrir les merveilleux mondes de l'électronique numérique et de l'enseignement.

Mes remerciements vont à Luc Hébrard, pour son expertise et le temps qu'il m'a dédié au sujet des éléments finis. Merci aussi à Yoshitate Takakura pour son aide sur le plan expérimental et théorique. Je tiens aussi à remercier les membres de l'équipe, pour les diverses conversations que j'ai pu avoir avec chacun d'entre vous, sur des sujets scientifiques mais pas que… Je tenais tout particulièrement à remercier Norbert, sans qui les pauses midi auraient connu beaucoup moins de débats et d'aléas.

Côté biologiste, j'aimerais aussi remercier Annie-Paule Sibler, qui m'a ouvert les portes des salles de TP et offert une aide précieuse.

J'aimerais remercier tous les stagiaires et les étudiants en projet qui ont été impliqués de près ou de loin à mon travail de thèse, et notamment : Anaïs, Michaël, Nicolas, Rémi. Vous êtes les cailloux qui ont parsemé mon chemin.

Je tenais aussi à mentionner les doctorants de l'équipe (et orbites proches), bien que rencontrés sur le tard (même si n'étant parfois pas loin du tout !), pour les sympathiques soirées passées en votre compagnie : Alexi, Alexis, François, Lucas, Timothé, Vinh.

Parce que je ne vous considère plus comme mes stagiaires mais comme mes amis, je tenais à vous remercier dans un paragraphe à part, vous la petite bande sans qui ces 3 ans (et quelques) auraient été bien moins agréables et drôles : Florian, Laurent, Quentin, Thomas, Thomas, Zoé

Merci à tous mes amis, avec qui je n'ai cessé et ne cesserai de passer de bons moments.

Je n'oublie pas ma famille, dont le soutien s'est manifesté en de nombreuses occasions. Vous avez toujours été présents, jusqu'à la dernière minute !

Et enfin, last but not least, j'aimerais exprimer toute ma gratitude à mon compagnon, pour son indéfectible soutien et sa précieuse aide pendant cette ardue période de rédaction. La liste complète de tes hauts faits est bien trop longue pour apparaître ici ! Merci pour tout.

Abstract

In synthetic biology, Gene Regulatory Networks (GRN) are one of the main ways to create new biological functions to solve problems in various areas (therapeutics, biofuels, biomaterials, biosensing). However, the complexity of the designed networks has reached a limit, thereby restraining the variety of problems they can address. How can biologists overcome this limit and further increase the complexity of their systems?

The goal of this thesis is to provide the biologists with tools to assist them in the design and simulation of complex GRNs. To this aim, the current state of the art was examined and it was decided to adapt tools from the micro-electronic field to biology, as well as to create a Genetic Programming algorithm for GRN design. On the one hand, models of diffusion and of other various systems (band-pass, prey-predator, repressilator, XOR) were created and written in Verilog A. They are already implemented and well-functioning on the Spectre solver as well as a free solver, namely NgSpice. On the other hand, the first steps of automatic GRN design were achieved. Indeed, an algorithm able to optimize the parameters of a given GRN according to a specification was developed. Moreover, Genetic Programming was applied to GRN design, allowing the optimization of both the topology and the parameters of a GRN.

These tools proved their usefulness for the biologists' community by efficiently answering relevant biological questions arising in the development of a system. With this work, we were able to show that adapting micro-electronics and Genetic Programming tools to biology is doable and useful. By assisting design and simulation, such tools should promote the emergence of more complex systems.

Résumé

En biologie synthétique, il existe plusieurs manières d'adresser les problèmes soulevés dans plusieurs domaines comme la thérapeutique, les biofuels, les biomatériaux ou encore les biocapteurs. Nous avons choisi de nous concentrer sur l'une d'entre elles : les réseaux de régulation génétique (RRG). Un constat peut être fait : la diversité des problèmes résolus grâce aux RRGs est bridée par la complexité de ces RRGs, qui a atteint une limite. Quelles solutions s'offrent aux biologistes, pour repousser cette limite et continuer d'augmenter la complexité de leur système ?

Cette thèse a pour but de fournir aux biologistes les outils nécessaires à la conception et à la simulation de RRGs complexes. Un examen de l'état de l'art en la matière nous a mené à adapter les outils de la micro-électronique à la biologie ainsi qu'à créer un algorithme de programmation génétique pour la conception des RRGs. D'une part, nous avons élaboré les modèles Verilog A de différents systèmes biologiques (passe-bande, proie-prédateur, repressilator, XOR) ainsi que de la diffusion spatiotemporelle d'une molécule. Ces modèles fonctionnent très bien avec plusieurs simulateurs électroniques (Spectre et NgSpice). D'autre part, les premières marches vers l'automatisation de la conception de RRGs ont été gravies. En effet, nous avons développé un algorithme capable d'optimiser les paramètres d'un RRG pour remplir un cahier des charges donné. De plus, la programmation génétique a été utilisée pour optimiser non seulement les paramètres d'un RRG mais aussi sa topologie.

Ces outils ont su prouver leur utilité en apportant des réponses pertinentes à des problèmes soulevés lors du développement de systèmes biologiques. Ce travail a permis de montrer que notre approche, à savoir adapter les outils de la micro-électronique et utiliser des algorithmes de programmation génétique, est valide dans le contexte de la biologie synthétique. L'assistance que notre environnement de développement fournit au biologiste devrait encourager l'émergence de systèmes plus complexes

Table of Contents

Glossary of terms

ABC	The part compiler: combines element of a library to achieve a target function
AHL	Acyl Homoserine Lactone, a small molecule involved in cell-to-cell communication
Band-pass	System that exhibits a bell-shaped response
BioBrick	Also called part, DNA sequences that are the building blocks of GRNs
BLIF	Berkeley Logic Interchange Format, a language used to represent a logic circuit at a high-level of abstraction
Boolean	Boolean logic is a paradigm where entities can have two values, 0 or 1
Bottom-up	Design methodology composing a system starts by assembling elementary building blocks
CAD tools	Computer-Aided Design tools
COPASI	A software application for simulation and analysis of biochemical networks and their dynamics
Design flow	Set of methods and tools that are used sequentially during the process of system's design
DNA	Deoxyribonucleic acid, the base block of genetic information encoding
EDA	Set of tools for Electronic Design Automation
Evolutionary algorithms	Population-based metaheuristic optimization algorithm inspired by biological evolution
Flip-flop	Synchronous memory that is updates only on the tick of a clock signal
HDL	Hardware Description Language, language used for the description of hardware (electronic devices and systems)
Genetic Programming	A kind of evolutionary algorithm that manipulate programs
GPL	Genetic Part Library, a library of biological parts
Grounded	In electronics, connected to the reference of the circuit
GRN	Gene Regulatory Networks
Leakiness	Residual transcription of an inactivated gene
miR	Micro-RNA, non-coding RNA able to silence a gene by binding to its mRNA and inhibiting its translation and/or initiating its degradation
mRNA	Messenger RNA, an RNA molecule processed to be translation-ready
MSE	Mean Square Error
ODE	Ordinary Differential Equations
ODIN II	A digital synthesizer, converts a Verilog file into an RTL-netlist
PDE	Partial Differential Equations
RBS	Ribosome-binding site
RNA	Ribonucleic acid, intermediary molecule in the expression of a gene
RTL netlist	Register Transfer Level netlist, a circuit composed of Boolean functions and memories
SBML	Systems Biology Markup Language. SBML is a widely used format for computer models of biological processes

Search space	The space defining the set of all possible solutions of an optimization algorithm
SPICE	Simulation Program with Integrated Circuit Emphasis. SPICE is a freeware used to run different kind of simulation of analog electronic circuits.
Top-down	Design methodology which consists in decomposing the system hierarchically down to elementary building blocks
USBO	Universal Set of Boolean Operators, a set of Boolean operators sufficient to produce any Boolean expression when combined.
VCCS	Voltage Controlled Current Source
Verilog	a hardware description language used to describe digital electronic systems.
Verilog-A	a hardware description language used to describe analog and mixed electronic systems.
VHDL	VHSIC (Very High Speed Integrated Circuit) Hardware Description Language, a language used to describe the structure and the behavior of electronic systems.

General Introduction

Synthetic biology is a relatively recent field of investigation, as it emerged in the 2000's. Synthetic biology requires a general know-how in many disciplines and is commonly defined as the application of engineering principles on biology (http://www.synbioproject.org/topics/synbio101/definition/), although many definitions of the term can be found. The term appeared first under the pen of Stephane Leduc early in the XXth century, who declared that biology should be "successively descriptive, analytic and synthetic" (Le Duc 1910). The idea behind this is that in order to truly understand a biological phenomenon, one has to be able to recreate it. If description and analysis of biological organisms and systems are well-established disciplines, synthesis, hence synthetic biology is only at its beginnings.

Fig. 1 List of the words most commonly associated with Synthetic biology. Source: http://www.synbioproject.org/topics/synbio101/definition/, consulted on 22/11/16. Generated with www.worldclouds.com

Key words such as "design", "engineering", "systems", "devices", "parts" and "function" are often found in definitions of synthetic biology (Fig. 1). This is well illustrated by the definition established by a group of European experts, the European Commission for Biotechnologies, Agriculture and Food:

> *Synthetic biology is the engineering of biology: the synthesis of complex, biologically based (or inspired) systems which display functions that do not exist in nature. This engineering perspective may be applied at all levels of the hierarchy of biological structures – from individual molecules to whole cells, tissues and organisms. In essence, synthetic biology will enable the design of 'biological systems' in a rational and systematic way.*

Typical established characteristics of synthetic biology are summarized by the SynBERC (a U.S. research program) which states that synthetic biology's hallmarks are the following ("What Is Synthetic Biology? - Synthetic Biology Project" 2017):

> *predictable, off-the-shelf parts and devices with standard connections, robust biological chassis (such as yeast and E. coli) that readily accept those parts and devices, standards for assembling components into increasingly sophisticated and functional systems and open-source availability and development of parts, devices, and chassis.*

The application range of synthetic biology is very large, including therapeutics, the creation of innovative biomaterials, biosensors ... This field of research has several aspects. The main one concerns biotechnologies allowing the modification of existing biological systems or the creation of artificial biological systems. In parallel, efforts have been made in order to develop tools to facilitate the design process of these systems. This is the context of this thesis.

During the past few years, designed systems have grown in size and in complexity. The development of efficient computer-aided design (CAD) tools is therefore crucial. Many research teams from all over the world worked on the development of such tools, whether in a generic way or *ad hoc* to an application. Our team suggested a few years ago an alternative that consists in extending existing tools from engineering sciences to synthetic biology. This approach has two main advantages: i) engineering sciences' CAD tools are reliable and have already proven themselves in the recent decades and ii) adding synthetic biology to these CAD tools might also facilitate the integration of biological parts into large-scale transdisciplinary systems such as lab-on-chips or bio-sensors.

The starting point of this thesis is an existing environment composed of different tools that form a complete workflow, ranging from the specification of a biological system at a high level of abstraction to its practical realization. Several bricks of this workflow have already been developed by previous PhD or master students, in particular those related to the modeling and simulation of biological systems. My contribution to this work concerns two missing tools: the improvement of the design automation process for gene regulatory networks (GRN) during the early stages of the design process and the development of a simulation tool to support models for which spatial location plays a major role.

For the first point, as for electronic circuits, the automation of GRN design is highly dependent on their type and complexity. For GRNs that can be described by a Boolean behavior, the solution we implemented consists in reusing digital synthesizer from micro-electronics. Nevertheless, this tool has to be accurately parametrized because on some points, the specificities of GRN and electronic circuits differ. Digital circuits are split into two main families: the combinatorial circuits and the sequential circuits. The design of sequential circuits is generally trickier because of internal feed-back loop that may cause instability and malfunctions. This point is also addressed in this manuscript and solutions are brought to light.

On the other hand, for GRNs that are described by an "analog" function (transfer function, temporal or frequency characteristics, etc), digital synthesizer cannot be employed. Despite active research in the domain of analog circuit synthesizer, the design of such circuits is still mostly handmade. In this

thesis, we evaluate the potential of nature-inspired algorithms for the design automation of such GRN. In particular, two kind of algorithms are implemented: evolutionary algorithms that can be used to optimize the parameters of a model in order to fit a targeted response and genetic algorithms which evolve both the parameters and the model itself to reach a response defined *a priori*.

The other major contribution to the workflow concerns the development of a simulation tool to handle models mixing both local phenomena (biochemical mechanisms confined to a given space, diffusion through membranes or walls) and global phenomena (free diffusion in a medium, degradation, etc). This kind of systems are more and more encountered in systems biology or in synthetic biology. From a modeling point of view, their particularity is that they are no longer described by ordinary differential equations but by partial differential equations. The resolution of such equation requires quite sophisticated algorithms of space discretization and of resolution of large sets of equations. To tackle this issue, we take inspiration from a similar problem encountered in the design of microelectronics integrated circuits: the electro-thermal simulation. We dispose in our team of a tool developed for a previous project. Our approach has been to adapt this tool to the biological context. In this manuscript, this approach is presented, as well as the results obtained on actual biological systems.

The manuscript is divided into three main parts. Part Two is dedicated to the design automation of biological systems while Part Three describes the spatiotemporal simulator mentioned above. Beforehand, the main concepts for understanding the ins and the outs as well as the content of this work are discussed in Part One. This part is itself divided into three chapters. Chapter 1 is an introduction to synthetic biology and its applications, giving the general framework of this thesis. Chapter 2 provides the reader with the theoretical background in electronics and in biology required all along this manuscript. Finally, Chapter 3 deals with the design approach used in synthetic biology and presents a state-of-the-art of existing computer-aided design tools for synthetic biology.

Part One

Context and Objectives

Chapter 1
Synthetic Biology and Applications

1. A brief history of synthetic biology

Synthetic biology uses knowledge and techniques that have been proving their worth for decades. Here, we retrace the emergence of synthetic biology and its main enabling techniques. The most significant milestones are given in a timeline on Fig. 1.

The design of novel biological devices requires a variety of techniques, from DNA manipulation to cell culture as well as a good understanding of biological systems. One could date the first milestone that enabled the emergence of synthetic biology to 1961, when François Jacob and Jacques Monod established the model of an operon, inducing the first notions of regulatory circuits (Jacob and Monod 1961). In the following 40 years, many technical steps were taken, such as the first Polymerase Chain Reaction (PCR) (Mullis et al. 1986), the beginning of molecular cloning (Cohen et al. 1973; Mullis et al. 1986) or later affordable DNA sequencing methods via high throughput techniques (Fahnestock et al. 1991; Ronaghi et al. 1996). The latter notably lead to the sequencing of the whole human genome (draft version in 2001 (Lander et al. 2001; Venter et al. 2001), final version in 2004 (Hattori 2005)). Later on, so-called second generation high throughput DNA sequencing techniques arrived on the market (Illumina's Genome Analyzer II in 2006 (Mardis 2008), Life Technologies' Ion torrent's device in 2010 (Rothberg et al. 2011), a portable sequencer from Oxford Nanopore Technologies in 2014 (Mikheyev and Tin 2014)), greatly reducing the cost of sequencing.

This decrease is often compared to Moore's law (Moore 1965). Moore predicted that the number of transistors integrated per microprocessor would double every two years. When observing the cost of DNA sequencing (Fig. 2), we see that it decreases by a factor two every year, up until 2007 when the cost of sequencing dropped significantly more than predicted. This corresponds to the arrival on the market of the second-generation of sequencing devices.

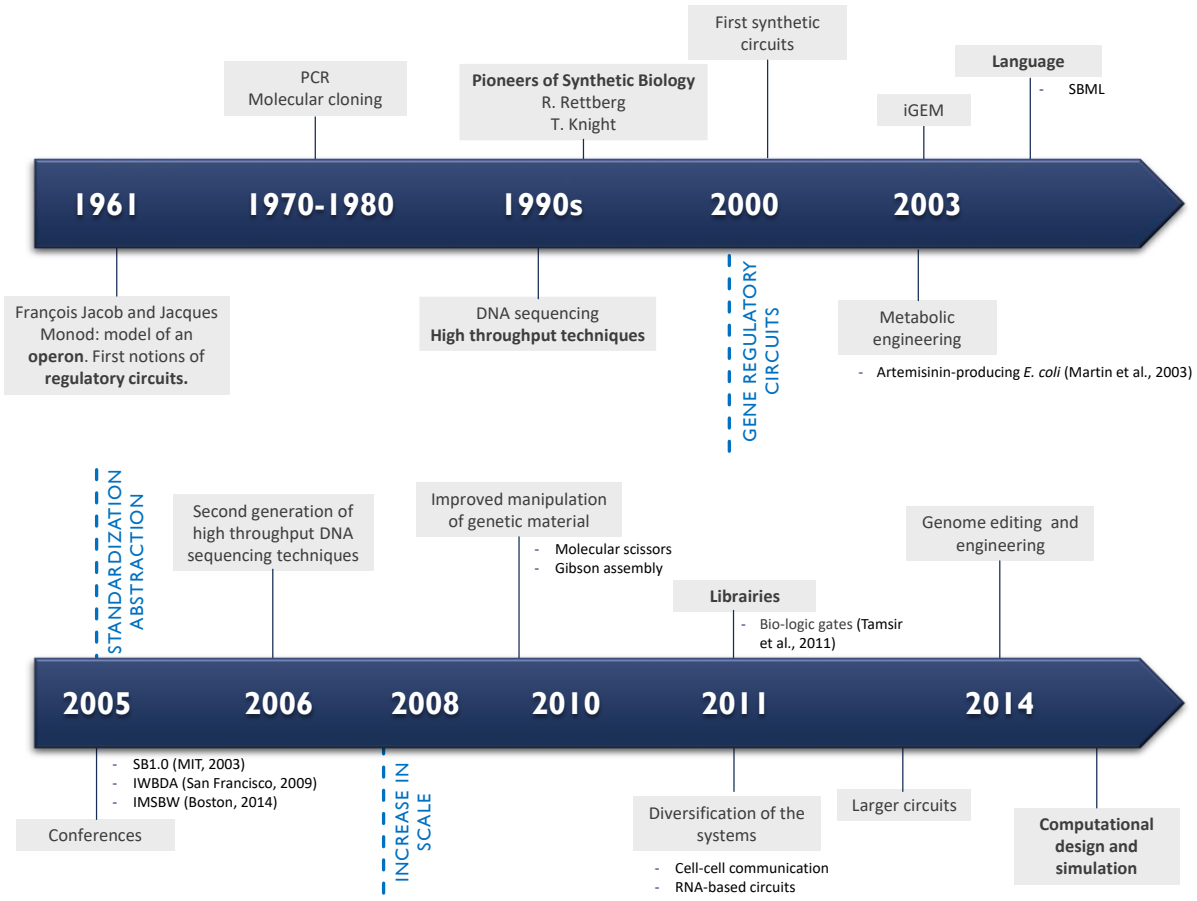


Fig. 1 Timeline of the main events and hallmarks leading to the development of synthetics biology

With the possibility to read DNA came the need to edit genes and genomes. To edit a genome, which is to insert, delete or replace DNA at a specific site in the genome, scientists developed so-called molecular scissors. They include nucleases, such as CRISPR with its associated protein Cas9 (Sander and Joung 2014), zinc finger proteins (Laity, Lee, and Wright 2001; Urnov et al. 2010) and TALE nucleases (Miller et al. 2011). Other techniques use viral systems such as the Recombinant Adeno-Associated Virus. These are tools we can engineer to target and edit a specific region in a genome. Novel molecular cloning techniques such as Gibson assembly and Golden gate (Engler, Kandzia, and Marillonnet 2008) facilitated the building of constructs made of several DNA fragments. Among others, these techniques allow an easy and repeatable manipulation of DNA material.

The pioneering of synthetic biology as a field of investigation *per se* can be attributed to Randy Rettberg and Tom Knight. Their story illustrates nicely the multidisciplinarity of synthetic biology and gives an interesting point of view on how to envision the scope of synthetic biology. R. Rettberg and T. Knight were computer and electrical engineers who both had had a successful career. Knight participated in the development of ARPAnet, the pre-internet. Rettberg has behind him a 30-years career as an executive in the computer industry (at Apple Computer and Sun Microsystems). In the 90s, these two engineers decided to learn all they could about biology, with the idea that the modularity found in biological systems would allow the design and rewiring of cells. Their ultimate goal was to make biological circuits through the assembly of standardized “parts”, just like electronical circuits are composed of an assembly of resistors, capacitors and transistors. Such circuits would allow synthetic biologists to perform highly specific and sophisticated tasks that would not be possible with only classical genetic engineering principles. They started a lab with Ron Weiss as a PhD student, a scientist

who has now become an important actor in the synthetic biology community (a tenured professor at MIT, with 33 publications on PubMed on Synthetic Biology). The first synthetic gene regulatory circuits appeared in the beginnings of the 21st century. Noteworthy are the repressilator (Elowitz and Leibler 2000), a three-genes oscillating system and the toggle switch (Gardner, Cantor, and Collins 2000), a bistable system able to switch from one state to the other depending on an external environment cue.

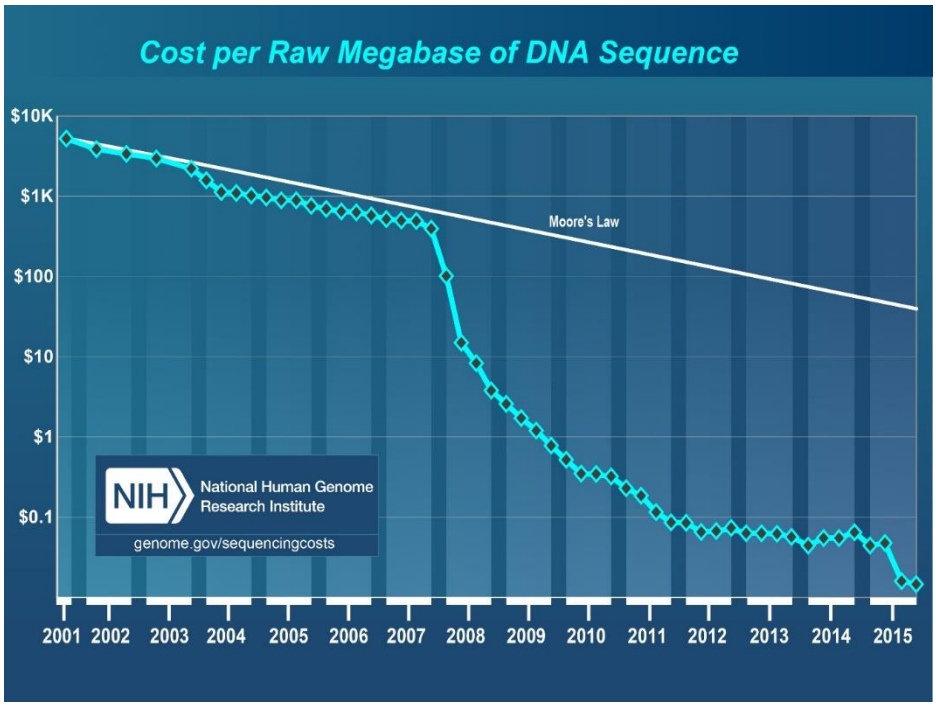


Fig. 2 Cost per raw megabase of DNA sequence compared to Moore's law (https://www.genome.gov/images/content/costpermb2015_4.jpg)

In 2003, Rettberg, Knight and Drew Endy started the iGEM (International Genetically Engineered Machine) competition, where undergraduate teams have a few months to elaborate any synthetic biology-related system based on the BioBrick standard biological parts library ("Registry of Standard Biological Parts" 2015). This library is a database of plasmids commonly used in synthetic biology and is equivalent to these found in electronics catalogs and design kits. Growing from a 5-team competition, iGEM became a worldwide highly wanted event with now, in 2017, 14 tracks gathering 5600 participants from 42 different countries. iGEM is held every year and its final is always at Boston, MA. After the project, each team has to the share the new DNA parts they may have come up with, therefore increasing the pool of available constructs.

Alongside with genetic regulatory networks, scientists also started to investigate metabolic systems and redesign them to enhance them or provide them with novel proprieties, with for example the reprogramming of *Escherichia coli* (*E. coli*) bacteria into artemisinin producing factories (Ro et al. 2006). Moreover, protein design shifted from random mutagenesis to directed evolution: scientists started to design proteins with novel structural and ligand-binding characteristics, specific protein-protein interactions or with hyperstable protein folds (Yoder and Kumar 2002).

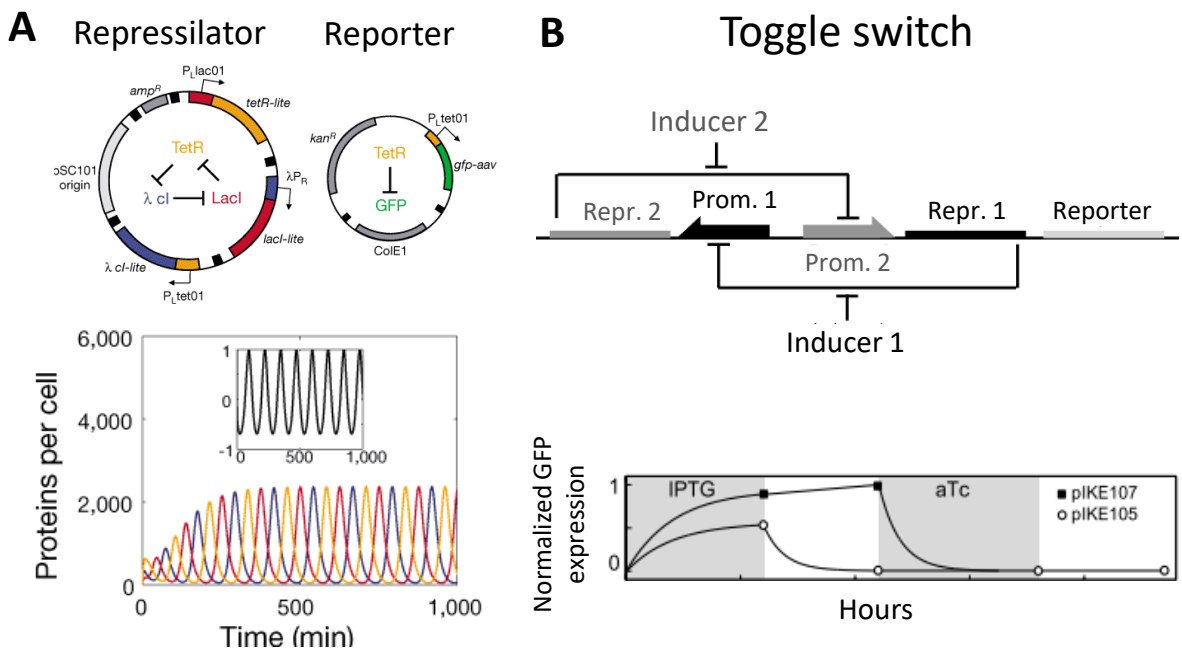


Fig. 3 Synthetic biology oscillator (the repressilator on the left, (Elowitz and Leibler 2000)) and switch (the toggle switch on the right, (Gardner, Cantor, and Collins 2000)), both constructed in bacteria. The graph on the left shows the oscillations of the three repressor proteins in the repressilator. The graph on the right shows a working toggle switch (pIKE107) and one that does not exhibit bistability as expected (pIKE105).

With a growing community appeared the first international conferences. In the U.S., the international conference for synthetic biology, SB1.0, was held at the MIT in 2003. Later on, other conferences emerged such as the 1st International Workshop on Bio-Design Automation (San Francisco 2009), or more recently the International Mammalian Synthetic Biology Workshop (Boston 2014) and the Synthetic Biology: Engineering, Evolution & Design (Los Angeles 2014). In Europe, the annual conference advances in Systems and Synthetic Biology (aSSB) had its first occurrence in 2002 in Grenoble, France. Synthetic biology also appeared in other major conference, whose thematic was related to synthetic biology in a way. For example, in 2009 IEEE International Symposium on Circuits and Systems (ISCAS) had a special session dedicated to synthetic biology.

In the following years, a focus was put on standardization of the parts. This promoted applications such as the library of bio-logical gates established by (Tamsir, Tabor, and Voigt 2011). Synthetic biologists diversified their circuits, by using RNA (Xie et al. 2011) or by redesigning natural cell-to-cell communication systems (Bacchus and Fussenegger 2013). As shown by the previous examples, the systems designed by the scientists became larger, containing up to a dozen of genes, or spread on different cells. Another field of interest in the community of synthetic biology is genome editing and engineering, with the ultimate goal to find a minimal genome. The idea behind this project is close to the fundamental perspective of Stephane Leduc, in that creating a protocell, a fully synthetic and functional cell, would ensure deep understanding of biological phenomena and evolution, as well as giving a ready-to-use chassis for synthetic systems. An ongoing example can be found in project Sc2.0 (Richardson et al. 2017). The goal is to modify the yeast (*Saccharomyces cerevisiae*) genome, to strip it from as many non-essential genes as possible, chromosome by chromosome, in a bottom up approach. In this project, many teams from all over the world collaborate (see Fig. 4), each one working on a different chromosome. This is not the first project that requires scientists from different teams to collaborate: in order to simplify communication, it soon appeared that a common language was

needed. Just like a micro-electronic system can be described in Verilog A, synthetic biologists created SBML (System Biology Markup Language (Hucka et al. 2003)), an HTML-like language for the description of biological systems. It features lists of species, compartments, reactions rates…

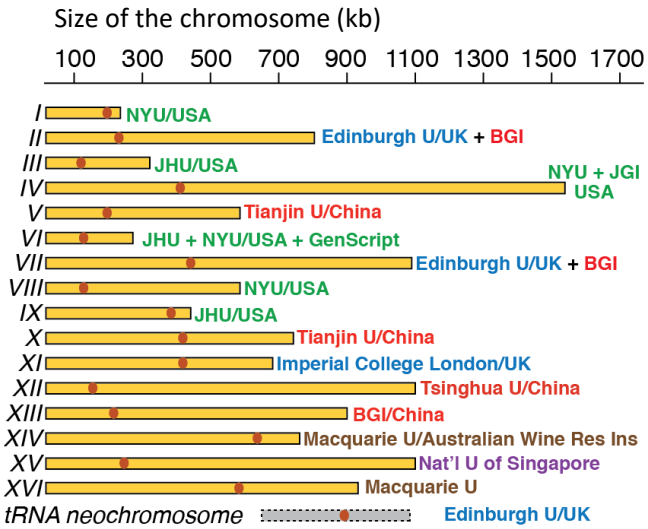


Fig. 4 Project Sc2.0 : list of Saccharomyces cerevisiae chromosomes and teams devoted to it (Richardson et al. 2017). Each line represents a chromosome of the yeast, the name on the right correspond to the team who worked on it.

The development of a common language, though human-readable, helped the development of user-friendly design and simulation tools, providing an easier manipulation of SBML files. Software such as Copasi (Hoops et al. 2006) or VCell (XML-based at date of creation) allow the simulation of biological reactions through a set of ordinary differential equations (ODE), and therefore the simulation of gene regulatory networks.

Currently, a new effort is made to increase the complexity of designed systems, by for example distributing networks on different cells. The wet-bench research is also accompanied by research in models and tools to implement the latter, as well as design tools, since "computational tools […] should make routine the engineering of biology", says Drew Endy (Endy 2005), who also states that in synthetic biology, one should "consider past lessons from other engineering disciplines".

2. Applications

Over the last decades, a growing interest was demonstrated towards synthetic biology (Fig. 5). Not only does it implicate scientists from various domains and display interesting challenges, it also opens the door to a next level of applications.

2.1. Therapeutics

In medicine, drug delivery is a hot topic which has already received many solutions. However, the crucial problem of delivering a drug in a specific way, in terms of target and timing, still lacks relevant solutions. Synthetic biology approaches promise such a delivery via elaborate genetic networks embedded in living or non-living compartments. Indeed, these systems allow the sensing of various parameters (O2 concentration, cell density, mRNA pattern…) to "decide" what command to execute: slowly delivering the drug, fast lysis of surrounding cells, etc. Progress in this direction includes the

work of Xie et al.(Xie et al. 2011), with a system allowing the detection of cancerous cells, a first step towards curing cancer. Spatially precise targeting can be envisioned with the work of Basu et al. (Basu et al. 2005), whose system is able to sense a continuous cue (a molecular gradient) for a timed and precise response.

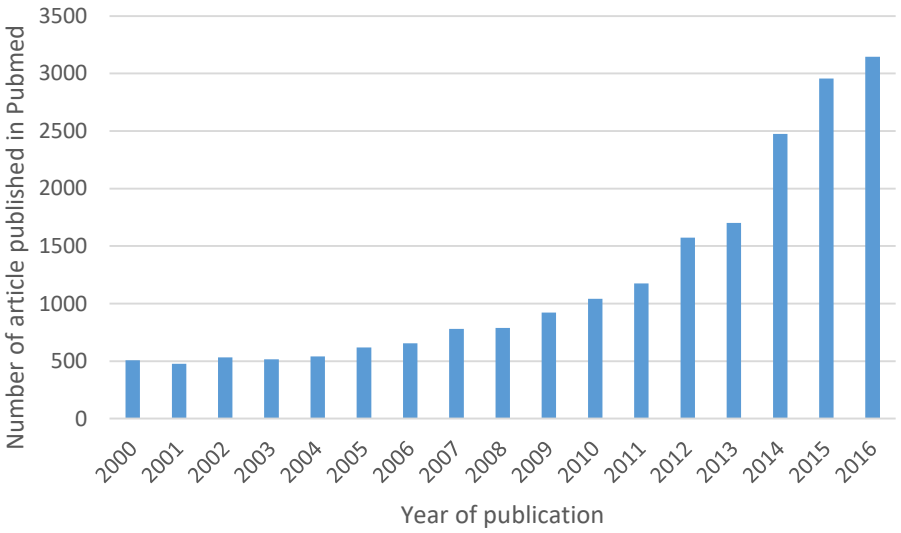


Fig. 5 Increase in research interest on synthetic biology

The manufacturing of drugs to be delivered is also a field of investigation for synthetic biologists, as metabolic engineering would allow low-cost synthesis of these (Ro et al. 2006). A famous example is the Golden Rice (Paine et al. 2005). Researchers from Syngenta enabled the production of beta-carotene, a precursor of vitamin A, in the edible parts of rice, by adding two genes from others species (namely *psy* and *crtI*). With this rice, vitamin A-deficient population would be able to alleviate their lack of it at a low cost.

Synthetic biology was also used to design a new metabolic function in mammalian cells with the potential of curing obesity (Kemmer et al. 2011).

2.2. Biofuels

Metabolic engineering can also find applications in the environment with the production of biofuels. Replacing fossil fuels by biofuels is the road toward a sustainable bioeconomy, an attractive goal for many. Biofuel production relies on an efficient use of the biomass, which is renewable.

The current main ideas are to take advantage of the photosynthesis machinery of various organisms or to engineer autotrophic microorganisms (Aro 2016). Cyanobacteria, prokaryotic microorganisms, can be used as cell factories to produce biomass or directly fuel, from light and/or CO_2. Various synthetic biology tools were applied to cyanobacteria (e.g. a controllable synthetic promoter library like TetR- regulated promoters (Huang and Lindblad 2013), oxygen-responsive genetic circuits (Immethun et al. 2016), the CRISPR/Cas9 system (Wendt et al. 2016)). With these tools at hand, it is possible to tinker the metabolic pathways of *Synechococcus elongates*, a cyanobacterium, into producing free fatty acids from CO_2, a potential biofuel precursor (Ruffing and Jones 2012). This study and many others reported that producing and excreting a product not naturally present in the microorganism, or in lesser amounts, can lead to reduced cell growth. A notable consequence is a low

yield in the excreted product of interest. Synthetic biology appears here as a promising technology: an envisioned solution is switchable systems. The organism alternatively grows without producing the molecule of interest or, on a certain criterion, switches to producing it (Aro 2016).

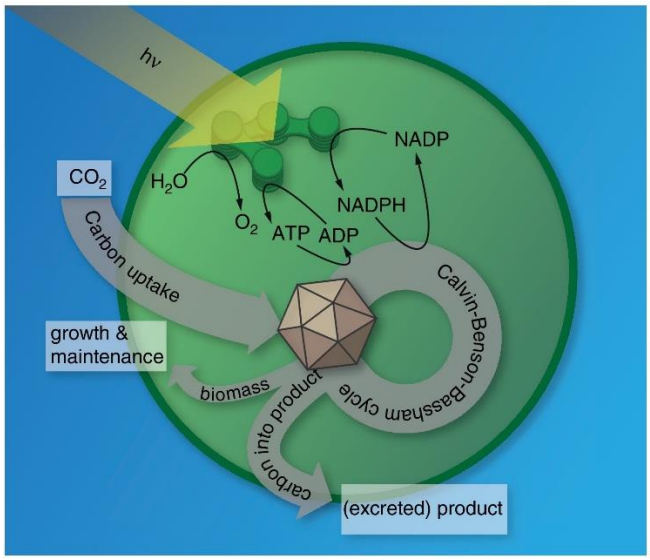


Fig. 6 Example of an engineered micro-organism. Here cyanobacteria metabolic pathways are designed to produce biofuels (Savakis and Hellingwerf 2015).

2.3. Biosensors

In the environmental domain, focus is also put on biosensors. The idea is to have a sensory module responsive to the environment and a transducer module delivering a biological response based on the state of the sensory module (see Fig. 7).

A classical solution is to have an environment-responsive promoter sensitive to a molecule you want to detect (like a pollutant). This promoter is coupled to a genetic circuit, in charge of processing the information detected by the promoter (more information on genetic circuit will be given in Chapter 3). This genetic circuit has the final goal to output a signal like a protein (for example a toxin or a reporter protein) in response to the presence or absence of the molecule. This is illustrated by the *E. coli* bacteria engineered by Kobayashi et al. (2004) which combine a toggle switch (described above) with an acyl-homoserine lactone (AHL)-sensitive module. Their system was able to detect and memorize the presence of the AHL molecule in the medium. The next step would be to change the sensibility of the system: instead of AHL, have a system sensitive to a relevant molecule.

Another typical illustration is the work of Wang et al. (Wang, Barahona, and Buck 2013) in which they describe a set of biosensors they engineered (Fig. 8). These cellular biosensors allow the precise identification of various chemical signals, and combinations thereof, with a quantitative fluorescent output. We find in this example an actuation of each module presented on Fig. 7, with the different inputs, the sensory part, the internal logic circuit and the output signal.

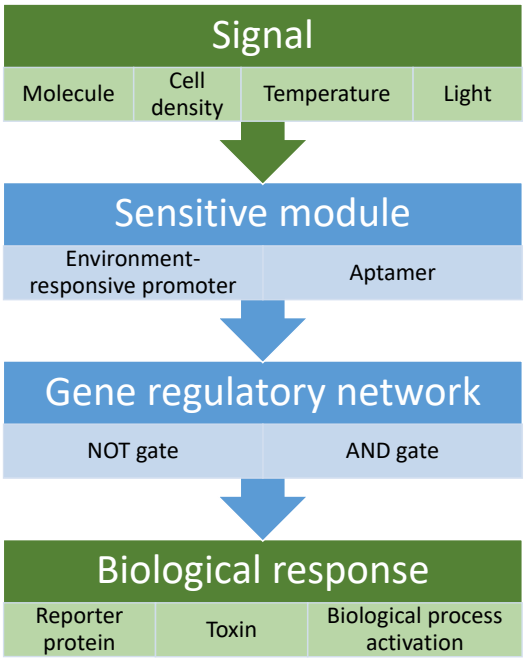


Fig. 7 Components of a biosensor. The green components (Signal and Biological response) are typically expected to interact in the environment whereas the blue components (Sensitive module and Gene regulatory network) are traditionally embedded in a cell, though recent focus has been put on cell-free systems (Zhang and Ruder 2015).

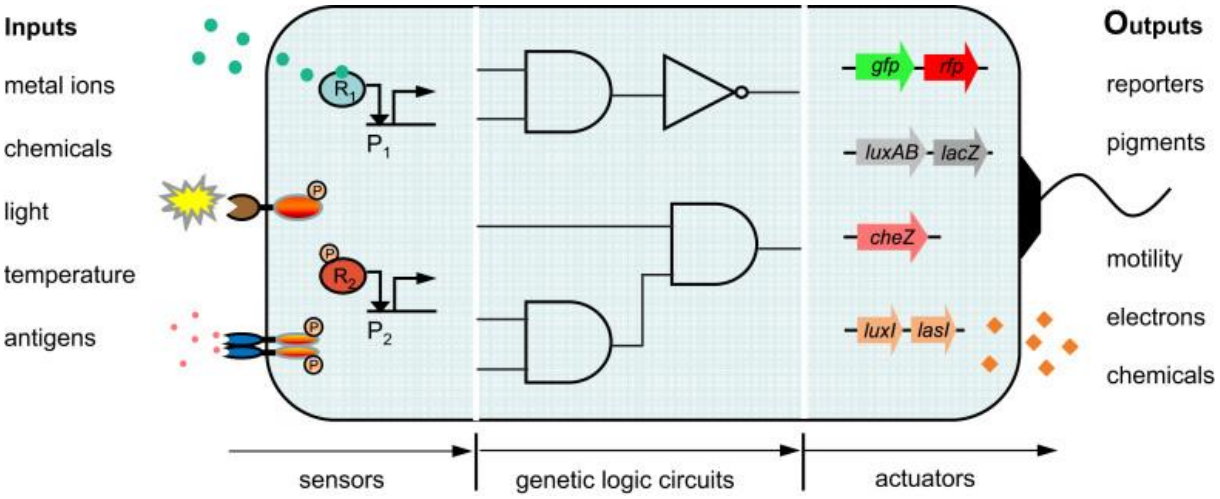


Fig. 8 Architecture of a synthetic modular cell-based biosensor (Wang, Barahona, and Buck 2013). The modules described on Fig. 7 can be found here in an actual biosensor.

Biosensors are somewhat modular: with adjustments of the different components (like the properties of the promoters) it is possible to envision a wide variety of actors for each part. For example, the signal could be any biologically-detectable signal, like cell density, temperature or light. Reading the signal can be done by environment-responsive promoters but also by aptamers, which are RNA molecules able to recognize and bind a specific molecule. Upon binding, the RNA aptamer changes its conformation, resulting in translation activation or inhibition. Both of these modules can be connected to different types of gene regulatory networks, like a NOT gate for a one input gate or an AND gate for a two inputs gate (Anderson, Voigt, and Arkin 2007). Finally, the outcome being the expressed protein,

it can be fully modular: the gene can code for a reporter protein, or a toxin, or even trigger a biological process.

2.4. Other domains

What falls under the label 'wild and crazy' ideas: biocomputers. We are already capable of storing data on DNA (Bornholt et al. 2016; Mike Brunker 2016). Qian et al. also reported the possibility to create neural networks with DNA (Qian, Winfree, and Bruck 2011). Their system relies on DNA base-pairing properties (see Chapter 2 for more information on so-called Watson-Crick base-pairing). They use DNA strand displacement cascades, in which a single-stranded DNA binds to a partially double-stranded DNA. Upon binding, the originally bound strand is released. Such a reaction can also produce a fluorescent signal. With this system, they could build a scalable neural network able to remember 4 single-stranded DNA patterns (see Fig. 9).

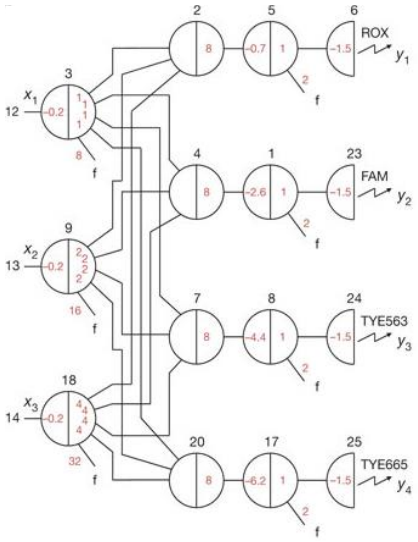


Fig. 9 Biological neural network implemented with DNA molecules. (Qian, Winfree, and Bruck 2011)

3. Main synthetic biology stakeholders

As synthetic biology is still in its infancy, research groups in Europe are still scattered and concentrated in a few teams (Gaisser et al. 2009). This report also highlights the need for increased interdisciplinary research to access the next step in the roadmap they established for synthetic biology. They also state that the situation is more advanced in North America.

Here is a list of a few teams working on synthetic biology and whose work we came across during the making of this thesis:

- At the MIT (USA) work Ron Weiss and Tom Knight
- At Berkeley (USA) work Adam Arkin and Drew Endy
- At the Genopole (France), the first synthetic biology lab iSSB (institute of Systems and Synthetic Biology)
- At the Imperial College London (England) work Tom Ellis and Geoff Baldwin
- Part of the Systems Biology program at the Centre for Genomic Regulation (Spain), Luis Serrano's team focuses on synthetic biology

These are only a few examples highly biased by the course taken by this thesis.

To reflect the impact on synthetic biology on the different countries, I used my personal reference library of articles to try to see where most of the published work comes from. To this end, I searched in the keyword section for the name of one of these countries. The pie chart below (Fig. 10) shows the proportion of articles according to the country filled as a keyword. This procedure is somewhat inaccurate in the sense that an article can show up in a research for a given country even though it was not produced by a team from this country (e.g. the name "France" could appear in the text body).

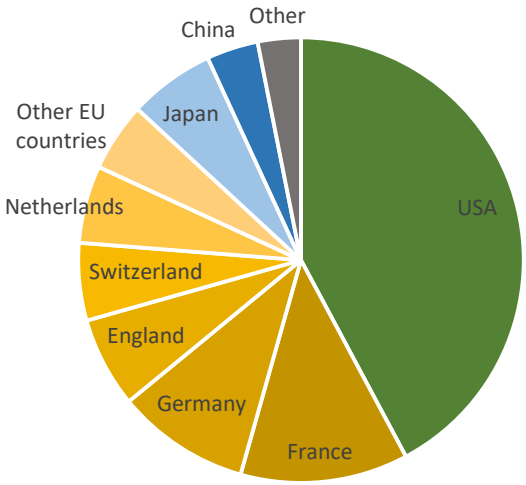


Fig. 10 Country of origin of 320 articles on the subject or related to synthetic biology. Other EU countries include Spain and Belgium. Other corresponds to the number of article not found by any of the queries showed in this chart.

The fact that France appears as the first of the European countries in terms of published articles is most probably a bias due to the fact that I have worked in a French lab with mainly French collaborators. However, a clear observation is the fact that the USA are the leader in the field. This personal analysis is confirmed by a large-scale rational scientometric analysis performed by Raimbault in (Raimbault, Cointet, and Joly 2016).

These elements all lead to the conclusion that trying to catch up with the USA, which are also a step ahead, will prove difficult and probably fruitless in terms of innovation. Instead of competing on the same ground, our research team promotes an alternative approach that puts synthetic biology as a new field of physics and engineering. By this way, we would like to bring an alternative approach to existing solutions, especially in terms of design methodologies and computer-aided design tools, which take up concepts from more mature domains of physics (as for instance electronics, mechanics and optics). This approach provides solutions built on tools whose reliability and robustness is no longer to prove. Moreover, these solutions are directly operational for the study of more complex transdisciplinary systems including other domains of physics besides synthetic biology.

4. From handmade design to the next step

The historic of synthetic biology showed that the first designed circuits contained a low number of genes (two genes for the toggle switch (Gardner, Cantor, and Collins 2000) and three genes plus a reporter gene for the repressilator (Elowitz and Leibler 2000)). These systems were designed by hand.

Often, hand-designed systems require a back and forth between the bench and the desk where scientists plan their systems. Systems biology provided mathematical models describing biological phenomenon (some of them are described in Chapter 2), enabling the simulation of basic reactions with computer tools. With a growing size and complexity of the designed systems, it has become necessary to use computer tools to simulate the designed systems. Adapted computer tools are indeed key to the further developing of complex systems in synthetic biology. Not only do we need appropriate simulation tools but also design tools, to go from a trial-error approach to a predictive approach, reducing the time spent at the bench. Existing computer-aided design tools are discussed in the Chapter 3 of this manuscript. Beforehand, the key concepts of biology and electronics that are used along this manuscript is described in the next chapter.

5. References

Anderson, J Christopher, Christopher A Voigt, and Adam P Arkin. 2007. "Environmental Signal Integration by a Modular AND Gate." *Molecular Systems Biology* 3. European Molecular Biology Organization: 133. doi:10.1038/msb4100173.

Aro, Eva-Mari. 2016. "From First Generation Biofuels to Advanced Solar Biofuels." *Ambio* 45 Suppl 1 (Suppl 1). Springer: S24-31. doi:10.1007/s13280-015-0730-0.

Bacchus, William, and Martin Fussenegger. 2013. "Engineering of Synthetic Intercellular Communication Systems." *Metabolic Engineering* 16 (March). Elsevier: 33–41. doi:10.1016/j.ymben.2012.12.001.

Basu, Subhayu, Yoram Gerchman, CH Collins, FH Arnold, and R Weiss. 2005. "A Synthetic Multicellular System for Programmed Pattern Formation." *Nature* 434 (April).

Bornholt, James, Randolph Lopez, Douglas M. Carmean, Luis Ceze, Georg Seelig, Karin Strauss, James Bornholt, et al. 2016. "A DNA-Based Archival Storage System." *ACM SIGOPS Operating Systems Review* 50 (2). ACM: 637–49. doi:10.1145/2954680.2872397.

Cohen, Stanley N, Annie C Y Chang, Herbert W Boyert, and Robert B Hellingt. 1973. "Construction of Biologically Functional Bacterial Plasmids In Vitro (R Factor/restriction Enzyme/transformation/endonuclease/antibiotic Resistance)" 70 (11): 3240–44.

Elowitz, Michael B, and S Leibler. 2000. "A Synthetic Oscillatory Network of Transcriptional Regulators." *Nature* 403 (6767): 335–38. doi:10.1038/35002125.

Endy, Drew. 2005. "Foundations for Engineering Biology." *Nature* 438 (7067): 449–53. doi:10.1038/nature04342.

Engler, Carola, Romy Kandzia, and Sylvestre Marillonnet. 2008. "A One Pot, One Step, Precision Cloning Method with High Throughput Capability." *PLOS ONE* 3 (11). Public Library of Science: e3647. https://doi.org/10.1371/journal.pone.0003647.

Fahnestock, M, A J Johnston, P Ross, and R Y Tsien. 1991. Dna sequencing, issued 1991.

Gaisser, Sibylle, Thomas Reiss, Astrid Lunkes, Kristian M Müller, and Hubert Bernauer. 2009. "Making the Most of Synthetic Biology. Strategies for Synthetic Biology Development in Europe." *EMBO Reports* 10 Suppl 1 (August): S5-8. doi:10.1038/embor.2009.118.

Gardner, T S, C R Cantor, and James J Collins. 2000. "Construction of a Genetic Toggle Switch in Escherichia Coli." *Nature* 403 (6767): 339–42. doi:10.1038/35002131.

Hattori, Masahira. 2005. "Finishing the Euchromatic Sequence of the Human Genome." *Tanpakushitsu Kakusan Koso. Protein, Nucleic Acid, Enzyme* 50 (2): 162–68. doi:10.1038/nature03001.

Hoops, Stefan, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. 2006. "COPASI--a COmplex PAthway SImulator." *Bioinformatics*

(Oxford, England) 22 (24). Oxford University Press: 3067–74. doi:10.1093/bioinformatics/btl485.

Huang, Hsin-Ho, and Peter Lindblad. 2013. "Wide-Dynamic-Range Promoters Engineered for Cyanobacteria." *Journal of Biological Engineering* 7 (1): 10. doi:10.1186/1754-1611-7-10.

Hucka, M., A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, et al. 2003. "The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models." *Bioinformatics* 19 (4): 524–31. doi:10.1093/bioinformatics/btg015.

Immethun, Cheryl M., Kenneth M. Ng, Drew M. Delorenzo, Ben Waldron-Feinstein, Ying Chiang Lee, and Tae Seok Moon. 2016. "Oxygen-Responsive Genetic Circuits Constructed in Synechocystis Sp. PCC 6803." *Biotechnology and Bioengineering* 113 (2): 433–42. doi:10.1002/bit.25722.

Jacob, F, and J Monod. 1961. "Genetic Regulatory Mechanisms in the Synthesis of Proteins." *Journal of Molecular Biology* 3 (3): 318–56. doi:10.1016/S0022-2836(61)80072-7.

Kemmer, Christian, David Andreas Fluri, Ulrich Witschi, Alain Passeraub, Andreas Gutzwiller, and Martin Fussenegger. 2011. "A Designer Network Coordinating Bovine Artificial Insemination by Ovulation-Triggered Release of Implanted Sperms." *Journal of Controlled Release* 150 (1): 23–29. doi:10.1016/j.jconrel.2010.11.016.

Kobayashi, Hideki, Mads Kærn, Michihiro Araki, Kristy Chung, Timothy S Gardner, Charles R Cantor, and James J Collins. 2004. "Programmable Cells : Interfacing Natural and Engineered Gene Networks."

Laity, J H, B M Lee, and P E Wright. 2001. "Zinc Finger Proteins: New Insights into Structural and Functional Diversity." *Current Opinion in Structural Biology* 11 (1): 39–46.

Lander, E S, L M Linton, Bruce W Birren, C Nusbaum, M C Zody, J Baldwin, K Devon, et al. 2001. "Initial Sequencing and Analysis of the Human Genome." *Nature* 409 (6822): 860–921. doi:10.1038/35057062.

Le Duc, Stéphane. 1910. *Théorie Physico-Chimique de La Vie et Générations Spontanées.* Paris: A. Poinat. doi:10.5962/bhl.title.32591.

Mardis, Elaine R. 2008. "The Impact of next-Generation Sequencing Technology on Genetics." *Trends in Genetics* 24 (3): 133–41. doi:10.1016/j.tig.2007.12.007.

Mike Brunker. 2016. "Microsoft and University of Washington Researchers Set Record for DNA Storage - Next at Microsoft."

Mikheyev, Alexander S., and Mandy M. Y. Tin. 2014. "A First Look at the Oxford Nanopore MinION Sequencer." *Molecular Ecology Resources* 14 (6): 1097–1102. doi:10.1111/1755-0998.12324.

Miller, Jeffrey C, Siyuan Tan, Guijuan Qiao, Kyle A Barlow, Jianbin Wang, Danny F Xia, Xiangdong Meng, et al. 2011. "A TALE Nuclease Architecture for Efficient Genome Editing." *Nat Biotech* 29 (2). Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.: 143–48. http://dx.doi.org/10.1038/nbt.1755.

Moore, Gordon E. 1965. "Cramming More Components onto Integrated Circuits." *Electronics*, 114–17.

Mullis, K, F Faloona, S Scharf, R Saiki, G Horn, and H Erlich. 1986. "Specific Enzymatic Amplification of DNA in Vitro: The Polymerase Chain Reaction." *Cold Spring Harbor Symposia on Quantitative Biology* 51 Pt 1. Cold Spring Harbor Laboratory Press: 263–73. doi:10.1101/SQB.1986.051.01.032.

Paine, Jacqueline A, Catherine A Shipton, Sunandha Chaggar, Rhian M Howells, Mike J Kennedy, Gareth Vernon, Susan Y Wright, et al. 2005. "Improving the Nutritional Value of Golden Rice through Increased pro-Vitamin A Content." *Nat Biotech* 23 (4). Nature Publishing Group: 482–87.

Qian, Lulu, Erik Winfree, and Jehoshua Bruck. 2011. "Neural Network Computation with DNA Strand Displacement Cascades." *Nature* 475 (7356). Nature Publishing Group: 368–72. doi:10.1038/nature10262.

Raimbault, Benjamin, Jean-Philippe Cointet, and Pierre-Benot Joly. 2016. "Mapping the Emergence of Synthetic Biology." *PLoS ONE* 11 (9): e0161522. doi:http://dx.doi.org/10.1371/journal.pone.0161522%23sec016.

"Registry of Standard Biological Parts." 2015. Accessed November 6. http://parts.igem.org/Main_Page.

Richardson, Sarah M, Leslie A Mitchell, Giovanni Stracquadanio, Kun Yang, Jessica S Dymond, James E DiCarlo, Dongwon Lee, et al. 2017. "Design of a Synthetic Yeast Genome." *Science (New York, N.Y.)* 355 (6329). American Association for the Advancement of Science: 1040–44. doi:10.1126/science.aaf4557.

Ro, Dae-Kyun, Eric M Paradise, Mario Ouellet, Karl J Fisher, Karyn L Newman, John M Ndungu, Kimberly A Ho, et al. 2006. "Production of the Antimalarial Drug Precursor Artemisinic Acid in Engineered Yeast." *Nature* 440 (7086). Nature Publishing Group: 940–43.

Ronaghi, Mostafa, Samer Karamohamed, Bertil Pettersson, Mathias Uhlén, and Pål Nyrén. 1996. "Real-Time DNA Sequencing Using Detection of Pyrophosphate Release." *Analytical Biochemistry* 242 (1). Academic Press: 84–89. doi:10.1006/abio.1996.0432.

Rothberg, Jonathan M., Wolfgang Hinz, Todd M. Rearick, Jonathan Schultz, William Mileski, Mel Davey, John H. Leamon, et al. 2011. "An Integrated Semiconductor Device Enabling Non-Optical Genome Sequencing." *Nature* 475 (7356). Nature Research: 348. doi:10.1038/nature10242.

Ruffing, Anne M, and Howland D T Jones. 2012. "Physiological Effects of Free Fatty Acid Production in Genetically Engineered Synechococcus Elongatus PCC 7942." *Biotechnology and Bioengineering* 109 (9). NIH Public Access: 2190–99. doi:10.1002/bit.24509.

Sander, Jeffry D, and J Keith Joung. 2014. "CRISPR-Cas Systems for Editing, Regulating and Targeting Genomes." *Nat Biotech* 32 (4). Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.: 347–55. http://dx.doi.org/10.1038/nbt.2842.

Tamsir, Alvin, Jeffrey J Tabor, and Christopher A Voigt. 2011. "Robust Multicellular Computing Using Genetically Encoded NOR Gates and Chemical 'Wires'." *Nature* 469 (7329): 212–15. doi:10.1038/nature09565.

Urnov, Fyodor D, Edward J Rebar, Michael C Holmes, H Steve Zhang, and Philip D Gregory. 2010. "Genome Editing with Engineered Zinc Finger Nucleases." *Nature Reviews. Genetics* 11 (9): 636–46. doi:10.1038/nrg2842.

Venter, J C, M D Adams, Eugene W Myers, P W Li, Mural RJ, Granger G Sutton, Hamilton O Smith, et al. 2001. "The Sequence of the Human Genome." *Science* 291 (5507): 1304–51. doi:10.1126/science.1058040.

Wang, Baojun, Mauricio Barahona, and Martin Buck. 2013. "A Modular Cell-Based Biosensor Using Engineered Genetic Logic Circuits to Detect and Integrate Multiple Environmental Signals." *Biosensors & Bioelectronics* 40 (1). Elsevier: 368–76. doi:10.1016/j.bios.2012.08.011.

Wendt, Kristen E., Justin Ungerer, Ryan E. Cobb, Huimin Zhao, and Himadri B. Pakrasi. 2016. "CRISPR/Cas9 Mediated Targeted Mutagenesis of the Fast Growing Cyanobacterium Synechococcus Elongatus UTEX 2973." *Microbial Cell Factories* 15 (1). BioMed Central: 115. doi:10.1186/s12934-016-0514-7.

"What Is Synthetic Biology? - Synthetic Biology Project." 2017. Accessed October 6. http://www.synbioproject.org/topics/synbio101/definition/.

Xie, Zhen, Liliana Wroblewska, Laura Prochazka, Ron Weiss, and Yaakov Benenson. 2011. "Multi-Input RNAi-Based Logic Circuit for Identification of Specific Cancer Cells." *Science (New York, N.Y.)* 333 (6047). American Association for the Advancement of Science: 1307–11. doi:10.1126/science.1205527.

Yoder, Nicholas C., and Krishna Kumar. 2002. "Fluorinated Amino Acids in Protein Design and Engineering." *Chemical Society Reviews* 31 (6): 335–41. doi:10.1039/b201097f.

Zhang, Ruihua, and Warren C. Ruder. 2015. "A New Environmental Biosensor for Cell Free Synthetic Biological Systems." *Biophysical Journal* 108 (2). Cell Press: 481a. doi:10.1016/j.bpj.2014.11.2630.

Chapter 2
Theoretical Background

Before exposing the findings of this thesis, this chapter presents useful biological and electronical knowledge for the rest of this reading. Let us start with some basic electronic concepts.

1. Basic electronic concepts

1.1 The main electronic devices

We present here the main electronic components used in a model described later in this thesis. In electronic circuits, the relation between two main variables serves to define the behavior of a component: the electric potential at a point or difference of potential between two points (also called voltage) and the electric current. The potential at point i corresponds to the amount of electron at this point in the circuit and is often expressed as V_i. The difference of potential between point A and point

B is named the voltage between A and B and is defined as follows: $U_{AB} = V_A - V_B$. A potential is expressed in Volts V. The electric current I is defined on a section as the quantity of electrons flowing through this section per s. A current is expressed in Amperes A.

In the following, we assume that the components are linked by perfect wires, so that the voltage between two points that are not separated by a component is null. Moreover, we assume that the ground is at a potential of 0. Indeed, the ground is a reference point used in electrical circuits.

1.1.1 Resistor

As its name suggests, a resistor resists the flow of electron passing through it. It acts as an obstacle. A resistor is a two-terminal passive device that simply reduces the current flow of a circuit: the energy loss is translated by an emission of heat by the component. The current I_R flowing through a resistor and the voltage U between its terminals are bound by Ohm's law (Ohm 1827):

$$I_R = \frac{U}{R} = \frac{V_1 - V_2}{R}$$

With V_1 and V_2 the potential at its terminals and R a constant modeling the resistance of the component. A resistor is often represented by a rectangle (Fig. 1) or by a see-saw line. By convention, the voltage U is represented as an arrow opposite to the current. As the resistance is a positive constant, if the current I_R is positive we can deduce that V_2 is inferior to V_1, confirming the role of obstacle of a resistor.

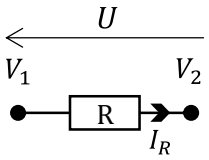


Fig. 1 Schematic representation of a resistor and its current, voltage and resistance.

1.1.2 Capacitor

Capacitors are passive devices that store electrons. They can also release them. They act as electrical potential energy storage. They have two terminals. We define the capacitance of the device as the electrical charge it carries; the capacitance is often noted C but in this thesis, we term it K because C will refer to the concentration of a soluble chemical species. The classical model of a capacitor binds the current I_K flowing through it to the time derivate of the voltage U between its terminals:

$$I_K = K \cdot \frac{dU}{dt}$$

If the capacitor is connected to the ground, we have the following:

$$I_K = -K \cdot \frac{dV}{dt}$$

with V the potential at the terminal that is not grounded (see Fig. 2). A capacitance is often represented by two thick bars of the same size, perpendicular to the wire (Fig. 2). The rake-like symbol is a common representation of the ground.

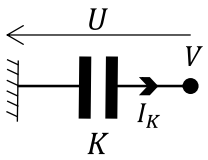


Fig. 2 A capacitor of capacitance K, traversed by a current I_K

1.1.3 Transistor

Transistors are active devices that are used to generate or process electrical signals in a circuit. Metal Oxide Semiconductor Field Effect Transistors (MOS-FET) have the following 3 terminals: the drain, the source and the gate. The current flowing through the transistor is controlled by the voltage between the gate and the source. In Fig. 3, the source is connected to the ground so that the voltage V controls the current I exiting the drain of the transistor. Thus, it can be modeled by a Voltage Controlled Current Source (VCCS).

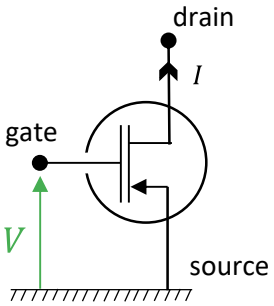


Fig. 3 A MOS-FET with its 3 pins acting like a Voltage Controlled Current Source (VCCS). V is the voltage controlling the current I flowing through the component.

1.2 Kirchhoff's laws

Kirchhoff's laws constitute a set of two equalities that govern electrical circuits. Kirchhoff's Current Law (KCL) states that the algebraic sum of the electrical currents meeting at a node is zero. Kirchhoff's Voltage Law (KVL) states that the algebraic sum of the electrical voltages along a closed loop is also equal to zero.

KCL and KVL can be combined in order to find the relationship between currents and voltages in a circuit. To illustrate this point, let us consider the circuit of Fig. 4. We would like to compute the voltage V_{BD} as a function of the voltages V_{AD} and V_{CD} generated by external sources. First, we can apply the KVL on the left loop (incl. A, B, D and the resistors R1 and R2):

$$V_{AB} + V_{BD} + V_{DA} = 0$$

We do the same on the right loop (incl. B, C, D and the resistors R2 and R3):

$$V_{BD} + V_{DC} + V_{CB} = 0$$

Using Ohm's laws, these two equations can be rewritten:

$$i_{R1} \cdot R_1 + i_{R3} \cdot R_3 - V_{AD} = 0$$

$$i_{R3} \cdot R_3 - V_{CD} + i_{R2} \cdot R_2 = 0$$

Then, we can write the KCL at the nodes B:

$$i_{R1} + i_{R2} - i_{R3} = 0$$

The three last equations form a set of linear equations that can be solved in order to find the three unknown current i_{R1}, i_{R2} and i_{R3}. For i_{R3} we obtain:

$$i_3 = \frac{R_1 \cdot V_{CD} + R_2 \cdot V_{AD}}{R_1 \cdot R_2 + R_1 \cdot R_3 + R_2 \cdot R_3}$$

Finally,

$$V_{BD} = R_3 \cdot \frac{R_1 \cdot V_{CD} + R_2 \cdot V_{AD}}{R_1 \cdot R_2 + R_1 \cdot R_3 + R_2 \cdot R_3}$$

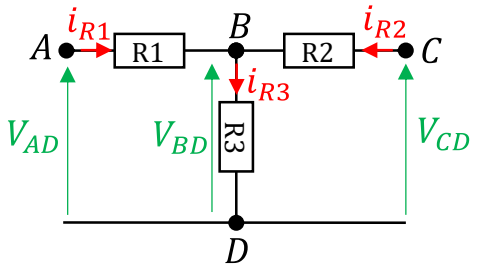


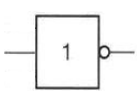
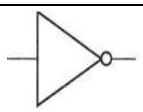
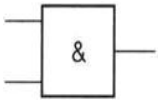
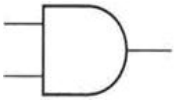
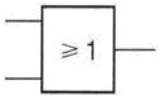
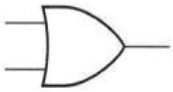
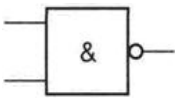
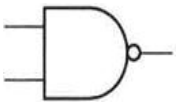
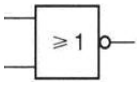
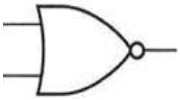
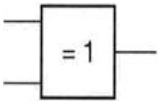
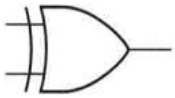
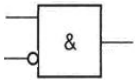
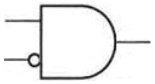
Fig. 4 Electronic circuit used to illustrate the Kirchhoff's laws.

By analogy, Kirchhoff's laws can be generalized to any conservative domain, such as mechanics or thermal physics or even biology. This last point is discussed in more details in the last section of this chapter.

1.3 Digital electronics vs analog electronics

Analog electronics are electronic systems that generate or process continuously variable signals (voltages or currents) using some basic passive (resistor, capacitor, inductor) or active (transistor, amplifier) devices. By opposition, digital electronics consist in circuits for which signals can only have two values: *true* (or '1') or *false* (or '0'). In this case, signals are processed by logic gates that performs Boolean operation on signals. The three main Boolean operators are the inverter (NOT), the AND operator for which output is *true* only if both inputs are *true* and the OR operator for which output is *true* if at least one input is *true.* These operators are summarized on Table 1. Different symbols exists to write these operators in equations. In this manuscript, we will use a dot (·) for an AND, a plus (+) for an OR and a bar before or over the expression ($\bar{}$) for a NOT. They form a universal set of Boolean operators (USBO), which means that any Boolean function can be realized with a combination of these operators.

Table 1 The main logic operators and their representation.

Operator	Mathematical symbol	Symbol of the gate		Logical propoition
		European	US	*Output is true if …*
NOT	$\bar{A}$			… input is *false*
AND	$A \cdot B$			… both inputs are *true*
OR	$A + B$			… at least one input is *true*
NAND	$\overline{A \cdot B}$ $A \uparrow B$			… at least one input is *false* (see de Morgan's theorem)
NOR	$\overline{A + B}$ $A \downarrow B$			… both inputs is *false* (see de Morgan's theorem)
XOR	$A \oplus B$			… one and only one input is *true*
INH	A/B			… the first input is *true* and the second is *false*

Several other operators can be defined from this USBO. The most common are the XOR, the NAND and the NOR operator. Noteworthy, the NOR in itself also constitutes a USBO. The INH operator is less common in digital electronics but we mention it here because we will reuse it later in this manuscript. In addition to these operators, a large amount of properties and theorems can be used to manipulate Boolean functions. Most of them are straightforward because they stem from a simple logical reasoning. The de Morgan's theorem (also used later in this chapter), is an example. It states that "*A is false and B is false"* is equivalent to "(*A or B) is false"* and that "*A is false or B is false"* is equivalent to "(*A and B) is false":*

$$\bar{A} \cdot \bar{B} = \overline{A + B} \quad \text{and} \quad \bar{A} + \bar{B} = \overline{A \cdot B}$$

Most of the time, the behavior of a digital circuits is described by a Boolean function. As inputs can only have two states, the number of possible input combinations is finite. Thus, a digital circuit can also be described exhaustively by a table that gathers all the possible combinations. This table is named a truth table (Table 2).

In practice, digital electronics is an abstraction of analog electronics. A digital signal can be a voltage (or a current) and the '0' and '1' corresponds to the fact that this voltage is above or below a given threshold. Moreover, digital gates are transistor-based circuits which function can be computed in an analog transfer function ($v_{out} = f(v_{in})$) that is so discriminating that it can be abstracted by a truth table.

Table 2 Truth table of different logic operators.

A	B	$A \cdot B$	$A + B$	$\overline{A \cdot B}$	$\overline{A + B}$	$A \oplus B$	A/B
0	**0**	0	0	1	1	0	0
0	**1**	0	1	1	0	1	0
1	**0**	0	1	1	0	1	1
1	**1**	1	1	0	0	0	0

2. The classical dogma of biology and its limitations

After these fundamental reminders of electronics, let us deal with the central dogma of biology.

2.1 Transcription and translation

The central dogma of molecular biology refers to how information is coded and exploited (read and used) in living organisms. This dogma states that the genetic information is coded with deoxyribonucleic acid (DNA); this code is named the genetic code and is composed of four different nucleotides (cytosine (C), guanine (G), adenine (A), or thymine (T)) as seen on Fig. 5. DNA is a double-stranded molecule: each cytosine is paired with a guanine, and each adenine is paired with a thymine base.

The cell is able to read the DNA-encoded information and create a cognate ribonucleic acid (RNA) molecule by a process called transcription. This molecule is single-stranded and composed of three similar bases as DNA, namely cytosine, guanine and adenine, whereas the thymine is replaced by uracil (U). With the replacement of T by U, the sequence of RNA is the same as the coding strand of its DNA (see Fig. 5).

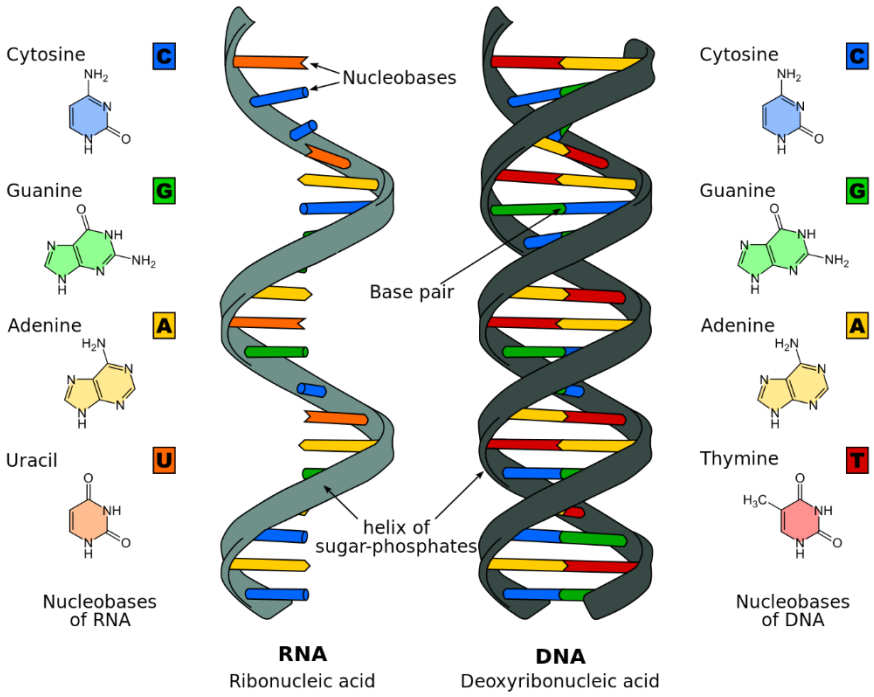


Fig. 5 DNA and RNA bases (Roland1952 2010). DNA bases pair according to the so-called Watson-Crick base-pairing: Adenine binds Thymine (two hydrogen bounds) and Cytosine binds Guanine (three hydrogen bounds).

The RNA strand is processed into a messenger RNA (mRNA) and then, in a step called translation, the cell reads the mRNA strand and produces necklace of aminoacids that fold into a protein (see Fig. 6).

Three RNA bases form a codon. Each codon codes for an aminoacid according to the genetic code. As there are 22 aminoacids and one stop codon to code with 4 different bases, this code is redundant.

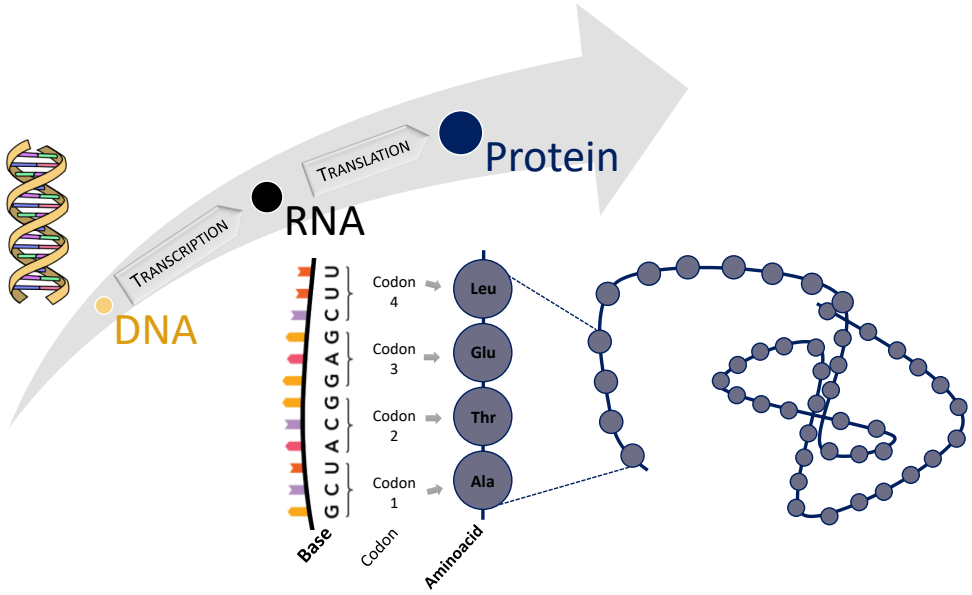


Fig. 6 The central dogma of biology

These processes require the cell machinery, namely enzymes and ribozymes. An enzyme is a protein that catalyzes a reaction, like the RNA-polymerase which is responsible for the transcription. A ribozyme is a biomolecule composed of RNA with catalytic properties, such as the ribosome, responsible for the translation (the ribosome is a complex made from protein and RNA, but the RNA is the catalytic actor).

This processes are similar across all the kingdoms of life, with the main point being that the genetic code is shared by all living species. However, some distinctions are to be made. Eukaryotic cells, such as mammalian cells, are cells that have a nucleus, an organelle holding genomic DNA. Prokaryotic cells are cells that do not have a nucleus, such as bacteria. In prokaryotic cells, both transcription and translation occur in the same compartment, the cellular compartment named the cytoplasmic compartment. In eukaryotic cells, transcription happens in the nucleus, then the strand of RNA travels out of the nucleus to be translated by a ribosome.

2.2 Limitations

2.2.1 Other functional RNAs

Most of the DNA does not code for a protein. Indeed, some sequences code for functional RNAs like ribosomal RNA, transfer RNA (involved in translation), micro RNA.

Reverse transcription

Biological information is not always initially coded as DNA. Some viruses code genetic information on RNA that has to be reverse transcripted into DNA and sometimes integrated into the host's genome. This is the case of the human immunodeficiency virus (HIV).

Post-translational modifications

Protein post-translational modifications correspond to the covalent addition of functional groups or proteins, proteolytic cleavage of regulatory subunits, or degradation of entire proteins, often executed by enzymes. They occur during or after protein biosynthesis and can alter the protein's function,

affecting almost all aspects of the cell biology. In fact, a given gene does not correspond exactly to a given molecule: taking into account also alternative splicing (see (Black 2003) for more details), a single gene could potentially give rise to 100 different proteins. These modifications therefore increase the proteome (set of expressed proteins of an organism) complexity. Common post-translational modifications include phosphorylation, acetylation, methylation and ubiquitination (the covalent linkage to the protein ubiquitin).

The maturation of pre-proinsulin into insulin is a rich example of post-translational modifications, like cleavage (see Fig. 7).

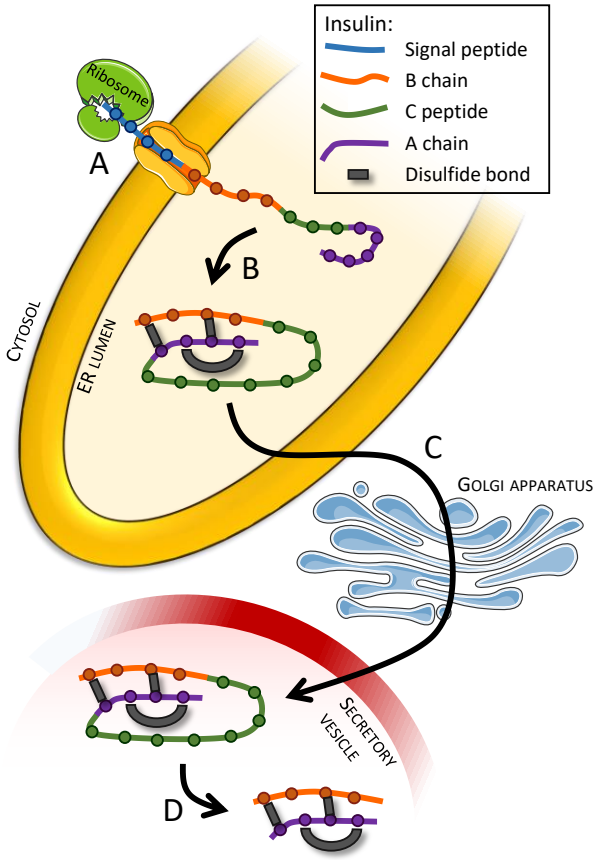


Fig. 7 Insulin maturation. A: Pre-proinsulin is translated with the help of a ribosome. The polypeptidic chain is translocated into the endoplasmic reticulum (ER). B: in the ER, the signal peptide (blue) is cleaved giving rise to proinsulin. Proinsulin fold in the ER and 3 disulfide bonds form between the different peptide chains of the protein: two between the B chain and the A chain and one between 2 amino acids from the A chain. C: proinsulin is exported from the ER to the Golgi apparatus and then packed into a secretory vesicle. D: proinsulin becomes mature insulin through the cleavage of the C peptide. Upper case names correspond to cell compartments (they are not to scale in this representation).

Post-translational modifications are used in synthetic biology (Park, Zarrinpar, and Lim 2003).

2.2.2 Epigenetics

Epigenetics traits are stable heritable traits resulting from changes in a chromosome without alterations in the DNA sequence (Berger et al. 2009). Epigenetic traits can be triggered by the environment, like the temperature for the plants. Epigenetic marks on the genome include covalent modifications of the DNA (e.g. DNA methylation) or of the histone proteins (such as several post-translational modifications mentioned above).

Prions are another example of epigenetics. Prions are infectious protein that are able to pass from a soluble state to an aggregate state (Koonin 2012). In this latter state, they can induce the aggregation of the native version of the same protein. Hence they are capable of these changes without changing the genome. This violate the central dogma by allowing "information flow from proteins to the genome" (Koonin 2012).

3. Genetic regulatory networks

Of particular focus for us are the Genetic Regulatory Networks (GRN). Genetic regulation is performed naturally by cells, as it was discovered by Monod and Jacob (Jacob and Monod 1961). A gene is a DNA sequence containing the information to create a protein as seen previously. This sequence is preceded by a promoter sequence, onto which can bind different actors of gene expression. Other elements play an important role in the expression (i.e. the production) of a protein. These elements are also DNA-encoded, often before the coding sequence of the protein. They include the ribosome binding site (RBS), whose sequence affects the rate at which the ribosome reads the mRNA ((Shine and Dalgarno 1975) for bacteria and (Kozak 1981) for eukaryotes). Hereafter, we will only focus on the binding of transcription regulators.

3.1 Transcription factors

As a reminder, the first step in the production of a protein from DNA is transcription: the DNA molecule is "read" by an RNA-polymerase, which produces a cognate RNA molecule. In order for this step to happen, the RNA-polymerase needs to bind to the afore-mentioned promoter sequence. Other proteins, named regulators or transcription factors, can also bind this region, and modulate transcription. A regulator enhancing transcription is an activator whereas a regulator prohibiting it is an inhibitor (Fig. 8). Certain promoter sequences are constitutive: they do not need any activators to initiate transcription, i.e. they are always "on". In most cases, in synthetic biology's GRNs, when a promoter is only targeted by a repressor, it is a constitutive promoter. In the rest of this thesis, we assume this when not otherwise stated. Please note that this does not include promoters that are both repressed and activated, as they need an activator to be "on".

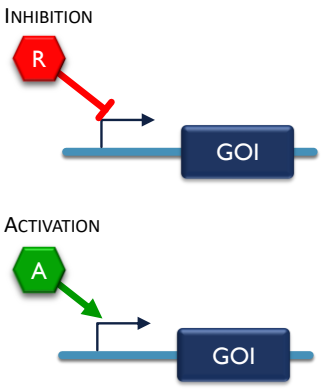


Fig. 8 Schematic representation of inhibition (upper part) or activation (lower part) of the transcription of a gene of interest (GOI) by a transcription factor.

Negative regulation often works by steric hindrance. Steric hindrance is the prohibition of a reaction (such as transcription) by a large molecule, obstructing the reaction by its large size. The transcription

factors that act this way bind to a DNA sequence which is located near the RNA-polymerase binding site. Hence the RNA-polymerase cannot bind or initiate transcription.

Another mechanism of inhibition is the deformation of DNA. This is illustrated by the inhibitor LacI. Target sequences of LacI are located upstream and downstream of the RNA-polymerase binding region (Fig. 9). LacI proteins bind to these sequences and form dimers. Upon dimer formation, the DNA strand is folded in a way that prevents its access by RNA-polymerase. Transcription is therefore halted.

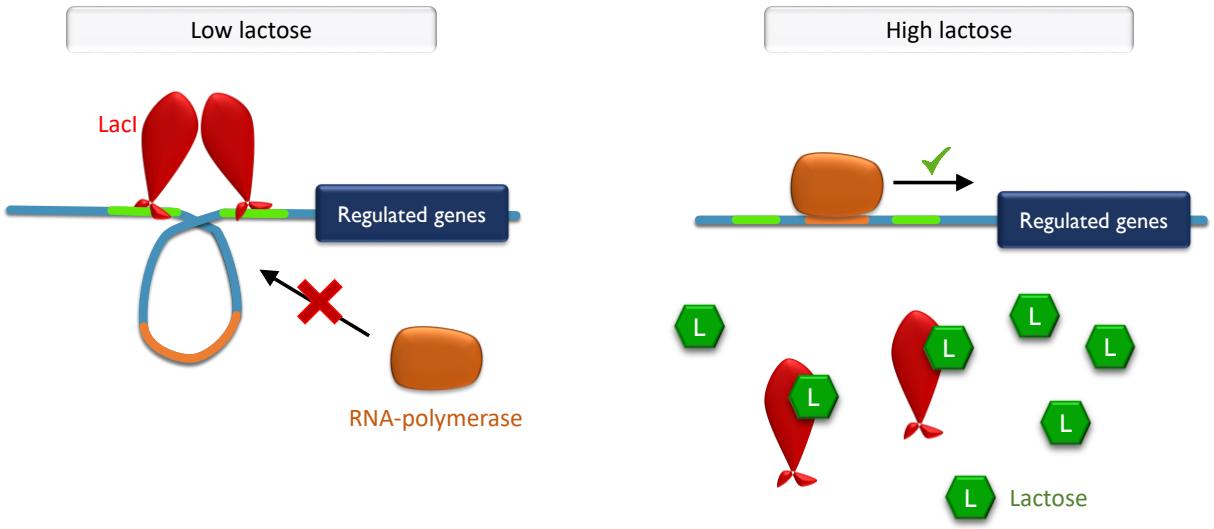


Fig. 9 LacI-regulated genes' induction by lactose. LacI is an inhibitory protein. Its binding to DNA provides the RNA-polymerase from binding and transcripting the downstream genes. Lactose relieves this inhibition, therefore acting as an activator.

Activators positively regulate transcription directly or indirectly. Direct effectors are composed of two domains: a DNA-binding domain targeting the transcription factor's regulating sequence on the gene's promoter and an activation domain recruiting the RNA polymerase or stabilize its binding (Ptashne and Gann 1997). For example, the activator protein Gal4 in yeast activates the genes responsible for the galactose metabolism (Lohr, Venkov, and Zlatanova 1995). Indirect regulation consists in the relieving of a negative regulation. For example, tetracycline activates the transcription of the gene TetA by binding to the inhibitor TetR, a protein that binds the TetA promoter region and inhibits its transcription (Ramos et al. 2005). Another similar example is the binding of lactose to LacI inhibitors (Fig. 9). LacI bound by lactose cannot repress the expression of downstream genes. Lactose therefore acts as an activator.

The produced proteins can be regulators that will in turn affect the expression of other genes (including their own gene). The produced proteins can also interact with other proteins or molecules, or produce molecules, which in turn affect gene expression.

3.2 Networks

Systems biology further demonstrated the organization of such interacting genes into systems, hence the notion of GRN. These systems are composed of small modules that interact together, giving rise to the final behavior of the whole system. Genes are « wired » to one another thanks to regulatory proteins (transcription factors). Typical circuits are composed of regulated genes coding for a regulating protein. Such networks exist in nature (Filloux 2012). However, synthetic biologists aim to

design their own networks to perform specific tasks (Moon et al. 2012; Bacchus and Fussenegger 2013).

3.3 Modeling

Here we present a classical ordinary differential equations model of GRNs. We write one equation per chemical species. The inputs of a system are often regulators and intervene in the equations as modifiers only: their concentration is considered to be externally controlled. In reality, these molecules are often small molecules, like arabinose or anhydrotetracycline, that bind to transcription regulators, like araC and TetR respectively. Upon binding, the behavior of the transcription regulator will change. In the two previous examples, araC and TetR are repressors when unbound and become activators in the presence of their cognate molecule. In such cases, the transcription regulators are often expressed continuously so that the small molecules act as activators.

In each of these equations, the time derivative of the concentration of X_i is simply equal to the sum of the rate of the N reactions that consume/produce the species. As no species is infinitely stable in biology, we further subtract a term of degradation of the first order (see Equation 1):

Equation 1

$$\frac{dX_i}{dt} = \sum_{j=1}^{N} v_{i,j} - d_i \cdot X_i$$

with $v_{i,j}$ the equation rate giving the production (if positive) or the consumption (if negative) of the species i by the reaction j. $v_{i,j}$ that may depend on other species of the system.

To model the dynamics of a regulated gene's expression, we first focus on the binding of a transcription factor to its DNA regulating sequence. Since the dynamics of transcription are often much slower than those of this binding (Alon 2006) we consider the binding reaction to be at its equilibrium and apply the classical law of mass action with the quasi-static states approximation. This gives rise to a Hill equation term.

We can now explicit the production of $mRNA_k$ from gene k, regulated by activators A_i and repressors R_i. In our case, $mRNA_k$ only has a synthesis term corresponding to its expression $Expr_k$ and no consumption term. We add the contribution Act_k and Rep_k of respectively each activators and each repressors of gene k:

$$Act_k = \sum_{i=1}^{N} \left(\frac{[A_i]}{K_{A,i,k}} \right)^{n_{A,i,k}}$$

$$Rep_k = \sum_{i=1}^{M} \left(\frac{[R_i]}{K_{R,i,k}} \right)^{n_{R,i,k}}$$

with n the Hill number and $K_{A,i,k}$ and $K_{R,i,k}$ the dissociation constant of respectively A_i and R_i to gene k. We also define the maximal transcription rate $k_{tr,k}$ of this process and the leakiness α_k of the promoter. The leakiness is a number comprised between 0 and 1 and models the leaky production of the unbound promoter. Indeed, without any activator, tight promoters (promoters that are not leaky) should not express any mRNA, as expected. On the other hand, leaky promoters produce a small

amount of mRNA even when they are not supposed to. With that we obtain the following expression term:

$$Expr_k = k_{tr,k} \cdot \left(\alpha_k + (1 - \alpha_k) \cdot \frac{Act_k}{1 + Act_k} \cdot \frac{1}{1 + Rep_k}\right)$$

And finally for $mRNA_k$ with its degradation term:

Equation 2

$$\frac{d[mRNA_k]}{dt} = Expr_k - d_{mRNA,k} \cdot [mRNA_k]$$

with $d_{mRNA,k}$ the degradation rate of $mRNA_k$.

The protein X_k translated from $mRNA_k$ also only features a synthesis term and a degradation term in this case:

$$\frac{d[X_k]}{dt} = k_{tl,k} \cdot [mRNA_k] - d_{X,k} \cdot [X_k]$$

with $d_{X,k}$ the degradation rate of X_k and $k_{tl,k}$ the translation rate.

Other types of regulation exist. It is possible to introduce an auxiliary protein that forms a complex with protein X_k, thereby adding both a term of production (proportional to k_{OFF} the dissociation rate of the binding reaction) and of consumption (proportional to k_{ON} the association rate of the binding reaction). It is also possible to introduce regulation at the level of mRNA. This is discussed in the following paragraph.

4. Micro-RNA

The regulation process described hereabove consists in modulating the transcription rate of a promoter with third-party molecules. Thus, this regulation is called transcriptional regulation, by opposition to translational regulation which consists in modifying the translation rate of an mRNA.

4.1 Micro-RNA silencing

Micro-RNAs or miR are non-coding eukaryotic single-stranded RNAs. They are generally 21 to 24 nucleotides long and bind to the 3' UTR of their target mRNA (located after the coding region of an mRNA). The target recognized by a miR is almost its complementary sequence (a few nucleotides may differ). MiR binding induces the degradation of the bound mRNA or inhibits its translation.

MiR are involved in many essential cellular processes like growth (Ambros et al. 2003) and cell differentiation (Ivey and Srivastava 2010), apoptosis (Baulcombe 2002), metabolism (Lee, Feinbaum, and Ambros 1993), etc.

MiR are also involved in several pathologies like cancer (Jansson and Lund 2012) and heart diseases (Latronico and Condorelli 2009). They are therefore a therapeutic target as well as biomarkers for cancer, as envisioned by (Xie et al. 2011). Indeed, each cancer has its own "micro-RNA pattern" and therefore its specific cure.

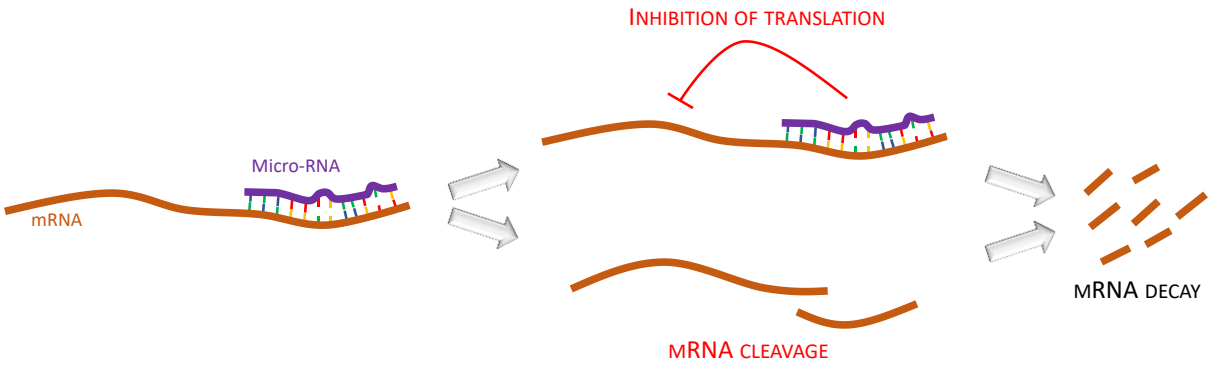


Fig. 10 miR silencing. The binding of a miR on a mRNA inbhibits its translation or induces its cleavage. The result is the same in both cases: mRNA decays.

4.2 Modeling micro-RNA

As seen above, miR can be used in synthetic biology to create networks able to identify the micro-RNA pattern of a cell. For a reminder, a miR forms a complex with an mRNA molecule, inhibiting its translation into a protein.

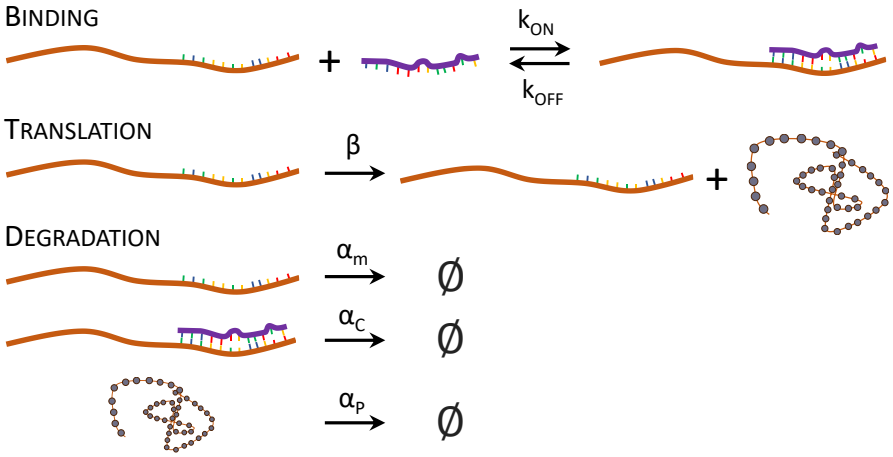


Fig. 11 Modeling the dynamics of micro-RNA silencing. The constants represent the reaction rates. In orange, the target mRNA, in purple the miR and in orange circles the protein coded by the mRNA.

To represent this phenomenon with ODEs, we model the dynamics of this binding. Let $[mRNA]$, $[miR]$, $[mRNA - miR]$ and $[P]$ be the concentrations of the mRNA, the miR, the mRNA and miR complex, and the protein respectively. The miR is an input in our system: we consider that its concentration is controlled externally.

We choose to model 5 reactions (see Fig. 11): the binding of the miR to the mRNA with an association rate of k_{ON} and the dissociation of the complex mRNA- miR at the rate of k_{OFF}, the translation of the mRNA (not consumed here) into a protein P at the rate beta and three degradations, one for the mRNA alone at the rate α_m, one for the complex mRNA-miR at the rate α_c and another one for the protein at the rate α_P. In this system, the miR is an input: therefore, we do not model its decay. We could model each process with more details (for example the degradation process can be divided into the dilution of the degraded species in the cell and its active degradation by the cellular machinery), but this level of details is sufficient for our needs.

Here are the equations representing the dynamics of mRNA, the complex mRNA-miR and the protein P:

Equation 3

$$\frac{d[mRNA]}{dt} = k_{OFF} \cdot [mRNA - miR] - k_{ON} \cdot [mRNA] \cdot [miR] - \alpha_m \cdot [mRNA]$$

Equation 4

$$\frac{d[mRNA - miR]}{dt} = k_{ON} \cdot [mRNA] \cdot [miR] - k_{OFF} \cdot [mRNA - miR] - \alpha_C \cdot [mRNA - miR]$$

Equation 5

$$\frac{d[P]}{dt} = \beta \cdot [mRNA] - \alpha_P \cdot [P]$$

Degradations are of the first order (Equation 3, Equation 4, Equation 5), as well as the production of the protein, dependent of the concentration of mRNA (Equation 5). The miR acts here as a repressor of the translation by pumping the mRNA and transforming it into an inactive complex, the mRNA-miR complex.

An alternative modeling approach consists in applying again the quasi-static states approximation by considering the binding miR-mRNA much faster than the translation itself. Under this assumption, the derivative of the concentration of the complex miR-mRNA is equal to zero and we can write:

$$[mRNA - miR] = \frac{k_{ON}}{k_{OFF} + \alpha_C} \cdot [mRNA] \cdot [miR]$$

Then, we introduce in Equation 5 the quantity of unbound mRNA $[mRNA]_F$ instead of the total concentration of mRNA $[mRNA]$. The relationship between $[mRNA]_F$ and $[mRNA]$ is:

$$[mRNA]_F = [mRNA] - [mRNA - miR]$$

Combining the two last equations leads to

$$[mRNA]_F = [mRNA] \cdot \left(1 - \frac{k_{ON}}{k_{OFF} + \alpha_C} \cdot [miR]\right)$$

4.3 Micro-RNA genetic circuits

Using only negative regulation of the DNA and miR binding, it is possible to realize any Boolean equation. An example can be seen with the work of (Xie et al. 2011).

4.3.1 NOT gate and NOR gate

The sole action of a miR on a gene corresponds to a NOT gate (see the upper panel on Fig. 12): the signal of the input miR is inverted as the target gene is only expressed when the input is absent. Moreover, multiple miR binding sites can be inserted on a target mRNA. Since the presence of any of the miR would inhibit translation, this construct acts as a NOR gate (see the lower panel on Fig. 12): the miRs must all be absent for the target to be expressed.

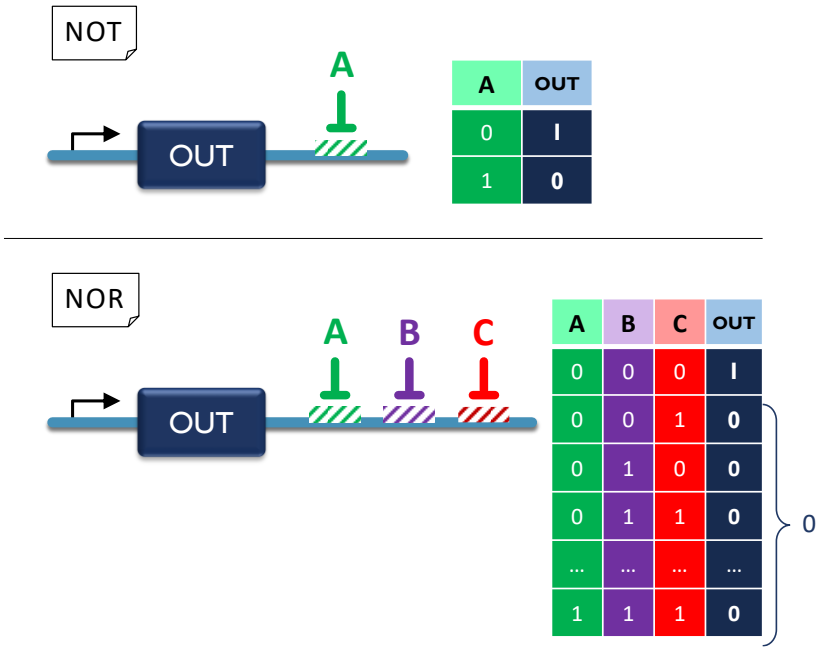


Fig. 12 MiR NOT and NOR gates. The promoters are constitutive (they always initiate transcription). The presence of only one miR suffices to inhibit translation. The upper panel shows a NOT gate: the input signal A is inverted, as seen on the logic gate. The lower panel shows a 3-inputs NOR gate: translation happens only when none of the three inputs A, B and C are present (all the other combinations of inputs results in translation inhibition). Note that this construct is generalizable to any number of inputs.

Noteworthy here is that a NOR gate with only one miR is a NOT gate.

4.3.2 Generalization of any Boolean equation

Using only repressors as intermediates, it is possible to express any Boolean equation with miR inputs. The first key observation is that a gene regulated by multiple repressors on the same promoter acts as a NOR, in a similar way than the miR NOR gate. The second key observation is that NOR gates are a universal set of operators: this means that any logic equation can be represented as NOR gates.

We will try to systematize the construction of a genetic circuit with miR inputs. Let A, B, C and D be miR. Let Y be a protein whose logical equation is the following:

$$Y = A \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot C \cdot \bar{D}$$

Every logical equation can be written under an OR-sum of AND terms (monomials). Let replace $A \cdot \bar{B} \cdot C$ by S_1 and $\bar{A} \cdot B \cdot C \cdot \bar{D}$ by S_2. The first operation consists in the double complement, which creates a NOT gate and a NOR gate:

$$Y = S_1 + S_2 = \overline{\overline{S_1 + S_2}} = \overline{S_1 \downarrow S_2} = \overline{S_F}$$

with $S_F = S_1 \downarrow S_2$. The first thing to implement is therefore a NOT gate that inverts signal S_F. In practice (see Fig. 13), we create a repressor protein RepF that represses the expression of Y. Signal S_F corresponds itself to a NOR gate of signal S_1, represented by the blue frame in Fig. 13, and signal S_2, represented by the green frame in Fig. 13. Both of these signals are coded by the same protein Rep1. This is not a problem as the only relevant combination is when both of these signals are absent (see above for details) and this reduces the number of components in the circuit.

The following step is to deal with signals S_1 and S_2. We will only detail S_1, as the same procedure can be applied for S_2. Let replace the AND gates by NOR gates:

$$S_1 = A \cdot \bar{B} \cdot C = \bar{A} \downarrow B \downarrow \bar{C}$$

Signals A and C need to be inverted. They are inverted by introducing a regulatory protein as intermediate. Therefore, we implement this NOR gate as a "mixed" NOR gate: both regulatory proteins and miR are used. As seen on Fig. 13, we create two separate genes both coding for the same regulator Rep0 (same explanation as above), one inhibited by A and the other by C. B can directly inhibit Rep1.

The same logic is followed for S_2.

In fact, this can be generalized for any number of monomial (Boolean equation with only AND and NOT operators). If the equation comprises n monomials, we create S_1, S_2, …, S_n in a similar fashion.

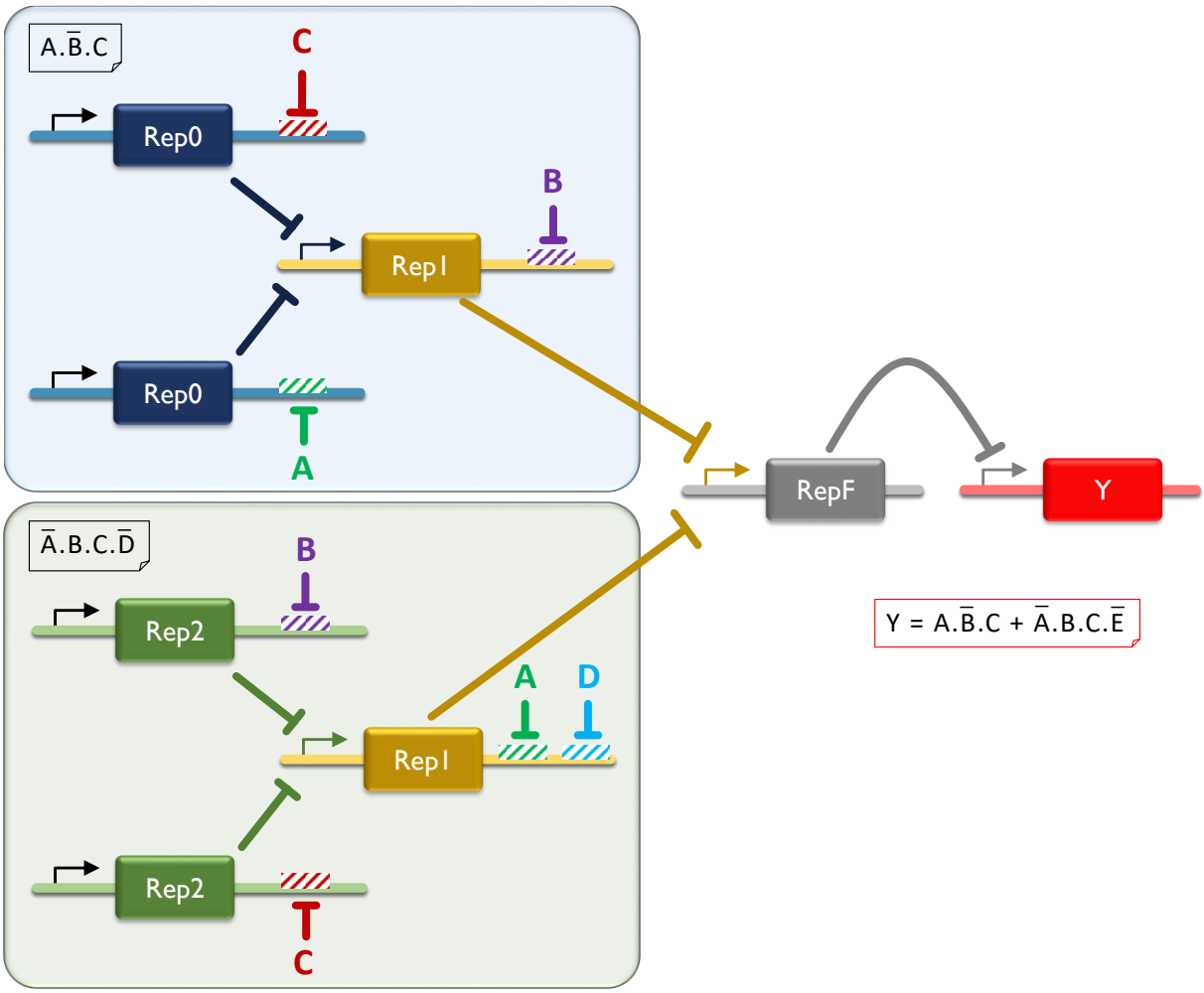


Fig. 13 Implementation of a miR logic equation. A, B, C and E are miR. Rep are inhibitor proteins. The equation to be implemented is shown in the red frame below the output gene Y. This equation is divided into its two monoms, in the blue frame and the green frame. Promoters are all constitutive (the protein is expressed in the absence of any regulatory protein).

5. Analogy between electronics and biology

In the first paragraph of this chapter, we introduced 3 electronic components. A resistor that is responsible of a first order loss of electron, a capacitor whose model involves a time derivate of its voltage and a transistor that can act as a source of electron controlled by an external voltage. With

these 3 components, it is possible to model a GRN. The concentration of a species corresponds to the voltage between the pins of a capacitor, its degradation is modeled by a resistor connected to the ground and any source of production or consumption can be materialized by a transistor connected to the voltage of another species (the regulator).

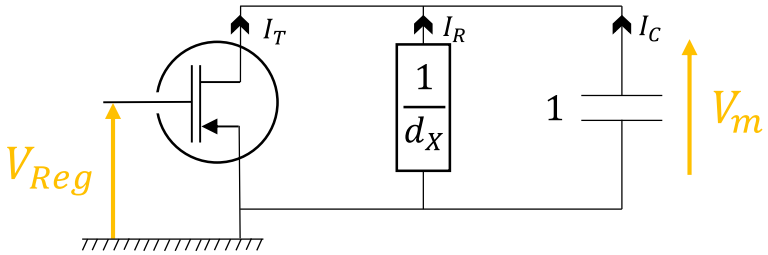


Fig. 14 Electronic equivalent of the transcription regulated by a regulator. The resistor has a resistance of $\frac{1}{d_X}$ and the capacitor has a capacitance of 1.

If we take the example of the transcription of mRNA (modeled by voltage V_m) regulated by one regulator (modeled by voltage V_{Reg}), we can draw the diagram shown on Fig. 14. As explained above, we have:

$$I_T = f(V_{Reg})$$

$$I_R = -d_X \cdot V_m$$

$$I_C = -\frac{dV_m}{dt}$$

with $f(V_{Reg})$ a function of the voltage V_{Reg} and d_X the degradation rate of the mRNA. Using the law of current conservation (Kirchhoff's law), we have the following:

$$I_T + I_R + I_C = 0$$

So that:

$$\frac{dV_m}{dt} = f(V_{Reg}) - d_X \cdot V_m$$

If we replace the function f by the appropriate function (Hill function times the transcription rate), we have indeed the ODE of transcription (Equation 2). Function f can also be replaced by a simple production term depending on the mRNA to find the equation of translation. With this paradigm, function f can be replaced by any desired function to obtain the corresponding dependency.

6. Conclusion

With this chapter, we give an overview of the main concepts addressed throughout this manuscript. This will ease the deciphering of the presented results for the unspecialized reader. We particularly focused on Boolean equations and associated abstraction, the main electronic components used in our models and the classic ODE models by which we simulate the behavior of biological systems.

7. References

Alon, Uri. 2006. *An Introduction to Systems Biology: Design Principles of Biological Circuits*.

Ambros, Victor, Bonnie Bartel, David P Bartel, Christopher B Burge, James C Carrington, Xuemei Chen, Gideon Dreyfuss, et al. 2003. "A Uniform System for microRNA Annotation." *RNA (New York, N.Y.)* 9 (3): 277–79.

Bacchus, William, and Martin Fussenegger. 2013. "Engineering of Synthetic Intercellular Communication Systems." *Metabolic Engineering* 16 (March). Elsevier: 33–41. doi:10.1016/j.ymben.2012.12.001.

Baulcombe, D. 2002. "DNA EVENTS: An RNA Microcosm." *Science* 297 (5589): 2002–3. doi:10.1126/science.1077906.

Berger, Shelley L, Tony Kouzarides, Ramin Shiekhattar, and Ali Shilatifard. 2009. "An Operational Definition of Epigenetics." *Genes & Development* 23 (7). Cold Spring Harbor Laboratory Press: 781–83. doi:10.1101/gad.1787609.

Black, Douglas L. 2003. "Mechanisms of Alternative Pre-Messenger RNA Splicing." *Annual Review of Biochemistry* 72 (1): 291–336. doi:10.1146/annurev.biochem.72.121801.161720.

Filloux, Alain. 2012. *Bacterial Regulatory Networks*. Caister Academic Press.

Ivey, Kathryn N., and Deepak Srivastava. 2010. "MicroRNAs as Regulators of Differentiation and Cell Fate Decisions." *Cell Stem Cell* 7 (1): 36–41. doi:10.1016/j.stem.2010.06.012.

Jacob, F, and J Monod. 1961. "Genetic Regulatory Mechanisms in the Synthesis of Proteins." *Journal of Molecular Biology* 3 (3): 318–56. doi:10.1016/S0022-2836(61)80072-7.

Jansson, Martin D., and Anders H. Lund. 2012. "MicroRNA and Cancer." *Molecular Oncology* 6 (6). No longer published by Elsevier: 590–610. doi:10.1016/j.molonc.2012.09.006.

Koonin, Eugene V. 2012. "Does the Central Dogma Still Stand?" *Biology Direct* 7 (August). BioMed Central: 27. doi:10.1186/1745-6150-7-27.

Kozak, M. 1981. "Possible Role of Flanking Nucleotides in Recognition of the AUG Initiator Codon by Eukaryotic Ribosomes." *Nucleic Acids Research* 9 (20): 5233–52.

Latronico, Michael V. G., and Gianluigi Condorelli. 2009. "MicroRNAs and Cardiac Pathology." *Nature Reviews Cardiology* 6 (6). Nature Publishing Group: 418–29. doi:10.1038/nrcardio.2009.56.

Lee, R C, R L Feinbaum, and V Ambros. 1993. "The C. Elegans Heterochronic Gene Lin-4 Encodes Small RNAs with Antisense Complementarity to Lin-14." *Cell* 75 (5): 843–54.

Lohr, D, P Venkov, and J Zlatanova. 1995. "Transcriptional Regulation in the Yeast GAL Gene Family: A Complex Genetic Network." *FASEB Journal : Official Publication of the Federation of American Societies for Experimental Biology* 9 (9): 777–87.

Moon, Tae Seok, Chunbo Lou, Alvin Tamsir, Brynne C Stanton, and Christopher A Voigt. 2012. "Genetic Programs Constructed from Layered Logic Gates in Single Cells." *Nature* 491 (7423): 249–53. doi:10.1038/nature11516.

Ohm, Georg Simon (1789-1854). Auteur du texte. 1827. "Die Galvanische Kette : Mathematisch Bearbeitet / von Dr. G. S. Ohm."

Park, S.-H., Ali Zarrinpar, and Wendell A Lim. 2003. "Rewiring MAP Kinase Pathways Using Alternative Scaffold Assembly Mechanisms." *Science* 299 (5609): 1061–64. doi:10.1126/science.1076979.

Ptashne, M, and A Gann. 1997. "Transcriptional Activation by Recruitment." *Nature* 386 (6625): 569–77. doi:10.1038/386569a0.

Ramos, J. L., M. Martinez-Bueno, A. J. Molina-Henares, W. Teran, K. Watanabe, X. Zhang, M. T. Gallegos, R. Brennan, and R. Tobes. 2005. "The TetR Family of Transcriptional Repressors." *Microbiology and Molecular Biology Reviews* 69 (2): 326–56. doi:10.1128/MMBR.69.2.326-356.2005.

Roland1952. 2010. “File:Difference DNA RNA-EN.svg - Wikimedia Commons.”

Shine, J, and L Dalgarno. 1975. “Determinant of Cistron Specificity in Bacterial Ribosomes.” *Nature* 254 (5495): 34–38.

Xie, Zhen, Liliana Wroblewska, Laura Prochazka, Ron Weiss, and Yaakov Benenson. 2011. “Multi-Input RNAi-Based Logic Circuit for Identification of Specific Cancer Cells.” *Science (New York, N.Y.)* 333 (6047). American Association for the Advancement of Science: 1307–11. doi:10.1126/science.1205527.

Chapter 3
Design Flow of Synthetic Biology

As mentioned in the previous chapter, the complexity of the designed systems in synthetic biology is limited both by technological bottlenecks and by the shortcomings of the design tools. While the design process is often hand-made, virtual prototyping is coming of age. However, most of these tools are *ad hoc* and lack genericity and reusability. Standardizing design and simulation tools as well as the whole process of creating a new biological function/system has now become a key for advancing towards more complicated systems. To this aim, inspiration from micro-electronics can be taken. Indeed, this field has had decades of experience in system design, and instead of redeveloping system design for synthetic biology, it can be time-saving to adapt tools and knowhow from micro-electronics.

To this end, we first present the different design approaches used in synthetic biology and in other domains of physics. A focus is put on microelectronics, since it is the domain from which we draw

inspiration. Then, we establish a state-of-the-art of existing tools for the two main parts of system-level design flow, i.e. virtual prototyping (modeling and simulation) and design automation.

1. The design flow

The design flow of a system (incl. electronics, mechanics, thermodynamics, software etc.) is a set of steps, methods and tools that are implemented to go from the specification of a system to its practical implementation. Two traditional approaches can be envisioned: bottom-up and top-down. For the purpose of this work, we will see what they consist in and how they relate to designing a biological system.

1.1. Bottom-up

The bottom-up approach consists in building systems by assembling standardized well-characterized components, usually sorted in a library, into sub-systems, themselves being assembled to give rise to the final system (see Fig. 1). This final system is then compared to the expected result.

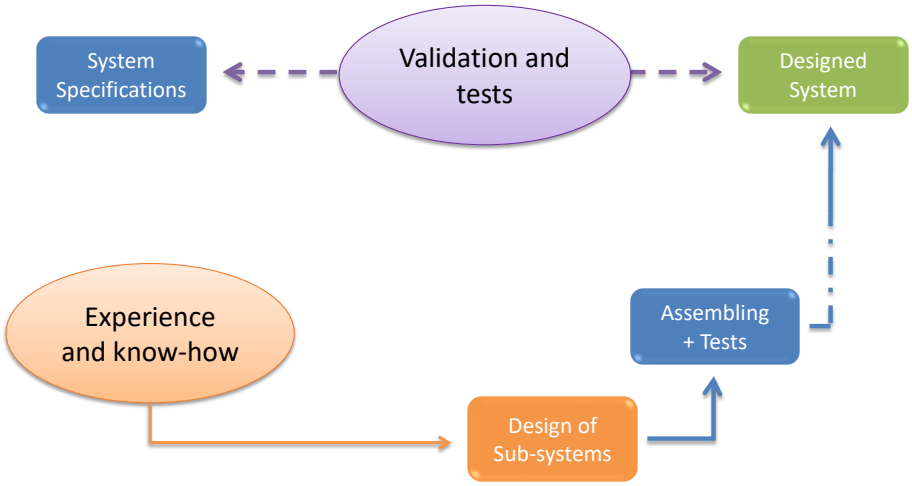


Fig. 1 Bottom-up approach

If this approach allows a higher control throughout the design process, notably by controlling the development of each sub-system, on the long term this can be expensive: with this approach, one lacks global understanding of the complete system to be handed, needing backtracking to correct errors in the sub-systems. Hence, the latest a mistake appears in the design process, the hardest it is to repair. Furthermore, this approach hinders multi-domain team work and complicates division of tasks.

1.2. Top-down

The top-down approach consists in decomposing hierarchically the expected systems into a set of sub-functions (see Fig. 2). At each step of the decomposition, the specification of each sub-function and the interfaces between them are established. The consistency of these sub-functions is checked by modeling and simulation.

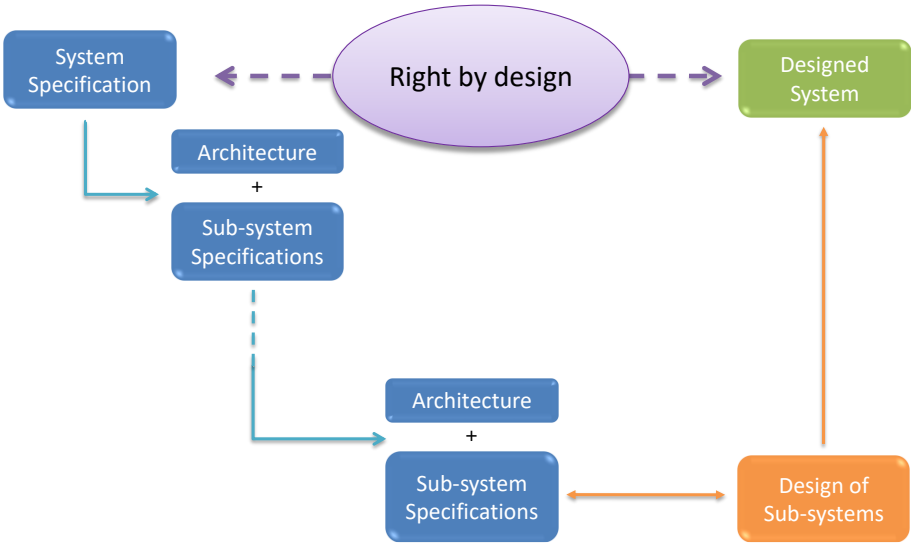


Fig. 2 Top-down approach

For each of the specified sub-functions, a sub-system is implemented by picking up standardized components from a library. In this hierarchic concept, each steps therefore consists in spreading the global specifications into sub-specification. This approach allows an understanding of all the sub-system composing the final system beforehand. Ideally, top-down approach ensures that if the hierarchical decomposition has been executed without mistakes and if each sub-system meets its specification, the assembly of the sub-systems is problem-free. The final system would be right by design.

Although this method is theoretically very efficient, applying it to an actual use case is very difficult. Indeed, such a comprehension requires predicting models of the behavior of each sub-system.

1.3. Meet-in-the-middle

Both aforementioned approaches have advantages and drawbacks. In practice, it is often very hazardous to design a system from scratch using a bottom-up approach without a global plan. Conversely, the top-down approach may be useful for the first stages of decomposition but can be troublesome and time-consuming when the number of sub-systems becomes large and/or the sub-systems are simple enough to be hand-designed. Thus, in the industry, a “meet-in-the-middle” approach is often used, benefiting both from a top-down approach for the earlier stages of the design and then from a bottom-up approach.

1.4. Common use of top-down and bottom-up in synthetic biology

“Top-down” and “bottom-up” are terms that are often met in the context of synthetic biology. They are related to the way a new artificial organism is built rather that the design methodology of the artificial function itself. To build a minimal cell using a bottom-up approach (see Fig. 3), the biologist would need to select non-living components (amino-acids, genes…), assemble them into subsystems (enzymes interacting with other components) that when combined together, give rise to a functioning cell. Continuing with the example of building a minimal cell (see Fig. 3), with a top-down the biologist would start with a living cell and modify it, by stripping it from various functions while the cell is still viable.

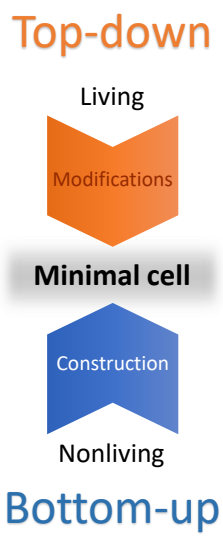


Fig. 3 Illustration of the top-down and bottom-up approaches in biology.

1.5. Microelectronics design flow

In order to envision a biology design flow, we first take a look at the traditional microelectronics design flow.

1.5.1. The beginnings

The density of integration of electronic circuits has increased since the 1970s (Moore's law). The complexity of current circuits is such that their design cannot be done without computer-aided design (CAD) tools. Historically, the first CAD tool is probably SPICE (Simulation Program with Integrated Circuit Emphasis). This software was created in the 1970s at the University of Berkeley (California) by the team of Larry Nagel. SPICE capabilities include the instantiation and the virtual connection of electronic devices as it would be done in actual electronic circuits, the simulation of the circuit realized and the analysis of simulation results for circuit behavior prediction and optimization purposes. SPICE quickly became a standard in the field of circuit simulation and has been adopted both by circuit manufacturers and by designers.

1.5.2. The path to standardization

Subsequently, other standards appeared for the representation of circuits, the description of their function and their characteristics. In particular, electronic devices are always provided with datasheets containing a set of information required for the designer, such as the pinout of the device, several characteristics, the mechanical drawing, etc. These information are strictly codified so that it is easy for a user to read them and to compare several components, even if they are provided from different vendors.

Another example of standardization concerns the design of digital circuits. In the early 1980s, when the number of digital gates integrated in a single circuit exceeded the thousand, many manufacturer started developing proprietary software dedicated to their circuits. In 1993, manufacturers of microelectronics decided to pool their know-how in order to define together and adopt standard languages for the description of digital circuits. This is how the VHDL (VHSIC (Very High Speed Integrated Circuit) Hardware Description Language) and Verilog languages were born. They are now widespread and allow an easier exchange of models and better interoperability between software.

The tools and standards developed for the design of electronic circuits allowed a separation of the trades between “those who know how to design a circuit by assembling elementary bricks” (the designer) and “those who know how to realize the basic bricks” (the process engineers and circuits manufacturers). They exchange information through model libraries, often called design kits, developed by process engineers or circuit manufacturers and which contain all information required by the designer to interconnect them with the rest of the circuit without being concerned by what it contains. This "black box" approach also led to a hierarchical approach of the design of circuits and the notion of “level of abstraction”. At each level of abstraction, the system is described with the level of detail which is suitable for its interface with the rest of the system. For example, an analog filter can be seen, by decreasing abstraction level, as a transfer function, as a quadrupole, as an assembly of passive components and operational amplifiers, or as an assembly of passive components and transistors.

1.5.3. The shift towards a top-down approach

Historically, electronic circuits were rather designed with a bottom-up approach. The designers started from their know-how and assembled elementary bricks until the desired function was achieved. With the increasing complexity of electronic systems, this approach is no longer possible because it becomes very difficult to predict what type of elementary brick are required from the high level specifications of the system. Thus, top-down approach gradually replaced bottom-up approach. As seen above, even though systems designed this way should be “right by design”, a practical implementation of this approach is difficult, especially for large multi-domain systems. Hence, a handful of CAD tools, generic or domain-specific have been developed to assist designers teams. As this design approach combines virtual prototypes (assembly of models) and actual ones, it is often called “functional virtual prototyping”.

1.5.4. Digital and analog parts in the design flow

In electronics, digital parts and analog parts are often distinguished. On the one hand, digital parts described by Boolean functions are very well adapted to the top-down approach, because there exist CAD tools to handle and automate the design flow. Thus, electronic gate-level schematics, integrated circuit layouts or a programming file for programmable logical devices can be generated directly and automatically from a high-level description. On the other hand, analog parts, described by Kirchhoff’s laws, are much more complex to design. The subdivision, the modeling of each sub-circuits and the wiring between models is manual and the design of each sub-circuit is mostly done with a trial and error approach. Up to now, plenty of tools have been developed in order to manage models, analyze and compare circuit performance or optimize circuit parameters. Nevertheless, despite some breakthroughs in the last 20 years, no generic automated analog circuit synthesizer has been capable of establishing itself as a standard amongst analog designers’ community. To illustrate the purpose, it is now estimated that more than 75% of manufactured integrated circuits have at least an analog part. This part represents on average 2% of the functionality of the circuit but its design requires 20% of the project’s resources and is responsible of 40% of the faults which lead to a redesign.

In hindsight, the evolution of electronic circuits since the 1970s has been spectacular. The density of integration, and therefore the complexity of the circuits realized has progressed by six orders of magnitude in forty years. This development is of course due to the technological progress made in the manufacturing of integrated circuits, but also to the development and acceptance by the community of standards and CAD tools. If we compare the timeline of electronics and the one of synthetic biology,

we are, for synthetic biology, at the beginning of the history. This corresponds to when the first precursor tools that are now widely used in electronics system design were born.

1.6. Synthetic biology design flow

Here is presented the design flow for synthetic biology (Fig. 4) and more particularly for gene regulatory networks (GRN) (Gendrault, Madec, Lallement, et al. 2014). The input is a set of specifications and the output is a list of gene/cells composing the GRN. This workflow could be implemented in a single software handling all these steps but also many tools can be adapted separately and exchange data in a compatible format. We will discuss in more details the four main stages that should compose this tool suite.

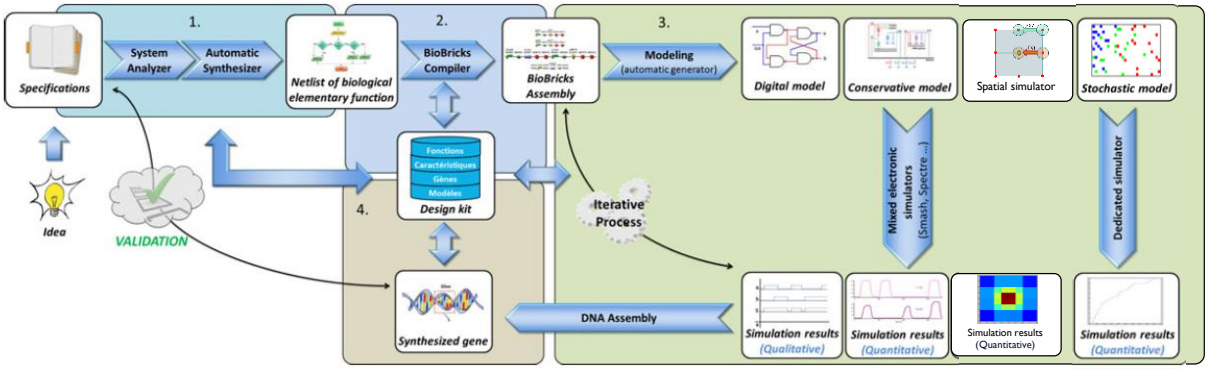


Fig. 4 A microelectronics-inspired design flow for synthetic biology (Gendrault, Madec, Lallement, et al. 2014)

The first block is the high-level system analyzer. The goal of this first step is to find the topology of the biological system to design (e.g. a GRN) that matches a specification given as an input. This specification can be either a Boolean function, a transfer function or the temporal evolution of the system after a given stimulus.

The second block is in charge of selecting appropriate biological components from a database to match the previously selected functions. These components are often called parts or Biobricks and mainly consist in plasmid or simply DNA sequence of a promoter, a gene and a terminator. A database like the iGEM (International Genetically Engineered Machine) Catalog ("Registry of Standard Biological Parts" 2015) contains more than 12000 parts, arranged in different categories (see Fig. 5). Unlike their microelectronic counterparts in TTL libraries, Biobricks are not documented in a standardized way. When taking two of the top 10 most documented parts (http://parts.igem.org/Part:BBa_K863005; http://parts.igem.org/Part:BBa_K404163), one can see many differences in the type of information available and how it is displayed.

The function of the parts are described in text and/or on a figure, one of the parts only (BBa_K863005) has a descriptive subtitle from which data could be extracted. Most of the data consists in cultivation, purification and analysis details of the DNA for BBa_K863005 whereas transfection and transduction details with microscopy pictures could be found for BBa_K404163. None of these two parts contain any standardized file containing their abstracted function, nor kinetic experiments or dynamic constants estimations. The only standard criterion shared by both entries is the BioBrick RCF standard. This is an assembly standard designed by Tom Knight for interchangeable parts. RCF-compatible plasmids display certain sequence properties (like a given prefix and suffix sequences, containing restriction enzyme sites for example). These type of entries are however still far from the standard

datasheet of parts envisioned by Drew Endy (see Fig. 6). Such a datasheet allows a direct and clear understanding of the inputs and outputs of a part, and its functioning.

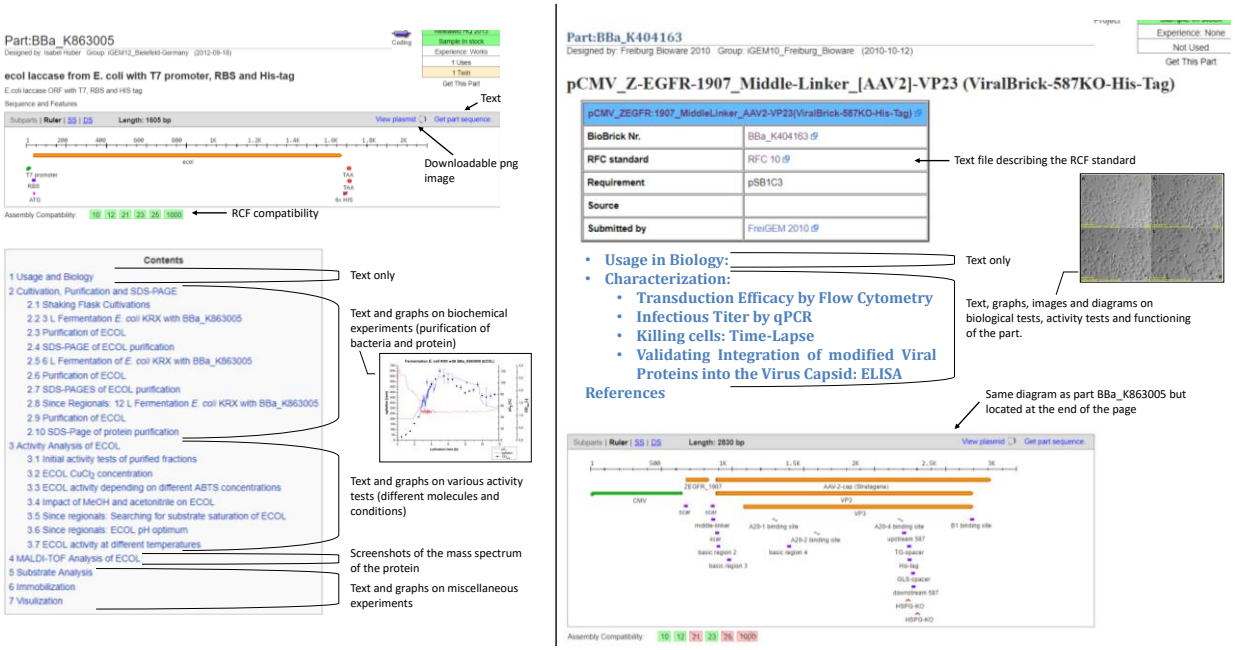


Fig. 5 Summary of the 12-pages long entry of part BBa_K863005 (left panel) and the 2-pages long entry of part BBa_K404163 (right panel). Black comments are added. Both of these entries appear in the top 10 most documented parts. On the left panel, the entry displayed a full table of contents as shown. On the right panel, the entry had no table of contents: the one displayed here is a copy/pastes of the titles of the paragraphs (without font modifications). Moreover, the 'plasmid' diagram was displayed at the bottom of the page on the right entry, contrary to the entry on the left. Comments and table of contents show how different the two entries are. Left panel URL: http://parts.igem.org/Part:BBa_K863005. Right panel URL: http://parts.igem.org/Part:BBa_K404163.

The third block consists in the virtual prototyping (modeling and simulation) of the assembly of BioBricks. Depending on the system complexity, the degree of accuracy required and the allowed simulation time, different levels of abstraction are relevant. First of all, as mentioned before, many biological systems sub-modules can be seen as bio-logic gates. The module can have an off and an on state. It is therefore possible to reuse digital abstraction models of microelectronics, with the difference that here the information is not carried out by electrons but by a molecule. As biology is not inherently digital, a lower level of abstraction can be used: multivalued logic. This type of logic abstraction is multi-valued: the module can have intermediary states between being completely off or completely on. At last, a more accurate way of describing biological systems is to use ordinary differential equations (ODE) that have many parameters to fit precisely a module's behavior. In this block, many tools have been implemented and others still lack; they will be further detailed in section…

The fourth block corresponds to the foundry of the circuit in micro-electronics. With this step, an actual system is created. This system can be tested at the wet bench and according to the result, a new loop of system design/optimization can be launched.

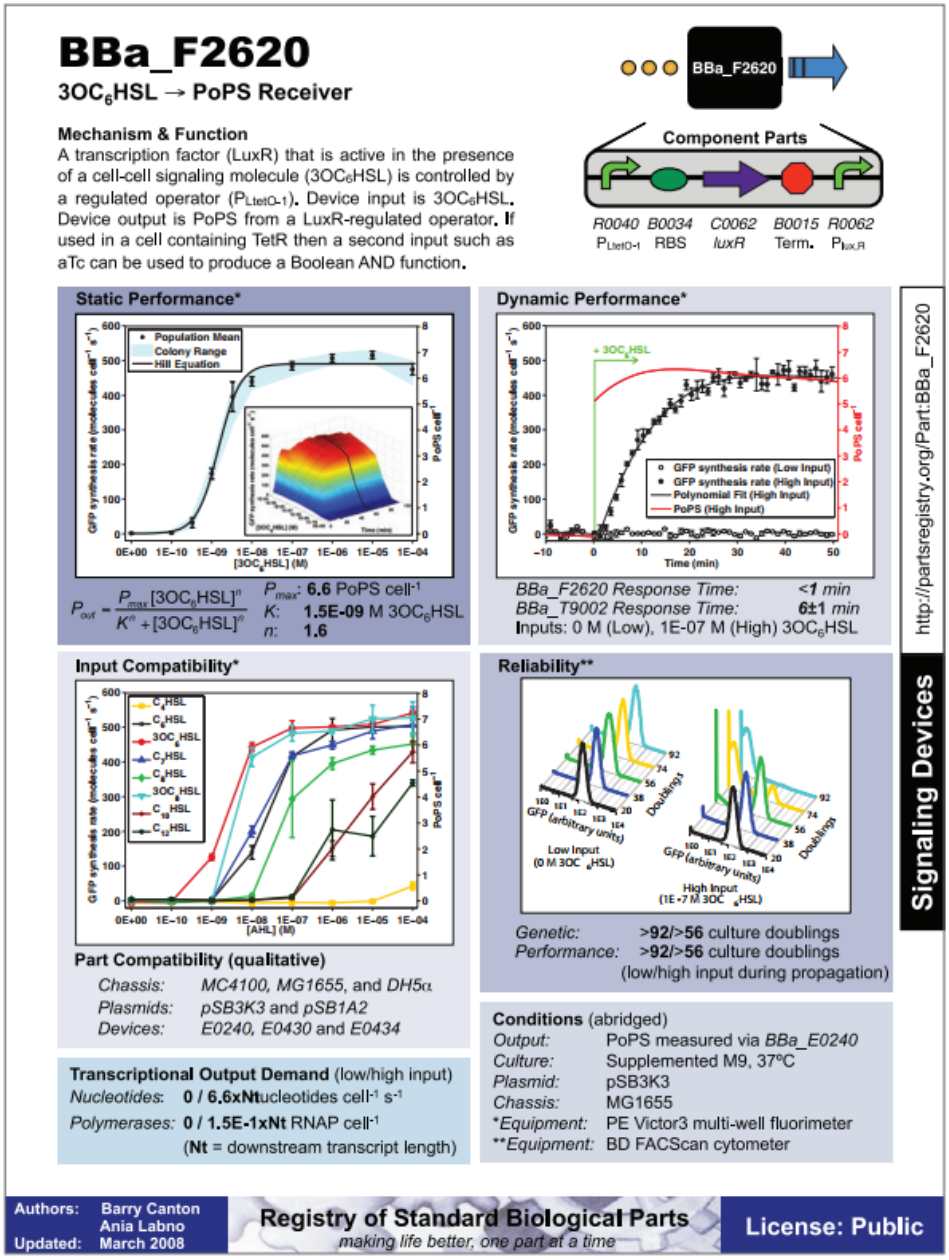


Fig. 6 Proposition for a standard datasheet for biological parts, by Drew Endy (Canton, Labno, and Endy 2008)

2. Models and simulators

As mentioned previously, the third block is composed of tools able to simulate the behavior of a GRN. According to the requirements (what kind of function is to be realized? With what precision should the GRN match the requirements?), different types of models can be used.

2.1. Digital abstraction

At the highest level of abstraction, a gene can be seen as having two states: an OFF state corresponding to its cognate protein not being expressed, and an ON state corresponding to the contrary. In reality, we define thresholds of the protein concentration: one under which the gene is OFF or the protein is 'absent', and another one, higher, above which the gene is ON or the protein 'present'.

The inputs and outputs of GRNs are molecules, such as transcription factors for the inputs or a drug for the output. The ON and OFF states can also be defined for these molecules. With this abstraction, it is possible to describe a GRN's function by the states of its outputs according to the states of its inputs. This corresponds to a logic function, this is why we can make the analogy between logic gates and GRN. The OFF state of a molecule correspond to a logic 0 and the ON state corresponds to 1.

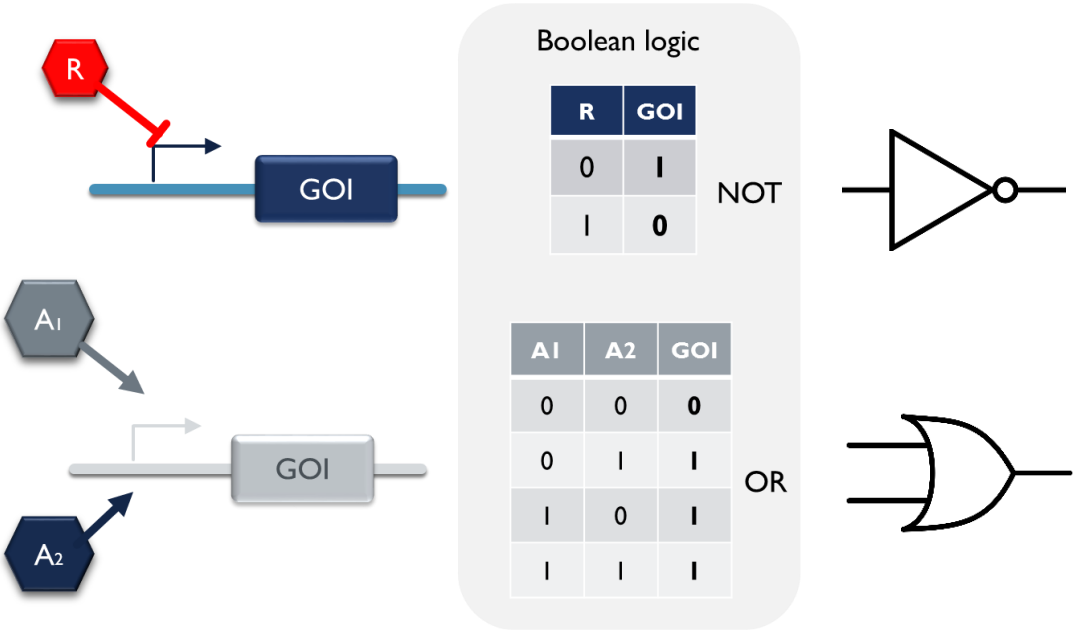


Fig. 7 Different representations of two different logic gates. The upper one is a NOT gate and the lower one is an OR gate. At the left, a possible implementation with genes and regulators. In the middle, the truth table of the gate. At the right, the electronic representation of the gate.

As an example, let be our gene of interest express the protein Gene Of Interest (GOI). In one case (Fig. 7 upper part), this gene is inhibited by a so called repressor R (in red on Fig. 7). While the repressor is absent, meaning its state is 0, the gene can express GOI, so that the state of GOI is 1. Conversely, while the repressor is present (1), the gene is no longer able to express GOI, which state becomes 0. This can be summarized in a truth table. We see that this behavior corresponds to the behavior of a NOT gate, with the input being R and the output GOI. In another example, our gene of interest is activated by two activators, A1 and A2 (in grey and blue on Fig. 7). Here, the GOI can only be expressed when at least one activator is present. This translate into a logic gate active when at least one of its inputs is at 1, a so-called OR gate. Many biological implementations of the classical logic gates can be found in the literature (Fig. 8).

2.2. Multivalued logic

A way to improve the modeling of the behavior of a promoter and its regulator is to use multivalued logic. Multivalued logic allows another distinction between the active and inactive state of a variable, by introducing intermediate levels. Both inputs and outputs can be at this intermediate level (see Fig. 9).

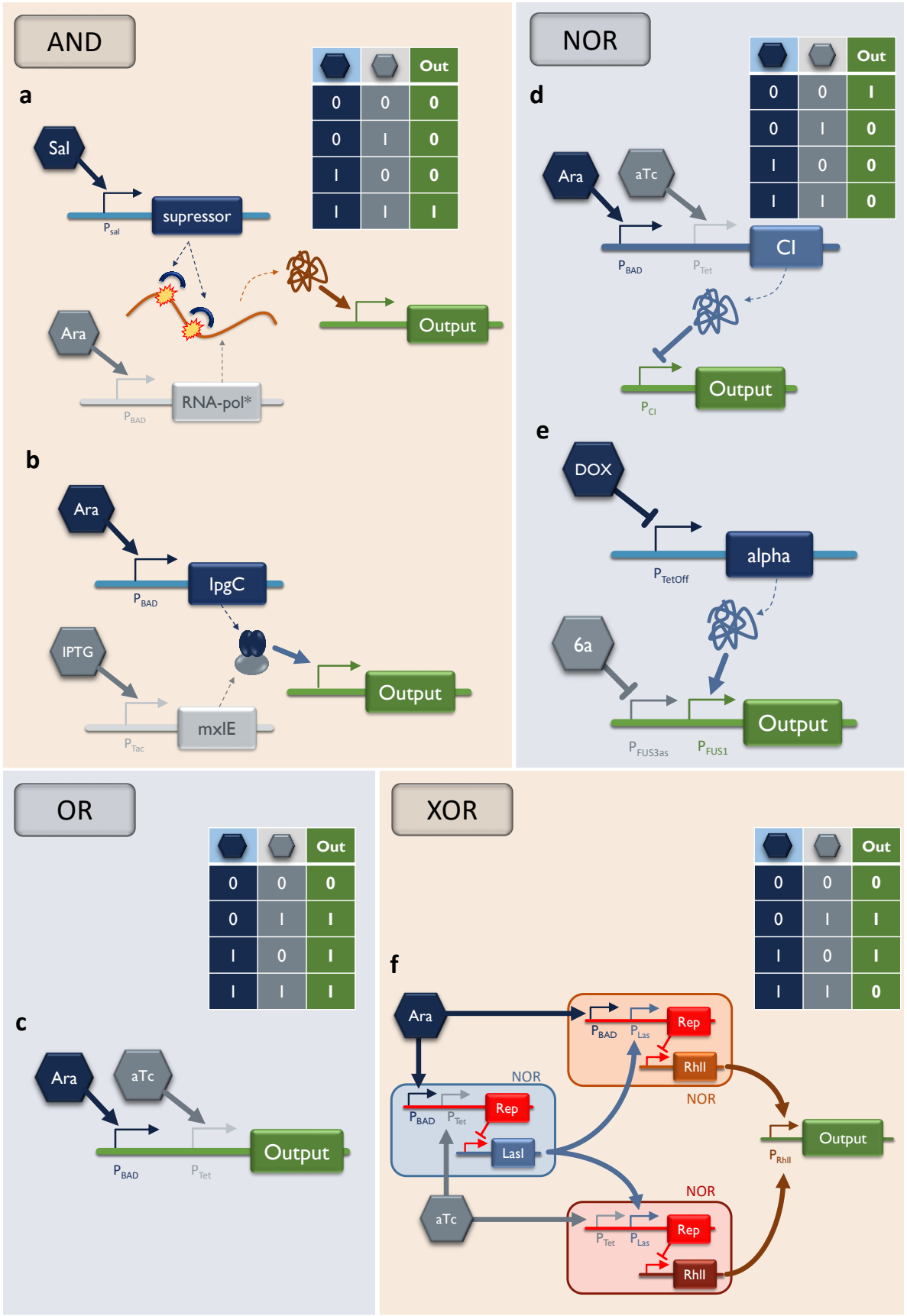
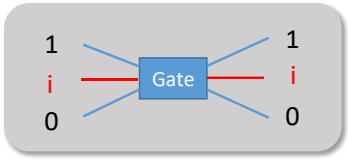

AND
a
Sal
supressor
P_{sal}
Ara
RNA-pol*
P_{BAD}
Output
Out
0 0 0
0 I 0
I 0 0
I I I
b
Ara
IpgC
P_{BAD}
IPTG
mxlE
P_{Tac}
Output
NOR
d
Ara
aTc
CI
P_{BAD}
P_{Tet}
P_{CI}
Output
Out
0 0 I
0 I 0
I 0 0
I I 0
e
DOX
alpha
P_{TetOff}
6a
P_{FUS3as}
P_{FUS1}
Output
OR
c
Ara
aTc
P_{BAD}
P_{Tet}
Output
Out
0 0 0
0 I I
I 0 I
I I I
XOR
f
Ara
P_{BAD}
P_{Las}
Rep
RhlI
NOR
P_{BAD}
P_{Tet}
Rep
LasI
NOR
aTc
P_{Tet}
P_{Las}
Rep
RhlI
NOR
P_{RhlI}
Output
Out
0 0 0
0 I I
I 0 I
I I 0

Fig. 8 : GRN implementations of classical logic gates and their truth tables: AND, NOR, OR and XOR. GRN a is based on a corrupted sequence of the T7 RNA-polymerase (Anderson, Voigt, and Arkin 2007). The sequence contains 2 premature stop codons that can be overstepped with the presence of the suppressor protein. The successful production of T7 RNA-polymerase enables the expression of the output. GRN b codes for a transcription factor, MxiE, that is activated in the presence of its chaperone IpgC (Moon et al. 2012). Only the activated transcription factor can activate the expression of the output. GRN c uses a promoter activated by either arabinose (Ara) or anhydrotetracycline (aTc) (Tamsir, Tabor, and Voigt 2011). GRN d is an extension of GRN c: the first gene codes for a repressor protein, CI, that inhibits the expression of the output when present. In its absence, the output is produced (Tamsir, Tabor, and Voigt 2011). GRN e uses two inhibitions (simplified from (Regot et al. 2011)). The production of the alpha pheromone is repressed by doxycycline (DOX). After unrepresented steps, the alpha factor activates the expression of the output, which is also inhibited by 6a. GRN f is composed of 3 similar NOR gates (see GRN d) split across 4 "wired" cells. Indeed, they use Pseudomonas aeruginosa cell-to-cell communication systems (LasI and RhlI) to pass the signal from a NOR gate to another one.



In	Out
0	1
i	0
1	0

In	Out
0	i
i	0
1	0

In	Out
0	1
i	1
1	0

Fig. 9 Multivalued logic truth tables. i represents an intermediate level which is between 0 and 1.

It is therefore possible to use different representation of a "NOT" gate, just as different proteins would have different level of activation. Several examples use multivalued logic to design systems. We can notably mention Bernot *et al.* (Bernot et al. 2004) and also an approach based on fuzzy logic by Gendrault et al. (Gendrault, Madec, Lemaire, et al. 2014).

2.3. Ordinary differential equations

Both digital and fuzzy logic abstractions allow the representation of only a quantified concentration of proteins. To represent concentrations as continuous variable, we must use differential equations (cf Chapter 2). This kind of description is more accurate, allowing to describe the variation of a protein's concentration in function of the concentration of its regulators. The model of protein expression can be fine-tuned with numerous parameters (e.g. kinetic parameters of reactions, affinity constants…).

In biology, ODE-level descriptions of biological systems are often formalized by an SBML (Systems Biology Markup Language) file. SBML is a language introduced in 2003 (M. Hucka et al. 2003) and composed of a large list of markups corresponding to the different elements of a biological model. For example, SBML allows the definition of units, of compartments where different reactions can take place, of the species involved in these reactions, of the parameters of these reactions, of events to be applied when simulating the model (see (Michael Hucka et al. 2017) for more details).

Many simulation tools handle SBML. We can notably cite COPASI, one of the most widely used simulation tool of biological systems. COPASI also has a GUI (Graphical Use Interface) that can be used to define the biological network to simulate and the associated parameters (Fig. 10). The classical time

course and steady state simulations can be performed. Moreover, COPASI proposes other types of simulations like parameter scan and flux balance analysis among others. COPASI allows the specification of compartments of different sizes but does not provide a space simulation. Simulation data can be viewed in the software or exported in a text file.

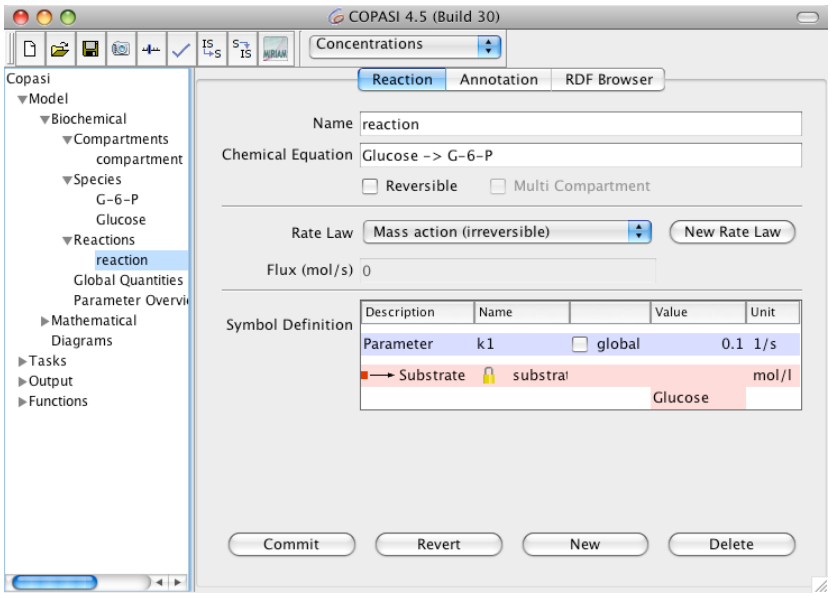


Fig. 10 COPASI interface

We can also mention BioCham, which is able to read SBML files as well as ODEs written in a specific format. BioCham shares similar features to COPASI in terms of simulations and plotting but does not have a GUI. Notably, BioCham proposes a furnished set of qualitative analysis of the network: the network is analyzed at the Boolean level. It is not possible to simulate space with BioCham.

Virtual Cell or VCell is also a major tool for simulating biological systems, with a GUI and a comprehensive set of analyses. As this tool features a space simulator, we will discuss it in more details in Chapter 6.

However, these tools are not robust enough to tackle very large number of equations. This is why a formalism based on the analogy between electronics and biology seen in Chapter 2 was developed by Madec *et al.* to simulate biological networks with electronic simulators (Madec, Lallement, and Haiech 2017). BB-SPICE is able to read an SBML file and produces a SPICE simulation ready file. It has been demonstrated that BB-SPICE can handle a model of a biological network with more than 10,000 reactions while COPASI crashes due to the lack of memory with only 1,000 reactions. This formalism is reused in Chapter 6.

2.4. Stochastic simulation

ODEs imply that the observed quantities are large and therefore the mean behavior of the particles is a good representation of each particle. However, in problems involving low numbers of molecules, a new level of simulation is needed: stochastic simulation. This level of abstraction does not compute the concentration of a molecule but its number. The reactions are not represented by kinetic constants but probabilities for the reaction to occur. In electronics, electrons are often considered as a whole and not individually. The stochastic simulation is therefore not used to simulate electrons. However,

physical reactions involved in the functioning of the circuit remain random at the microscopic scale. To account for this stochasticity, noise is added to the variables of a deterministically-solved system.

COPASI and BioCham both feature stochastic simulations. We see on Fig. 11 the difference between a deterministic and a stochastic simulation. The global behavior is the same but local small variations illustrate the stochasticity of any chemical reaction when observed on a microscopic scale.

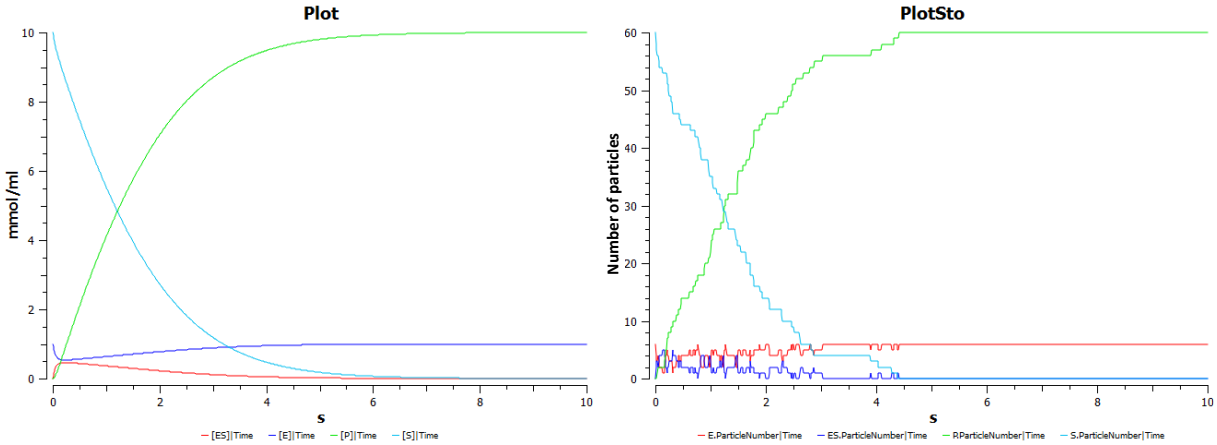


Fig. 11 COPASI results for a deterministic (left) and stochastic (right) simulation of an enzymatic reaction (E+S↔ ES→P). Light blue: substrate S. Dark blue: enzyme E. Red: enzymatic complex. Green: product.

2.5. Spatio-temporal simulation

As the designed systems grow in size, it has become necessary to split them into different cells. Indeed, contrary to electronics where the signal is confined in a wire that leads the electron only to where we want them to go, in biology the molecules are free to diffuse in the whole cell. This means that they can potentially interact with every component of the system. As the number of components (genes) in a system increases, it becomes harder to avoid cross talk between the components. However, this diffusion is limited by various membranes. The plasma membrane is one of them, it delimits the cell itself. Proteins cannot cross a naked plasma membrane without help. Furthermore, different compartments in the cell (e.g. the nucleus, mitochondria, endoplasmic reticulum…) are also delimited by membranes.

Taking these phenomena into account in biological models is mandatory to tackle the virtual prototyping of the next generation of artificial biosystems. Existing solutions as well as the original approach we developed are described in the third part of this manuscript (Chapter 6).

3. Designing GRNs

CAD tools allow an easy handling of models of the systems and cognate biological data. Such tools exist for synthetic biology. Design automation is the next step: the CAD tool would not only need to enable the manipulation and simulation of the system but also, given a set of requirements, create the system. Here we review the current tools and their capabilities.

3.1. Assisted-design tools

Many such tools exist for synthetic biology: TinkerCell (Chandran, Bergmann, and Sauro 2009), CellDesigner (Funahashi et al. 2003), BioJADE (Goler 2004), ProMot (Mirschel et al. 2009), GenoCAD (Czar, Cai, and Peccoud 2009). Most of them take advantage of the “design by parts” that the BioBricks

standard enables. These tools often have a visual interface for an easy manipulation of the parts, therefore facilitating the design of circuits composed of many parts.

3.1.1. TinkerCell

Developed by a team from Seattle, TinkerCell presents a graphical interface where the user can draw a biological systems. Many biological reactions are available (synthesis, degradation, enzymatic catalysis, gene regulation via transcription factor binding). Some reactions, like the enzymatic catalysis, can be specified. For example, the user can specify the different states of the enzyme/substrate complex (see Fig. 12).

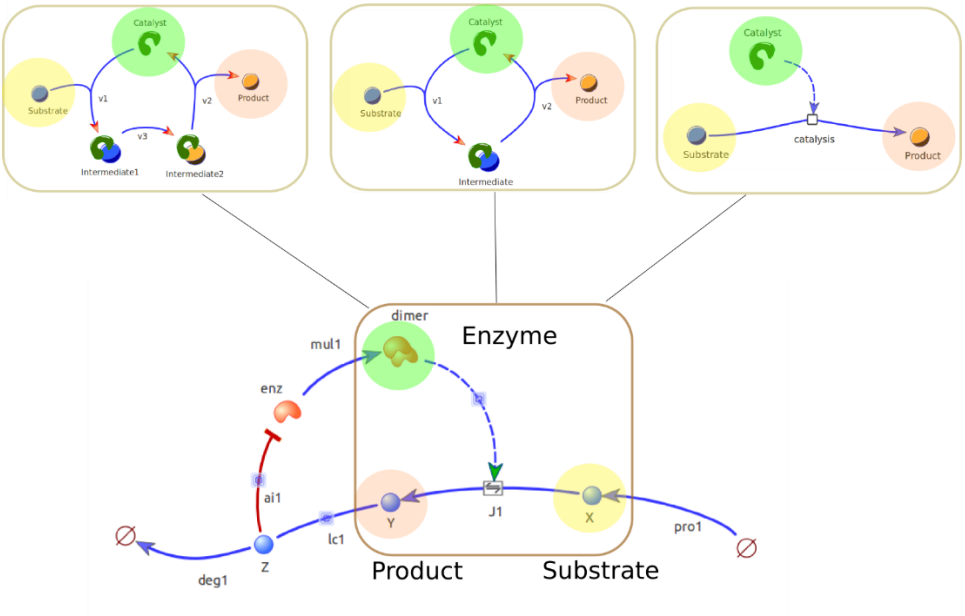


Fig. 12 TinkerCell diagram. The enzymatic catalysis can be described by different models

Each reaction and each type of reaction have an associated model, used to perform various analysis (deterministic simulation, stochastic simulation, steady state computation, parameter scan, etc). Most of them are performed using COPASI. The idea is that the biologist draws the imagined biological network on TinkerCell and simulates it. According to the delivered results, he can change elements of his systems and simulate it again. The user can also load a sequence which is analyzed by a Python script to determine what type of part it is (promoter, RBS (ribosome-binding site), coding region, or transcription factor). TinkerCell is pluggin-friendly, as it is the desire of the developers of the tool to allow external inputs and editions when certain functions/components are needed.

3.1.2. BioJADE

BioJADE was developed at the Massachusetts Institute of Technology (MIT) by the same team that created the BioBricks repository. BioJADE therefore makes an interactive use of the BioBricks. BioJADE key concept is “genetic component prototype”: they regroup circuits by their functions (e.g. NAND, NOT, reporter, etc) and use the abstracted function as the building block (see Fig. 13).

BioJADE enables system designers to specify a system abstractly with an XML file or a full RDBMS (a relational database). They can also design new parts.

Then, they can tune the system, simulate its behavior using a variety of simulators (from simple logic tests to the level of protein binding), and finally package the part for use by either the designer or the public. There is the possibility to interface tools written in other languages.

3.1.3. ProMoT

ProMoT was designed by a German team from the Max Planck institute. ProMoT uses models based on digital abstraction as well as differential algebraic equations. Systems can be editing with a graphic interface, or loaded from a SBML file. A validation again structural modeling errors is performed while editing.

The representation displayed by the graphical interface is zoom-dependent: zooming on a module reveals its specific components. Otherwise, only relevant components are showed. This adaptive visualization serves to aid the user in analyzing and interpreting the simulation results.

3.1.4. GenoCAD

Developed in Virginia, GenoCAD assists the user in constructing GRNs with the final goal to obtain the DNA sequence for synthesis. GenoCAD is an online tool, with a library of parts for various organisms (see Fig. 15).

GenoCAD also converts DNA sequences into genetic parts. Designs can be exported as FASTA (DNA sequence format) or text files. GenoCAD designs rely on the concept of grammar. A grammar defines an ordered list of parts a construct should have. They can be species specific and different grammars are available for a given construct (see Fig. 16). One can also create its own grammar.

Simulation of the design is performed using the COPASI engine.

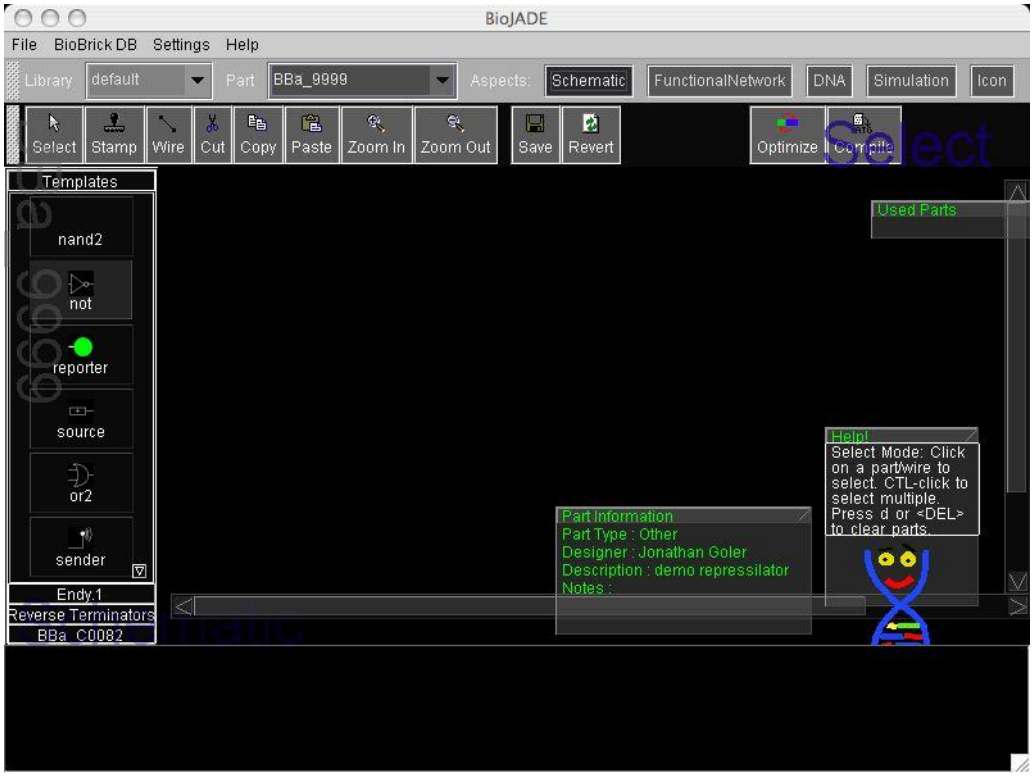


Fig. 13 BioJADE interface

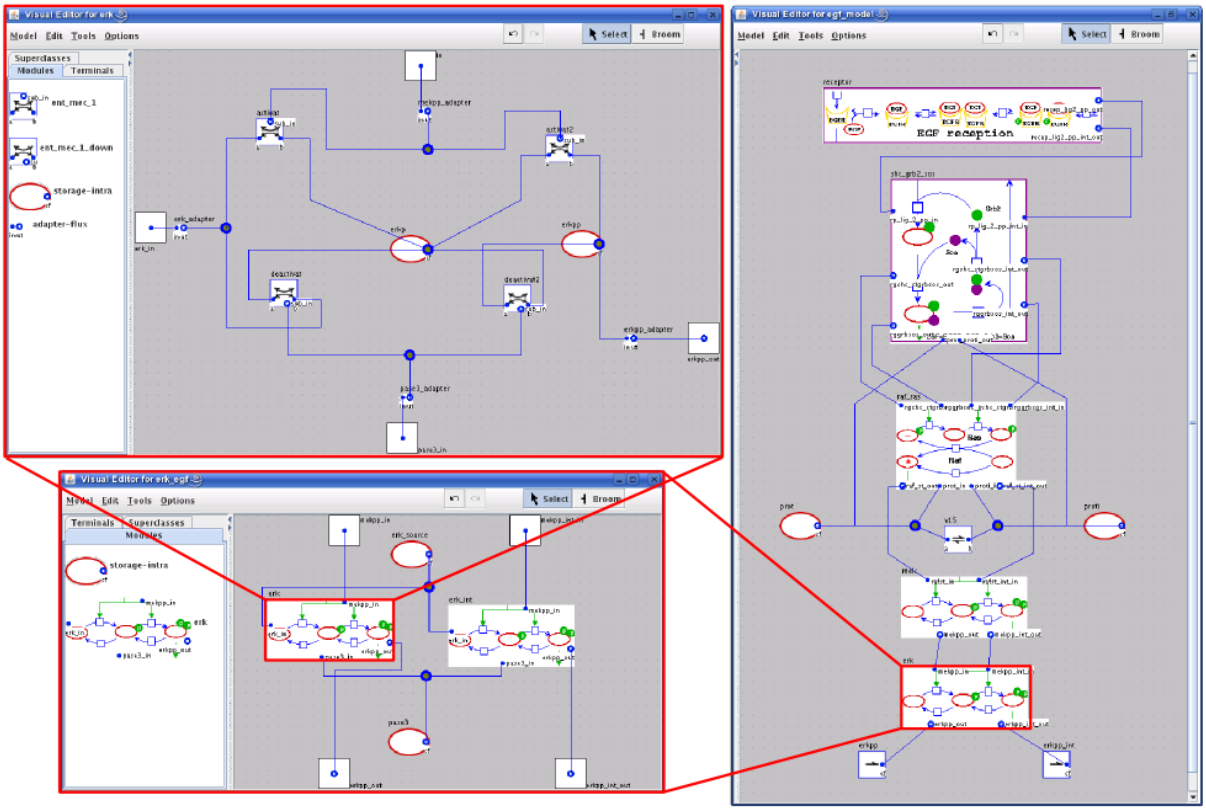


Fig. 14 ProMoT zoom-dependent visualization

Table 1 Summary of the reviewed biological CAD tools

	Input	*Output*	*DNA sequence support*	*Possibility to implement custom parts/functions*	*Automated design*
TinkerCell	Graphical interface, DNA sequence	SBML, Octave, Matlab	yes	yes	no
BioJADE			yes		
ProMoT	SBML	SBML, Matlab	no		
GenoCAD	Graphical interface	SBML	yes	yes	no

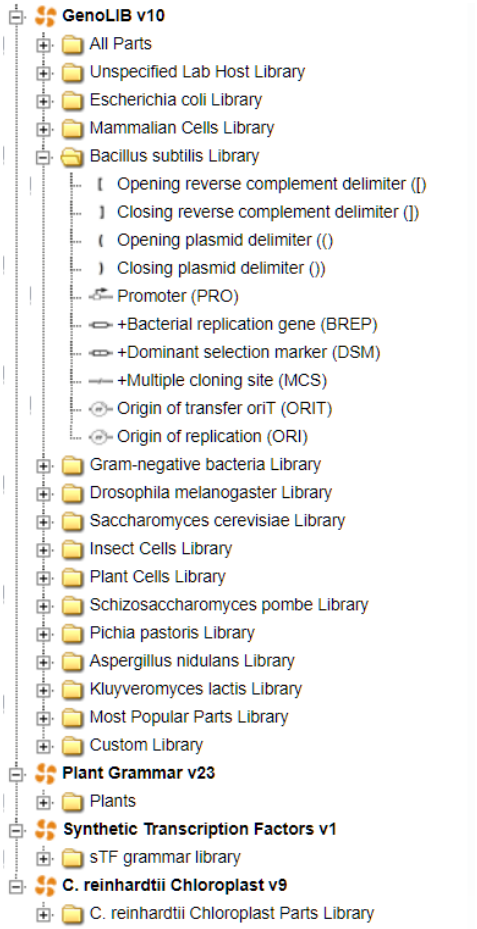


Fig. 15 Overview of the GenoCAD library of parts.

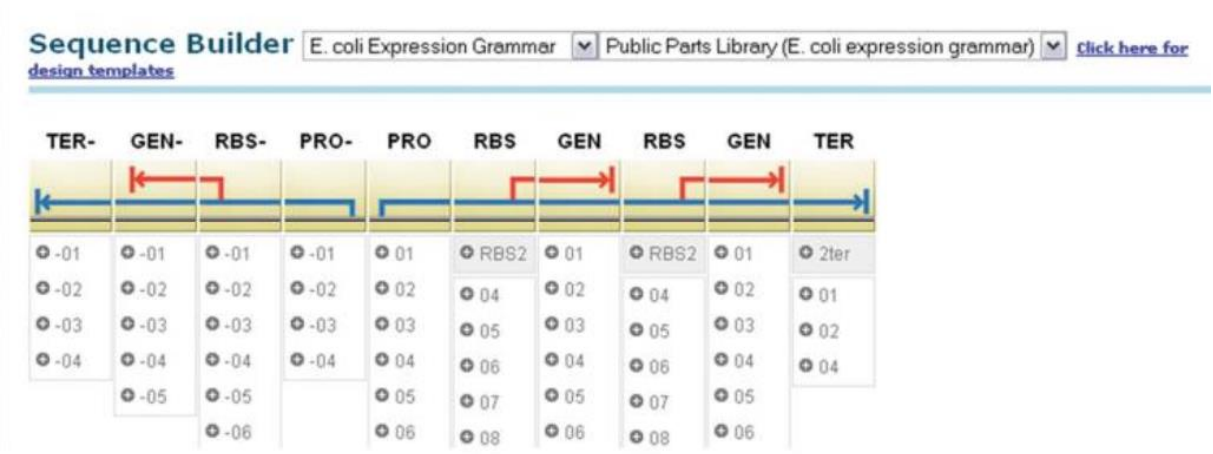


Fig. 16 Example of a construct (a toggle switch) with GenoCAD. The different parts (TER, RBS, …) and the available choices underneath are shown.

The reviewed tools are only a fraction of all the existing tools. Most of them support DNA sequences, a useful feature for a direct implementation of the system at the wet bench. Some tools (mainly BioJADE and GenoCAD) are connected to a furnished database of parts. Automatic retrieval of newly added parts in the BioBrick repository has however not been encountered during the reviewing of any tool. The possibility to add custom parts is also interesting, as biological construct are constantly evolving and new construct are appearing. The SBML language is expectably a common point for all format used for exchanges. Noteworthy is also the BioModels repository of computational models of biological processes, a European Bioinformatics Institute initiative (Juty et al. 2015). The models are written in SBML. In this database, more than 7000 models published in literature are available in a standard layout.

3.2. Automated-design tools

With the previous tools, the traditional approach of trial and error for handmade optimization of systems is aided. The next step is to automate this process. Given a specification, an automated design tool finds the optimal system fitting this specification. We will discuss these tools in more details in Chapter 4.

4. References

Anderson, J Christopher, Christopher A Voigt, and Adam P Arkin. 2007. “Environmental Signal Integration by a Modular AND Gate.” *Molecular Systems Biology* 3. European Molecular Biology Organization: 133. doi:10.1038/msb4100173.

Bernot, Gilles, Jean-Paul Comet, Adrien Richard, and Janine Guespin. 2004. “Application of Formal Methods to Biological Regulatory Networks: Extending Thomas’ Asynchronous Logical Approach with Temporal Logic.” *Journal of Theoretical Biology* 229 (3). Elsevier: 339–47.

Canton, Barry, Anna Labno, and Drew Endy. 2008. “Refinement and Standardization of Synthetic Biological Parts and Devices.” *Nature Biotechnology* 26 (7). Nature Publishing Group: 787–93. doi:10.1038/nbt1413.

Chandran, Deepak, Frank T Bergmann, and Herbert M Sauro. 2009. “TinkerCell: Modular CAD Tool for Synthetic Biology.” *Journal of Biological Engineering* 3 (1): 19. doi:10.1186/1754-1611-3-19.

Czar, Michael J, Yizhi Cai, and Jean Peccoud. 2009. “Writing DNA with GenoCAD.” *Nucleic Acids Research* 37 (Web Server issue). Oxford University Press: W40-7. doi:10.1093/nar/gkp361.

Funahashi, Akira, Mineo Morohashi, Hiroaki Kitano, and Naoki Tanimura. 2003. “CellDesigner: A Process Diagram Editor for Gene-Regulatory and Biochemical Networks.” *BIOSILICO* 1 (5): 159–62. doi:10.1016/S1478-5382(03)02370-9.

Gendrault, Yves, Morgan Madec, Christophe Lallement, and Jacques Haiech. 2014. “Modeling Biology with HDL Languages: A First Step toward a Genetic Design Automation Tool Inspired from Microelectronics.” *IEEE Transactions on Biomedical Engineering* 61 (4). IEEE Computer Society: 1231–40.

Gendrault, Yves, Morgan Madec, Martin Lemaire, Christophe Lallement, and Jacques Haiech. 2014. “Automated Design of Artificial Biological Functions Based on Fuzzy Logic.” In *Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE*, 85–88.

Goler, Jonathan Ari. 2004. “BioJADE: A Design and Simulation Tool for Synthetic Biological Systems.” https://dspace.mit.edu/handle/1721.1/30475.

Hucka, M., A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, et al. 2003. “The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models.” *Bioinformatics* 19 (4): 524–31. doi:10.1093/bioinformatics/btg015.

Hucka, Michael, Frank T Bergmann, Sarah M Keating, Chris J Myers, Brett G Olivier, Sven Sahle, James C Schaff, Lucian P Smith, Dagmar Waltemath, and Darren J Wilkinson. 2017. “The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 2 Core.” http://sbml.org/specifications/sbml-level-3/version-2/core.

Juty, N, R Ali, M Glont, S Keating, N Rodriguez, MJ Swat, SM Wimalaratne, et al. 2015. “BioModels: Content, Features, Functionality, and Use.” *CPT: Pharmacometrics & Systems Pharmacology* 4 (2): 55–68. doi:10.1002/psp4.3.

Madec, Morgan, Christophe Lallement, and Jacques Haiech. 2017. “Modeling and Simulation of Biological Systems Using SPICE Language.” *PloS One* 12 (8). Public Library of Science: e0182385. doi:10.1371/journal.pone.0182385.

Madec, Morgan, François Pecheux, Yves Gendrault, Elise Rosati, and Christophe Lallement. 2016. “GeNeDA: An Open-Source Workflow for Design Automation of Gene Regulatory Networks Inspired from Microelectronics.” *Journal of Computational Biology*.

Mirschel, Sebastian, Katrin Steinmetz, Michael Rempel, Martin Ginkel, and Ernst Dieter Gilles. 2009. “ProMoT: Modular Modeling for Systems Biology.” *Bioinformatics* 25 (5): 687–89. doi:10.1093/bioinformatics/btp029.

“Registry of Standard Biological Parts.” 2015. Accessed November 6. http://parts.igem.org/Main_Page.

Part Two

Design Automation of Biological Systems

Introduction of Part Two

As mentioned in the first part of this manuscript, automated design tools are still lacking for biology. However, automated part selection derived from a high-level specification has seen several successful breakthroughs, confined to the Boolean abstraction of Gene Regulatory Networks (GRNs). TASBE, developed by the Massachusetts Institute of Technology and Boston University, is one of them (Beal et al. 2012). It consists in a tool-suite (including several new or existing tools such as Proto, BioCompiler, MatchMaker and BiOCAD) that starts from high-level specifications given in a specific language and leads to a DNA assembly that meets the specifications. The upstream stages of TASBE include the high-level description and "Bio-compilation" (interpretation of the high-level description as a set of abstracted elementary biological functions) (Densmore et al. 2010; Bilitchenko et al. 2011; Beal, Lu, and Weiss 2011; Yaman et al. 2012). Another interesting approach, also based on Boolean abstraction of GRN, has been suggested by Marchiso *et al*. (Marchisio and Stelling 2011).

We also mention in the first part that many synthetic GRNs can be described by a Boolean equation. In this case the problems encountered for their design are very similar to the problems encountered in digital synthesis in microelectronics. We therefore want to benefit from electronics' years of experience in addressing this issue by reusing and adapting their tools to our problem at hand. This is tackled in the first chapter of this section (chapter 4). First, we present a tool suite (GeNeDA) adapted from electronics to automate the design of digitally-described GRNs. GeNeDA is first tested and validated on combinatorial circuits. Then, in a second time, the design of sequential functions is discussed. In particular the robustness of the designed system towards the variation of its biochemical parameters is tested. As for microelectronics circuits, the realization of synchronous circuit is often necessary to avoid the risks of malfunction and instability. This, however, implies the ability to design a robust biological D-flip-flop, which is presented in the third subpart of the chapter.

The second chapter of the section (Chapter 5) deals with the design automation of biological system that cannot be abstracted by a Boolean function. For these questions, we land into the field of "analog synthesis", an area of investigation of electronics for which several solutions have been put forward but none has been able to impose itself as a standard. Different optimization methods and algorithms have been tested in the past. Among them, nature-inspired algorithms, and in particular evolutionary algorithms have showed promising results (Koza et al. 1997). Its application to synthetic biology is evaluated in the Chapter 5. The work is composed of two main parts. First, we fixed a priori the topology of the GRN and evolve the model parameters in order to find the targeted behavior. Then, we use genetic programming to evolve both the network and its parameters in order to tend to a true design automation algorithm.

References

Beal, Jacob, Ting Lu, and Ron Weiss. 2011. “Automatic Compilation from High-Level Biologically-Oriented Programming Language to Genetic Regulatory Networks.” *PloS One* 6 (8): e22490. doi:10.1371/journal.pone.0022490.

Beal, Jacob, Ron Weiss, Douglas Densmore, Aaron Adler, Evan Appleton, Jonathan Babb, Swapnil Bhatia, et al. 2012. “An End-to-End Workflow for Engineering of Biological Networks from High-Level Specifications.” *ACS Synthetic Biology* 1 (8): 317–31.

Bilitchenko, Lesia, Adam Liu, Sherine Cheung, Emma Weeding, Bing Xia, Mariana Leguia, J Christopher Anderson, and Douglas Densmore. 2011. “Eugene-a Domain Specific Language for Specifying and Constraining Synthetic Biological Parts, Devices, and Systems.” *PloS One* 6 (4). Public Library of Science: e18882.

Densmore, Douglas, Joshua T. Kittleson, Lesia Bilitchenko, Adam Liu, and J. Christopher Anderson. 2010. “Rule Based Constraints for the Construction of Genetic Devices.” *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, May. Ieee, 557–60. doi:10.1109/ISCAS.2010.5537540.

Koza, John R JR, FH Forrest H Bennett, David Andre, Martin A Keane, and Frank Dunlap. 1997. “Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming.” *… IEEE Transactions on* 1 (2). IEEE: 109–28.

Marchisio, Mario A, and Jörg Stelling. 2011. “Automatic Design of Digital Synthetic Gene Circuits.” *PLoS Comput. Biol* 7 (2): e1001083.

Yaman, Fusun, Swapnil Bhatia, Aaron Adler, Douglas Densmore, and Jacob Beal. 2012. “Automated Selection of Synthetic Biology Parts for Genetic Regulatory Networks.” *ACS Synthetic Biology* 1 (8). American Chemical Society: 332–44. doi:10.1021/sb300032y.

Chapter 4
Design Automation at Boolean Level

1. Design automation in digital electronics

Automation of the design of electronic circuits from a high-level specification has been a very active area of research since the very beginning of microelectronics history in the 60s. From the outset, separated ways have been investigated for analog circuits and for digital circuits. Analog circuits are usually described by their transfer function, by analytic equations or by a set of coupled ordinary

differential equations. Despite some breakthroughs in the end of the 90's, the development of a generic fully automated analog synthesis tool is still an open field of investigation (Rutenbar 1993; Ismail and Franca 2012; Jerke and Lienig 2009; Koza et al. 1997). Up to now, CAD tools in analog context are helpful for the designer but are not automated. On the other hand, design automation of digital circuits has been a common practice since the early 80's (Rabaey, Chandrakasan, and Nikolic 2002; Micheli 1994; Brayton et al. 1996). Nowadays, several design automation tools exist and are integrated to high-performance commercial software suites (e.g. RTL Encounter of Cadence© tool suite or Design Complier for Synopsis©). They can carry out the synthesis of circuits with hundreds of thousands of logic gates. In addition several open-source tools developed by academics also exist.

Basically, design automation of digital circuits is a three-stage process that requires a formal description (high-level specification) of the system we would like to design and a toolbox (design kit) with the elementary functions (parts) that can be achieved with a given technology. The complete design flow is summarized on Fig. 1.

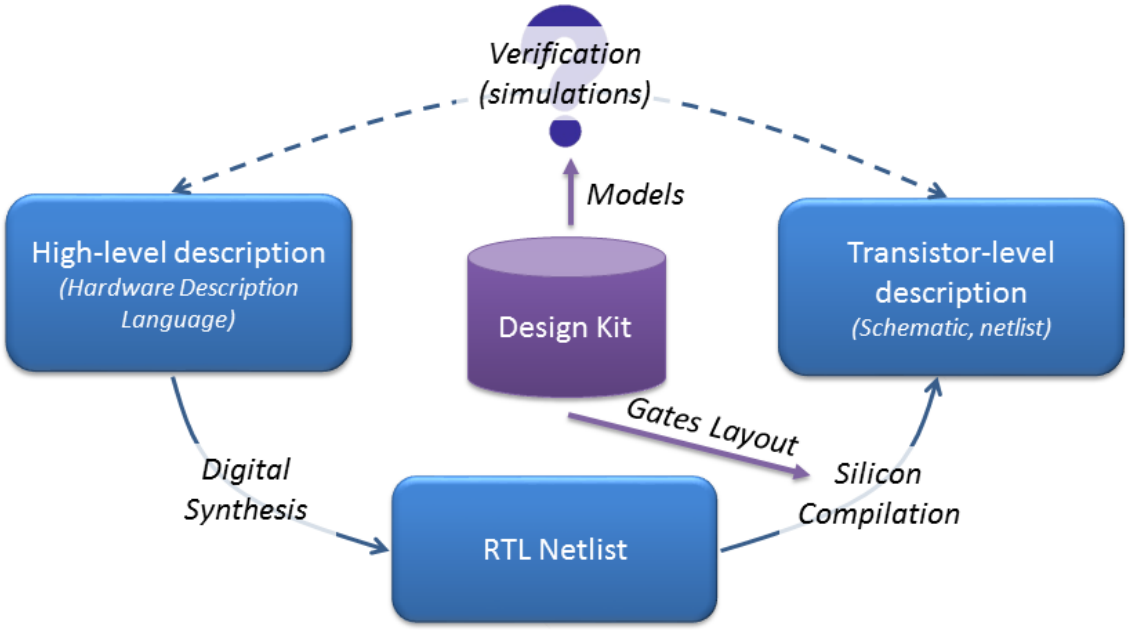


Fig. 1 Automated workflow for the design of electronic digital circuits

1.1. Digital synthesis

The first stage of the process is called digital synthesis. The high-level specification is usually given through a description written in hardware description languages (HDL) such as VHDL (Ashenden 2010) or Verilog (Thomas and Moorby 2002). At this level, the description can be purely behavioral and does not include any clue about how it will be performed. For instance, the HDL description of a counter is a memory device that is incremented at each rising edge of a clock signal.

The high-level specification is interpreted and converted into an interconnection of Boolean functions (gates) and memories (registers). This level of description is called Register Transfer Level (or RTL netlist). This step involves syntax analysis and Boolean computation.

1.2. Silicon compiler

The design kit comes into play in the second step and contains all the Boolean functions that can be achieved with a given technology. A tool named the silicon compiler finds the combination of parts from the design kit that is equivalent (from a Boolean viewpoint) to the RTL netlist while minimizing a user-defined cost function. After this step, the circuit is designed.

1.3. Back annotation and layout versus schematics

For the third step, back annotation consists in generating a model of the circuit fed with information from the design kit. This implemented version of the circuit is then simulated. The comparison between layout and schematics consists in comparing the results of this simulation to the high-level specification. Moreover, a set of design rules (size of the transistors, clearance between transistors, etc) is checked at this stage. This allows the validation *in silico* of the circuit before manufacturing.

2. Design of combinatorial GRNs

Among others, two open-source tools catch our interest because of their relative simplicity and accessibility: Odin II (Jamieson et al. 2010), which is a digital synthesizer developed by the University of Miami and ABC (Mishchenko 2015) which is a silicon compiler developed by the University of Berkeley for the configuration of FPGA (Field-Programmable Gate Array). In the following paragraph, we will describe how these tools have been reused, adapted and integrated into an automated design workflow for GRN.

2.1. Description of GeNeDA

GeNeDA stands for Gene Regulatory Network Design Automation and is described in Fig. 2. Its framework is composed of six parts: the input interface (1), the digital synthesizer (ODIN II - 2) and the GRN compiler (ABC - 3), a library of elementary mechanisms (parts) used in GRN (4) and an output interface (5). Moreover, GeNeDA provides models of the generated GRN that can be simulated by third-party analog and mixed simulator features (6). These parts are described in more details in this following section.

2.1.1. The input interface

ODIN II requires a high-level specification provided in Verilog (Thomas and Moorby 2002) or Berkeley Logic Interchange Format (BLIF) languages. The input interface bridges the gap between those specific languages and most common ways to describe a combinatorial digital system, such as a truth table or a set of Boolean equations. An online PHP script converts the specification given through a graphical user interface into a single Verilog file.

2.1.2. The digital synthesizer – Odin II

ODIN II converts the provided (or generated) Verilog file into an RTL netlist. As for microelectronics, this operation involves only syntax analysis and mathematical manipulations of Boolean equations. This operation does not depend on the technology (silicon integrated circuits or gene regulatory networks) used to perform the function. As a consequence, Odin II can be integrated directly to the framework without modification. Odin II provides the RTL netlist in a standard BLIF file.

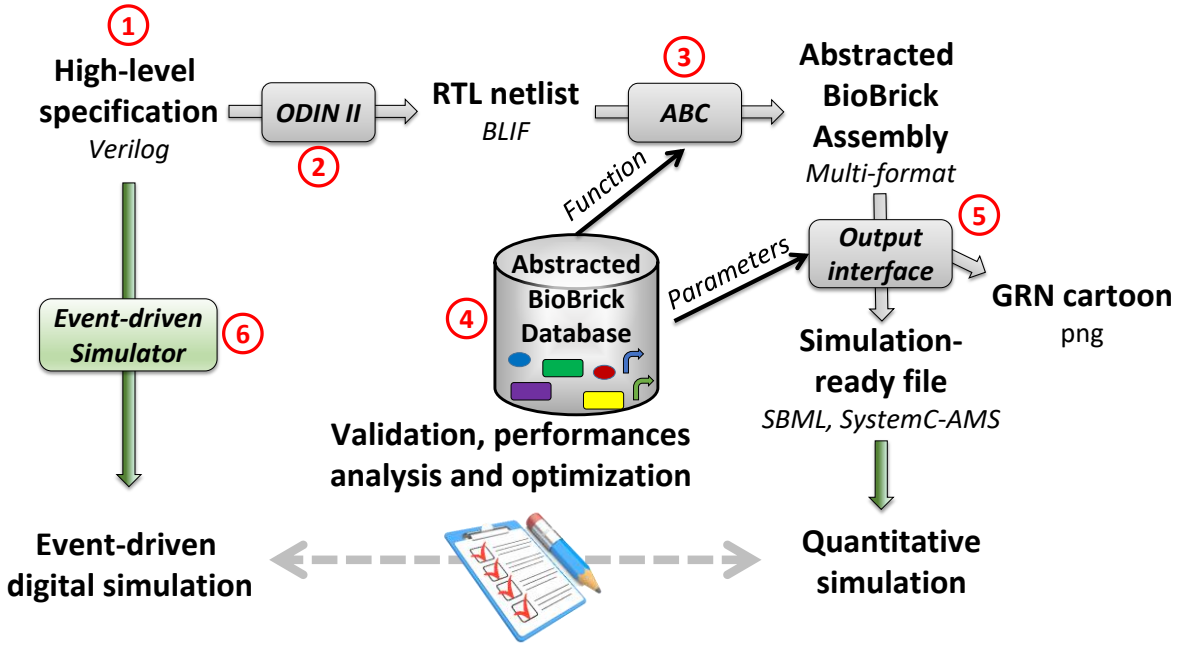


Fig. 2 GeNeDA workflow. The six parts composing the GRN synthesizer are detailed in text.

2.1.3. The GRN compiler – ABC

ABC maps the RTL netlist on a specific technology, GRN in this case. ABC uses BLIF files provided by ODIN II and is strongly linked to a Genetic Part Library (GPL) described hereafter. ABC computes a combination of GPL elements that achieves the targeted function in an optimal way. The two main performance criteria that are estimated are the cost of the system (computed as the sum of the cost for each instantiated part) and the delay of the critical path (the longest time separating an input and output changes). The results are returned as a netlist giving the instantiated parts and their connections.

Several adaptations have been made on ABC in order to fit to biology requirements. One of the trickiest issues concerns the differentiation between activators and repressors, which does not exist in microelectronics. This constraint arises from the assumption that a protein cannot carry on both the role of activator on one promoter and repressor on another one. ABC checks whether this case occurs on the suggested GRN and solves this problem by inserting a second gene (one coding for an activator and one for a repressor) on a construct for which the synthesized regulating protein have to play both roles. Should this occur on an input of the system, an additional buffer (inducible promoter without repressor) is implemented with two protein coding sequences, one for an activator and one for a repressor. This increases the complexity of the system and, as a consequence, should be taken into account in the cost of the construction. The remaining adjustments of ABC to biological context have been made directly in the genetic part library.

2.1.4. The genetic part library (GPL)

The GPL includes all the genetic mechanisms that may be involved in a GRN. The library should be written according to a proprietary GenLib format. The GenLib file contains one entry per part and for each of them the followings: the cost of the part, its Boolean function and its list of inputs. For each input, the following electrical parameters are provided: the phase (inverting or non-inverting), the input load, the max load, the rise and fall block delay and the rise and fall block fan-out. GeNeDA

involves a default library described hereafter and a GenLib generator allowing to define a subset of the default library.

Minimal library and generic library

To be consistent and usable by ABC, a GPL has to contain, at least, a Universal Set of Boolean Operators (USBO) and a synchronous memory (Paul Horowitz and Hill 1989). This memory is not required for the design of combinatorial circuits. Nevertheless ABC checks the consistency of the GPL (GenLib file) before the compilation and returns an error if the memory is missing.

In GRN context, the smallest USBO is a construct with an inducible promoter that can be activated by an activator A or repressed by a repressor R. This construct achieves the Boolean "inhibition" operator symbolized by the slash operator (see Chapter 2): $A/B = A \cdot \bar{B}$. Indeed, by cascading several instances of such constructs, any combinatorial Boolean function can be achieved. The three elementary operators of Boolean algebra NOT, AND and OR form a USBO. The following equations show how to perform NOT, AND and OR with the INH operator:

- $\bar{A} = 1/A$
- $A \cdot B = A \cdot \bar{\bar{B}} = A/(1/B)$
- $A + B = \overline{\bar{A} \cdot \bar{B}} = 1/((1/A)/B)$

However, it is also possible to enrich the GPL with more complex operators, as for instance a promoter driven by more than one activator or more than one repressor (Fig. 3A), cascaded constructs achieving advanced Boolean functions (multiplexer, encoder …), or constructs involving alternative mechanisms (see Chapter 2 for more details) such as Moon's AND gate (Moon et al. 2012) depicted in Fig. 3B. It can also be envisioned to add the operator corresponding to a promoter activated by *n* activators and repressed by *m* repressors: $(A_1 + A_2 + \cdots + A_n) \cdot \overline{R_1} \cdot \overline{R_2} \cdot \ldots \cdot \overline{R_m}$.

The default library of GeNeDA includes all the Boolean functions that can be achieved with a promoter regulated by a combination of up to four activators or repressors, the AND and NAND functions realized according to Moon's principle and a flip-flop which is not yet described.

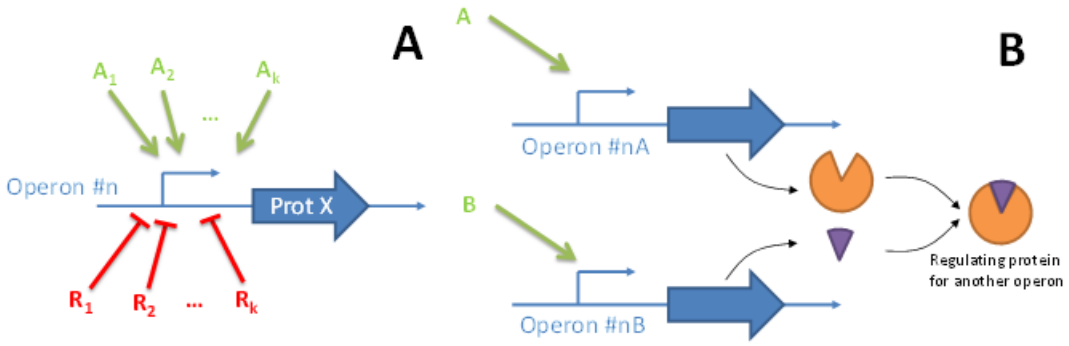


Fig. 3 Description of the BPL, which contains the basic functions that can be realized with GRN: (A) the generic combinatorial gate which consists in a promoter regulated by k transcription factors. It achieves the Boolean function A1 OR A2 OR … AND NOT R1 AND NOT R2…, (B) the AND gate which consists in two promoters and two genes, each of them synthesizing a part of a transcription factor (Moon et al. 2012).

Cost of the part

The cost of a part reflects the difficulty to construct an actual system (promoter, regulatory sequence, protein coding sequence) that uses this part. In the default library, costs have been affected to Boolean functions according to the following assumptions:

- It is more complex to add the $n+1$-th regulating protein on a promoter than adding the n-th. As a consequence, the relationship between the cost and the number of regulating proteins is a power law.
- In a general way, for a given promoter, it is easier to find a repressor than an activator. As a consequence, for an equivalent number of regulating proteins, a Boolean function involving promoter activation should be penalized in comparison with a function involving promoter repression.

The mathematical law that has been used to take these two facts into account in the cost C of each part is the following:

$$C = 1.6^{(A+R-1)} \times 1.25^{A}$$

where A and R are the numbers of activators and repressors respectively. The constants were found empirically.

Input and output parameters

For each operator, additional parameters have to be added in the GenLib file, namely the phase, the rise and fall delays, the input load and the fan out. The phase parameter is specific to electronics and is not used in a biological context. Nevertheless, we use this entry to specify the sign of the regulation (activation or repression). This information is processed by an ABC add-on we developed to tackle the problem of proteins that carry both the role of activator and repressor, as described in the previous section.

Rise and fall delays are used both for timing consideration during simulation as well as for the computation of the critical path of the system. Rise and fall delays can be defined similarly in biology and in electronics. It should be noticed that, by opposition to electronics, time constants of the biological phenomena involved in rise (gene expression) and fall processes (self or forced decay) may be very different.

An analogy can also be found for the input load and fan out which are important characteristics of digital gates. Let *X* be a regulating protein synthesized by the gene #0 and which represses (or activates) *N* other promoters. If *N* is too large, there might not be enough proteins for the regulation (repression or activation) to be effective of all the promoters. In this context, the input load can be seen as the number of proteins required to efficiently repress a promoter and the fan out as the maximum number of protein that a gene can synthesize. The GRN compiler ensures that, for each protein, the fan out is always greater than the sum of the input load of the promoter it regulates.

Current version of the GPL does not take these features into account (i.e. rise and fall time are fixed to 1 for each gate, the fan out is also 1 and the input load 0.1, allowing a transcription factor to regulate up to 10 promoters).

2.1.5. The output interface

The raw output of ABC tool is a C-structure which contains all the information about the circuit (input, output, part instances, connections…). Several add-ons have also been developed in order to provide descriptions that can be used by simulators. Up to now, 4 formats are supported:

- a BLIF file, *i.e.* a text file giving the name of the instantiated part as well as the regulation between parts

- a graphical representation of the GRN in PNG format generated by the online design visualizer Pigeon CAD (Bhatia and Densmore 2013)
- a SBML model (standard for the description in system's biology (Hucka et al. 2003)) that can be simulated with COPASI (Hoops et al. 2006)
- SystemC-AMS files (Vachoux, Grimm, and Einwich 2003), an *open source* C++ library used for the description and the simulation of heterogeneous systems such as GRNs (Pêcheux, Madec, and Lallement 2010). It is a powerful alternative simulation feature though unusual for the biologists' community. This feature manages several Models of Computation (MoC), namely non-conservative Timed Data Flow or conservative Electrical Linear Networks. With this language, the model is written in C++ and, as such, it has been designed to freely interact with other MoCs (i.e. Discrete Event of SystemC) as well as common C++ libraries (Cublas, FFT, access to databases, etc).

The SBML and the SystemC-AMS files provide a simulation-ready model based on the classical equations, described in Chapter 2.

Parameters as well as their default value for model generation are summarized in Table 1 (Alon 2006).

Table 1 Default parameters for the SBML and SystemC-AMS models.

SYMBOL	DESCRIPTION	DEFAULT VALUE
$K_{A,i,k}$	Activator-promoter binding affinity	0.01 µMol
$n_{A,i,k}$	Activator-promoter binding Hill's number	2
$K_{R,j,k}$	Repressor-promoter binding affinity	0.01 µMol
$n_{R,j,k}$	Repressor-promoter binding Hill's number	2
α_k	Promoter leakiness	0.01
$k_{tr,k}$	Transcription rate	10 µMol.s^{-1}
$k_{tl,k}$	Translation rate	1 µMol.s^{-1}
$d_{mRNA,k}$	mRNA decay rate	0.01 s^{-1}
$d_{X,k}$	Transcription factor decay rate	0.001 s^{-1}

2.2. Results on combinatorial systems

This section describes the results obtained by GeNeDA over a benchmark of standard digital circuits used in microelectronics. Eight of them (namely a 2-input AND, a 4-input AND, a 4-input NOR, a 2-input XOR, a 4-input XOR, a 1-bit half adder, a 1-bit full adder and an 8-bit comparator) are used.

To the emphasis on the importance of the GPL, five different GPL have been investigated. The simplest (GPL0) contains only one inverter (INV) and one inhibition operator (INH). GPL1 is enriched with constitutive promoters that can be inhibited by two repressors (leading to a 2-input NOR behavior or NOR2) or a promoter which can be induced by two activators (2-input OR gate, or OR2). GPL2 is enriched with 2-input AND and NAND gates designed according to Moon's construct (Moon et al. 2012). GPL3 involves GPL1's function as well as promoters that can be regulated by 3 transcription factors: 3-input OR (3 activators, OR3), 3-input NOR (3 repressors, NOR3) but also Boolean function that corresponds to 2 activators and 1 repressor (INH2A1R) and 1 activator and 2 repressors (INH1A2R). Finally, GPL4 involves all the Boolean functions that can be achieved with a promoter regulated by 4 transcription factors: OR4, NOR3, INH3A1R, INH2A2R and INH1A3R.

Table 2 Parameters of the 16 parts and as the list of GPL in which they are included.

Name	Boolean Function	#Promoter	# Regul. Proteins	# Acivators	Cost	Delay	GPL0	GPL1	GPL2	GPL3	GPL4
BUF	S=a	1	1	1	1.25	1	●	●	●	●	●
INV	S=!x	1	1	0	1.00	1	●	●	●	●	●
OR	S=a+b	1	2	2	2.50	1		●	●	●	●
INH	S=a+!x	1	2	1	2.00	1	●	●	●	●	●
NOR	S=!(x+y)	1	2	0	1.60	1		●	●	●	●
OR3	S=a+b+c	1	3	3	5.00	1				●	●
INH2A1R	S=(a+b)&!x	1	3	2	4.00	1				●	●
INH1A2R	S=a&!(x+y)	1	3	1	3.20	1				●	●
NOR3	S=!(x+y+z)	1	3	0	2.56	1				●	●
OR4	S=a+b+c+d	1	4	4	10.00	1					●
INH3A1R	S=a&!(x+y+z)	1	4	3	8.00	1					●
INH2A2R	S=(a+b)&!(x+y)	1	4	2	6.40	1					●
INH1A3R	S=a&!(x+y+z)	1	4	1	5.12	1					●
NOR4	S=!(a+b+c+d)	1	4	0	4.10	1					●
AND	S=a&b	2	2	2	5.00	1			●		
NAND	S=!(a&b)	2	2	0	4.00	1			●		

Table 3 Synthesis results for different functions with different GPLs. The tables display the gates instantiated for each library, the corresponding cost of the generated system, the number of promoters, the number of internal regulations and the maximum number of genes between the input and the output, named delay.

2-input AND gate

Library	Instantiated Gates																Cost	Promoters	Internal Regulation	Delay
	BUF	INV	OR	INH	NOR	OR3	INH2A1R	INH1A2R	NOR3	OR4	INH3A1R	INH2A2R	INH1A3R	NOR4	AND	NAND				
BIOLIB0		1		1													3.00	2	1	2
GPL1		1		1													3.00	2	1	2
GPL2		1		1													3.00	2	1	2
GPL3		1		1													3.00	2	1	2
GPL4		1		1													3.00	2	1	2

4-input AND gate

Library	Instantiated Gates															Cost	Promoters	Internal Regulation	Delay
	BUF	INV	OR	INH	NOR	OR3	INH2A1R	INH1A2R	NOR3	OR4	INH3A1R	INH2A2R	INH1A3R	NOR4	AND	NAND				
GPL0		3		3													9.00	6	5	4
GPL1		3	1	2													9.50	6	5	3
GPL2		3	1	2													9.50	6	5	3
GPL3		3		1				1									8.20	5	4	2
GPL4		1												1			8.10	5	4	2

4-input NOR gate

Library	Instantiated Gates															Cost	Promoters	Internal Regulation	Delay
	BUF	INV	OR	INH	NOR	OR3	INH2A1R	INH1A2R	NOR3	OR4	INH3A1R	INH2A2R	INH1A3R	NOR4	AND	NAND				
GPL0		3		3													9.00	6	5	4
GPL1			1	1	1												6.10	3	2	2
GPL2			1	1	1												6.10	3	2	2
GPL3					1			1									4.80	2	1	2
GPL4															1		4.10	1	0	1

2-input XOR gate

Library	Instantiated Gates															Cost	Promoters	Internal Regulation	Delay
	BUF	INV	OR	INH	NOR	OR3	INH2A1R	INH1A2R	NOR3	OR4	INH3A1R	INH2A2R	INH1A3R	NOR4	AND	NAND				
GPL0	2	2		3													10.50	7	8	5
GPL1	2		1	2													9.00	5	6	2
GPL2	2		1	2													9.00	5	6	2
GPL3	2		1	2													9.00	5	6	2
GPL4	2		1	2													9.00	5	6	2

4-input XOR gate

Library	Instantiated Gates															Cost	Promoters	Internal Regulation	Delay
	BUF	INV	OR	INH	NOR	OR3	INH2A1R	INH1A2R	NOR3	OR4	INH3A1R	INH2A2R	INH1A3R	NOR4	AND	NAND				
GPL0	2	6		9													26.50	17	21	11
GPL1	2	2	1	4	4												21.40	13	14	7
GPL2	2	2	1	4	4												21.40	13	14	7
GPL3	4	2	1				2	2									23.40	11	18	5
GPL4	4	2	1				2	2									23.40	11	18	5

1-bit half-adder

Library	Instantiated Gates															Cost	Promoters	Internal Regulation	Delay
	BUF	INV	OR	INH	NOR	OR3	INH2A1R	INH1A2R	NOR3	OR4	INH3A1R	INH2A2R	INH1A3R	NOR4	AND	NAND				
GPL0	2	3		4													13.50	9	9	5
GPL1	2	1	1	3													11.0	7	7	3
GPL2	2	1	1	3													11.0	7	7	3
GPL3	2	1	1	3													11.0	7	7	3
GPL4	2	1	1	3													11.0	7	7	3

1-bit full adder

Library	Instantiated Gates															Cost	Promoters	Internal Regulation	Delay
	BUF	INV	OR	INH	NOR	OR3	INH2A1R	INH1A2R	NOR3	OR4	INH3A1R	INH2A2R	INH1A3R	NOR4	AND	NAND				
GPL0	2	7	11														31.50	20	20	8
GPL1	2	3	3	3	5												26.50	17	18	5
GPL2	2	3	3	3	5												26.50	17	18	5
GPL3	3	3	1	2	3		1	1	1								27.81	15	18	4
GPL4	3	3	1	2	3		1	1	1								27.81	15	18	4

8-bit comparator

Library	Instantiated Gates															Cost	Promoters	Internal Regulation	Delay
	BUF	INV	OR	INH	NOR	OR3	INH2A1R	INH1A2R	NOR3	OR4	INH3A1R	INH2A2R	INH1A3R	NOR4	AND	NAND				
GPL0	16	33		75													203.00	121	157	15
GPL1	15		20	14	41												159.60	90	90	11
GPL2	15		20	14	41												159.60	90	90	11
GPL3	4	16	2		33			14	9								146.67	78	81	9
GPL4	2	16	8		28			2	3	2		6	2	7			194.70	76	73	7

We discuss the results obtained with the 4-input AND gate in particular. The GRNs obtained for this gate with different libraries are depicted in Fig. 4 and the instantiated parts and the GRN performances are shown in Table 3, 4-inputs AND gate. In GPL0, the synthesizer uses only an inverter (INV), which is a constitutive promoter that can be repressed and an inhibition gate (INH). The results involve 6 promoters and a delay of 4 (i.e. the critical path, D-R2-R3-R4-GFP, involves 4 promoters, each of them introducing a delay of 1). In another words, when D switches from one state to another, the expression of 4 promoters have to change sequentially in order to observe the switch of GFP at the output. With GLP1, the introduction of 2-activators and 2-repressors promoters reduces the critical path down to 6, despite an extra cost of 6%. With GPL3, a 1-activator 2-repressor promoter can be used instead of one OR and one INH. By this means, the system improves both characteristics: the delay is reduced to 2

and the cost to 8.2. Finally, the 4-input NOR gate of GPL4 makes the system less expensive. This last solution corresponds to the one that can be obtained theoretically by applying de Morgan's theorem: $A \cdot B \cdot C \cdot D = \overline{\bar{A} + \bar{B} + \bar{C} + \bar{D}}$.

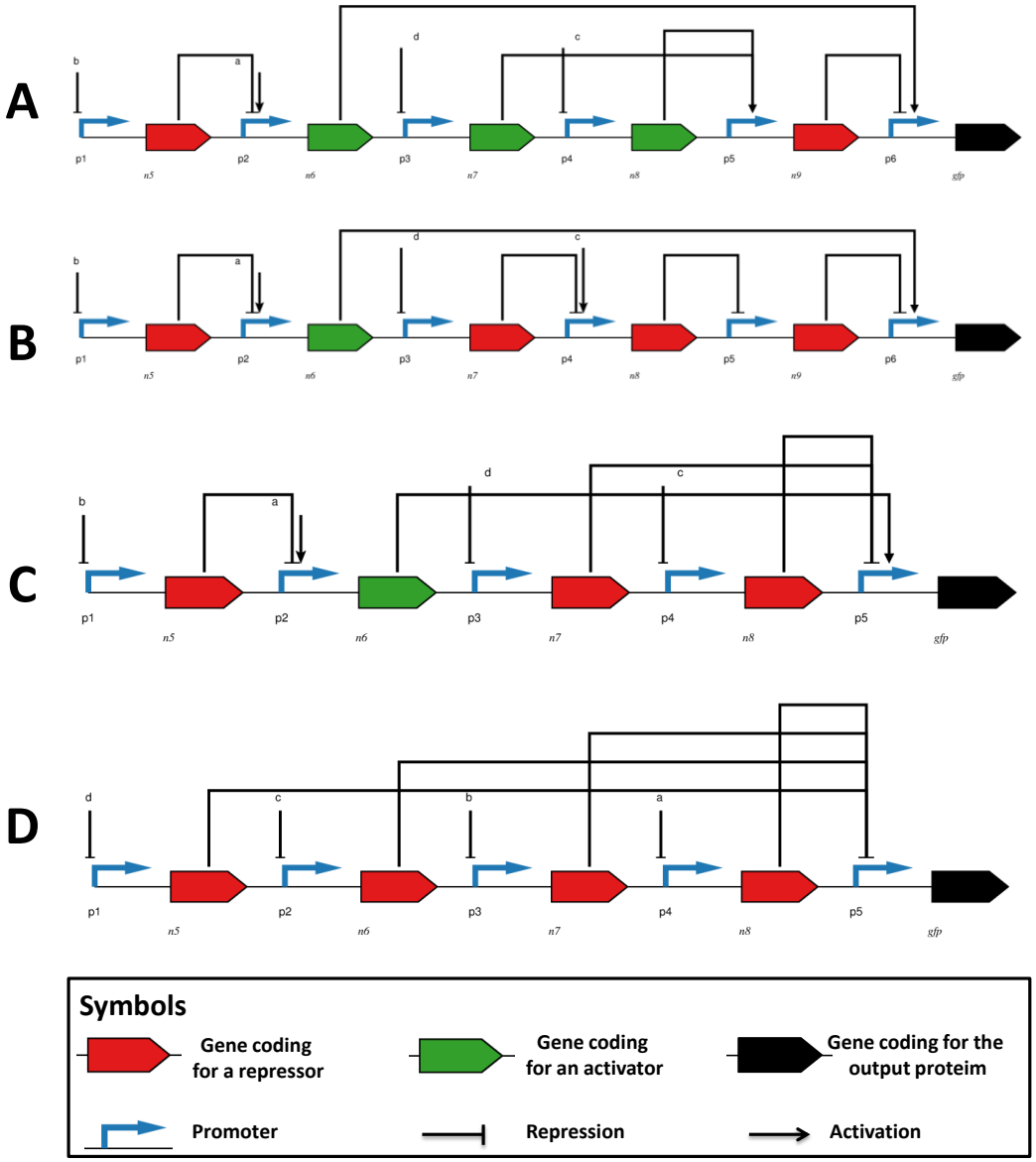


Fig. 4 Results of the synthesis of a 4-input AND gate with 4 different GPL. (A) GPL0, (B) GPL1, (C) GPL3 and (D) GPL4.

The main conclusions of this study on combinatorial circuits that deserve to be highlighted are the following:

- As expected, the complex (and thus expensive) parts added into library GPL3 and GPL4 are used sparingly in the designs.
- The AND gate designed by Moon's construct is never used by the synthesizer. This is explained by the cost affected to this construct (4.0) which is too high in comparison with other construct.
- For the very simple Boolean function, the use of complex parts on the same promoter does not bring remarkable improvement with respect to the cost or the circuit delay.
- For more complex functions, GPL3 and GPL4 lead to important cost or delay reductions. For instance, 25% of cost and delay reduction for the 8-bit comparator between GPL1 and GPL4.

3. Design of sequential GRNs

3.1. Definition of a sequential system

By opposition to combinatorial circuits, the output of the sequential circuits depends both on the combination of inputs and on the state of the circuit. The state of the circuit itself depends on previous values of inputs and is stored in internal memories as a signal. As depicted in Fig. 5 the circuit performing sequential function is divided into two parts: the output logic and the transition logic. The output logic is given by the value of the outputs S_i as a function of the inputs E_i and the state encoded in several signals X_i . The transition logic gives the next values of X_i as a function of the current ones and the inputs.

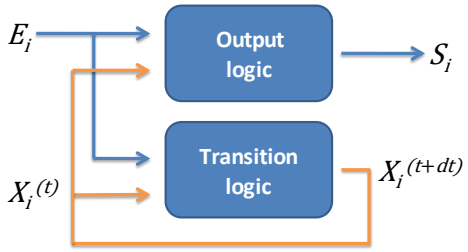


Fig. 5 Diagram of a sequential function. Ei and Si are respectively the inputs and the outputs of the system. Xi are the internal variables of the system ; their values depend on time t.

Since a given input combination can lead to different outputs, the representation of a sequential system by a truth table is no longer possible. Therefore more complex representations such as flow graphs are required. An example of a flow graph is given in Fig. 6. It corresponds to a *rendez-vous* gate. This example is used all along this subsection to illustrate Huffman's method. It is composed of states (bubbles) defined by a combination of inputs and outputs, and transitions (arrows) between states. A transition between two states can only be triggered by the change of one and only one input. Thus, in this flow graph, it can be read that when the system is in the state S1, the two inputs and the output are "0". If the input A rises to "1", the system goes into the state S3 and the output remains at "0". If A falls back to "0", the system returns to state S1. Otherwise, if the system is in state S2 and A rises to "1", the system goes in state S4 and, in this case, the output rises to "1".

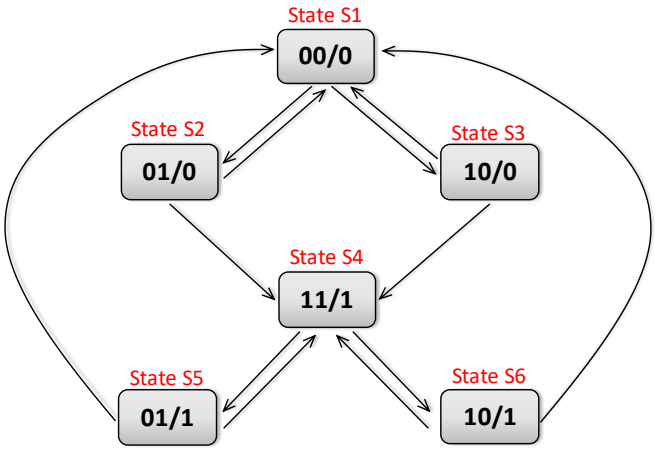


Fig. 6 Flow graph of the *rendez-vous* gate. The bubbles represent the different states with the values of the inputs A and B and output O corresponding to the side inside (AB/O).

This flow graph can also be represented as a transition table (Table 4). The header line corresponds to the inputs values. Then, each line corresponds to a state. The state number is written under its

corresponding combination (bold and gray values) whereas the transitions (italic) are written below the combination of entries that corresponds to the targeted state.

Table 4 Transistion table of the rendez-vous gate. Each line is a state, with in grey the cell corresponding to the input combination of this state. For each possible transition (one of the two inputs varies), the next state is written (not in bold) in the same line.

Inputs → State↓	**0 0**	**0 1**	**1 1**	**1 0**
S1	**S1**	S2		S3
S2	S1	**S2**	S4	
S3	S1		S4	**S3**
S4		S5	**S4**	S6
S5	S1	**S5**	S4	
S6	S1		S4	**S6**

3.2. Design of a sequential system

As depicted on Fig. 5, the sequential system is composed of two parts and involves internal signals that encode the states. For a given flow graph, there are several synthesis methods for computing a circuit at the gate-level. For sequential asynchronous systems the most common one is Huffmann's method (Giovanni De Micheli 1994).

The first stage of the design of a sequential system is to set the number of internal signals as well as the way each state is encoded. It should be noted that two states can be encoded in the same way as soon as they can be distinguished from each other with their inputs combination. The minimal number of internal variables required to encode all the states is given by a compressed form of the transition table. This form is obtained by grouping together the lines that do not exhibit contradictions (contradiction occurs when two lines have two different state number under the same input combination). In the example of Table 4, the first 3 lines can be grouped into one, as well as the last three. This leads to the compressed transition table on Table 5.

Table 5 Compressed transition table of a *rendez-vous* gate. The first row corresponds to the inputs. Inside the cell is reported the state number.

0 0	**0 1**	**1 1**	**1 0**
S1	**S2**	S4	**S3**
S1	**S5**	**S4**	**S6**

The second stage consists in choosing the way each line is encoded. Here, it is quite intuitive to encode states 1 to 3 with the interval variable X=0 and states 4 to 6 with X=1. For systems with a single internal variable, the way the states are encoded does not matter. Conversely, when there is more than one internal variable, the encoding has to be chosen such that a transition only induces a change in a single internal variable. This improves the stability of the system (see next section).

The third stage of the design consists in computing the Boolean equation of the transition logic. To do that, we replace in Table 5 each state number by its corresponding code. We obtain Table 6, which is a Karnaugh map (alternative representation of a truth table).

Table 6 Karnaugh map of the internal variable X in function of the inputs. Inside the cell is reported the future value of X. Grey cells correspond to stable states.

Inputs → Int. Variables (X) ↓	0 0	0 1	1 1	1 0
0	**0**	**0**	1	**0**
1	0	**1**	**1**	**1**

The Boolean equation corresponding to this Karnaugh map is:

Equation 1

$$X^{(t+dt)} = A \cdot B + X^{(t)} \cdot (A + B)$$

The last stage consists in computing the output. Again, from the compressed transition table, we replace the state number by the value of the output corresponding to the state. In our example, we obtain the same Karnaugh map. Thus, we can write directly that $S = X^{(t+dt)}$.

The gate-level circuit that corresponds to the function is given in Fig. 7. Green signal is the internal variable that realizes the feedback loop. As the transition logic and the output logic are combinatorial function, the GRN that performs these functions can be computed with GeNeDA (see Fig. 8 in the next paragraph).

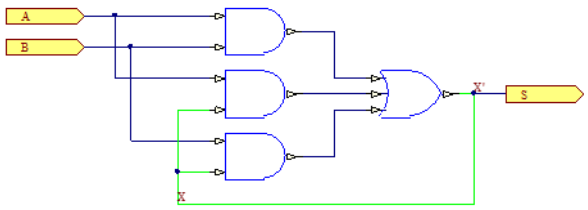


Fig. 7 Gate-level circuit of a *rendez-vous* gate.

3.3. Stability of sequential system

Feedback loops are unavoidable in sequential systems and, as in many other systems, can be a source of instability and malfunction. Here, the problem lies in the short-lived changes, also known as glitches. In electronics, glitches corresponds to delays introduced by the propagation time of signals inside gates. In biology, it corresponds to the gene activation/inhibition during transitions of the system and in the delays induced by the genetic mechanisms used in GRN. As an example, let's consider the GRN described here above. The transition from state 1 to state 2 occurs when B rises to “1” whereas A is “0”. According to the theory, the internal variable should stay at “0”. But a production of X may occur during the transition. Even if this production is weak, it can be sufficient to induce a self-activation of the corresponding gene, instead of going to state 2, the system goes into state 5, which is a malfunction.

Obviously, this phenomenon is more likely to occur when the number of internal variables and the coupling between internal variables increases. The risks also increase dramatically when several internal variables have to change at the same time, because delays induced by the genetic mechanisms are very hard to control. Consider a transition for which two internal variables have to rise from 0 to 1. If a delay occurs between the transition of the first and the second internal variable, the ghost states "10" appears and can lead the system into a unexpected state (other than "11") or can trigger oscillations. Thus, multiple changes of internal variables during a transition are generally avoided during the design process. The same rule prevails in electronics, although it is easier to control delays.

This question of the stability of an asynchronous sequential GRN is addressed in the following. MATLAB simulations are carried out on two examples: the *rendez-vous* gate, which corresponds to the system used to illustrate this section and a crossing system that requires two internal variables.

3.4. Results

3.4.1. Rendez-vous gate

GRN implementation

We used GeNeDA to compute the GRN corresponding to a *rendez-vous* gate. The expression of $X^{(t+dt)}$ given above (Equation 1) correspond to the "Transition logic" on Fig. 5 and is a combinatorial circuit (with the system's inputs and the current state as inputs and the next state as output). Thus, the GRN that corresponds to the output logic can be computed by GeNeDA. By the same way, GeNeDA can also be used to compute the ouput logic.

We obtain a GRN with 5 promoters and 5 repressors, including X which plays the role of the internal variable. Input A should be buffered (it is both an activator and a repressor) but for clarity we do not represent it here. The buffer is therefore also absent from the simulated system. The internal variable X and input B are both repressors here.

MATLAB simulations

A model of the GRN represented in Fig. 8 and based on the equations given in Chapter 2 is established with MATLAB and simulated on different test benches. Parameters are set to the values given in Table 7.

Table 7 Default values of the parameters used in simulations

Constants		Value
Dissociation constant	K_a	3.3 µmol/L
	K_r	3.3 µmol/L
Hill's number	n_a	2.5
	n_r	2.5
Transcription rate	K_{tr}	0.1 /s
Translation rate	K_{tl}	1 /s
Protein degradation	d_X	0.001 /s
mRNA degradation	d_{mRNA}	0.01 /s

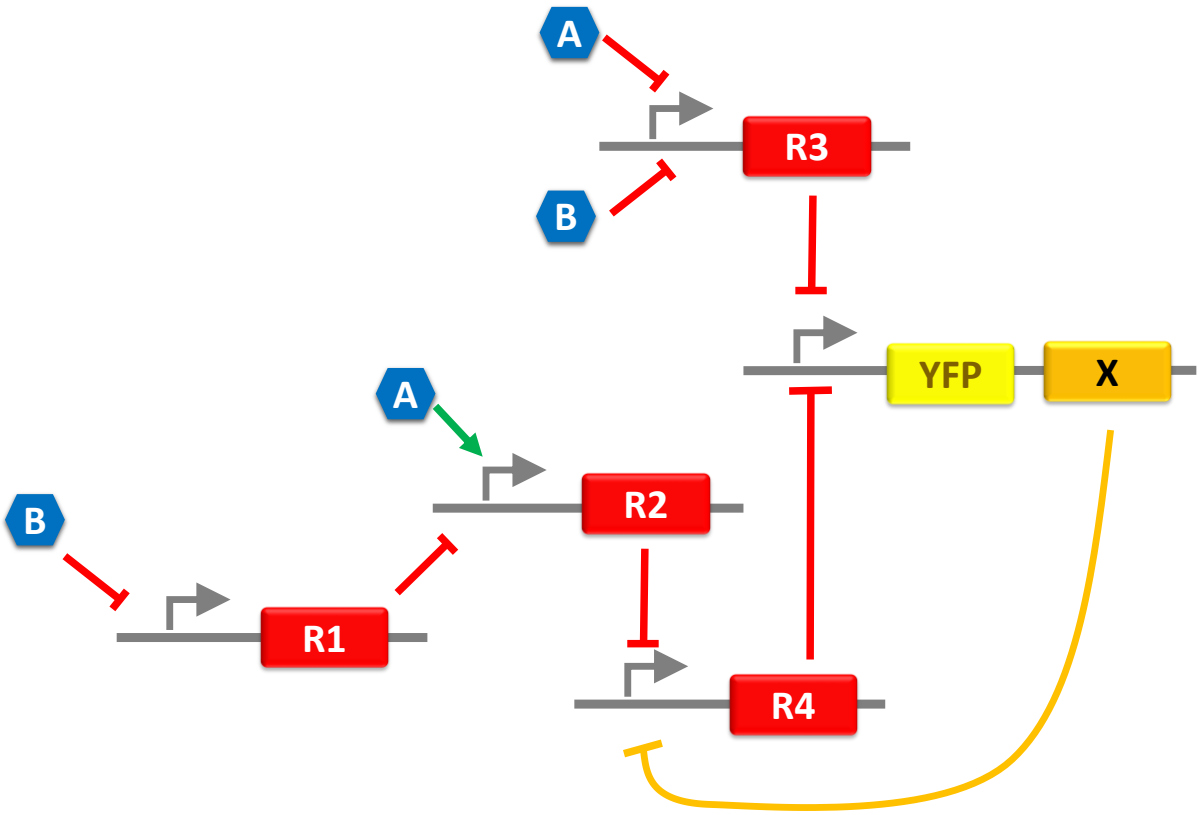


Fig. 8 GRN generated by GeNeDA to perform a *rendez-vous* gate

A test bench that cover all the possible paths in the flow graph is given is shown on Fig. 9. Simulation results are in accordance with the specifications of the system. We notably see that output is not produced before the first occurrence of both inputs A and B being high (concentration at 1) at the same time. The output is then returning to a low state only when both inputs are low at the same time.

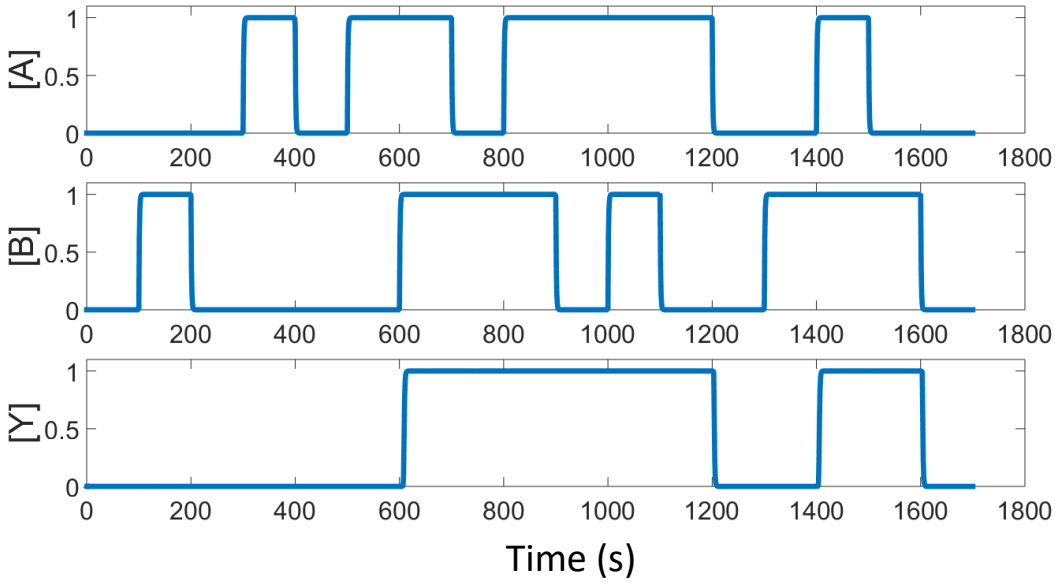


Fig. 9 MATLAB simulation of the *rendez-vous* gate

3.4.2. Crossing system

Presentation

The second system is more complicated. Its flow graph is represented of Fig. 10. The crossing system can be specified as following. It is composed of two inputs A and B and one output S. In the initial state S1, both the inputs and the output are low. When an input rises, the output rises and remains high until the other input rises and falls. The rise of the second input may occur before or after the fall of the first input.

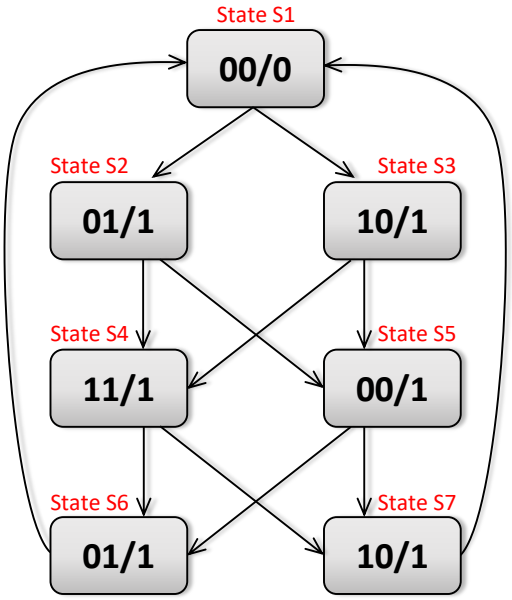


Fig. 10 Flow graph of the crossing function

The synthesis of such system requires two internal signals, Y and Z. Boolean equations giving the next state of Y and Z as a function of the actual ones and the inputs are:

$$Y^{(t+dt)} = Z^{(t)} \cdot (\bar{A} \cdot \bar{B} + A \cdot B) + Y^{(t)} \cdot (\bar{A} \cdot B + A \cdot \bar{B})$$
$$Z^{(t+dt)} = Z^{(t)} \cdot (\bar{A} \cdot \bar{B} + A \cdot B) + \overline{Y^{(t)}} \cdot (\bar{A} \cdot B + A \cdot \bar{B})$$

The output is then given by:

$$S = A + B + Z^{(t)}$$

MATLAB simulation

The GRN corresponding to this system involves 29 genes leading to 60 differential equations. A test bench that covers the 4 relevant paths in the flow graph is given (Fig. 11). The output signal is in accordance with the flow graph. As soon as one of the two inputs is high, if the output was low, it is produced. The output then stays in a high state until the other output is low after having been high. However, this accordance leans on a good choice of the biological parameters of the regulators. This will be discussed in the next section.

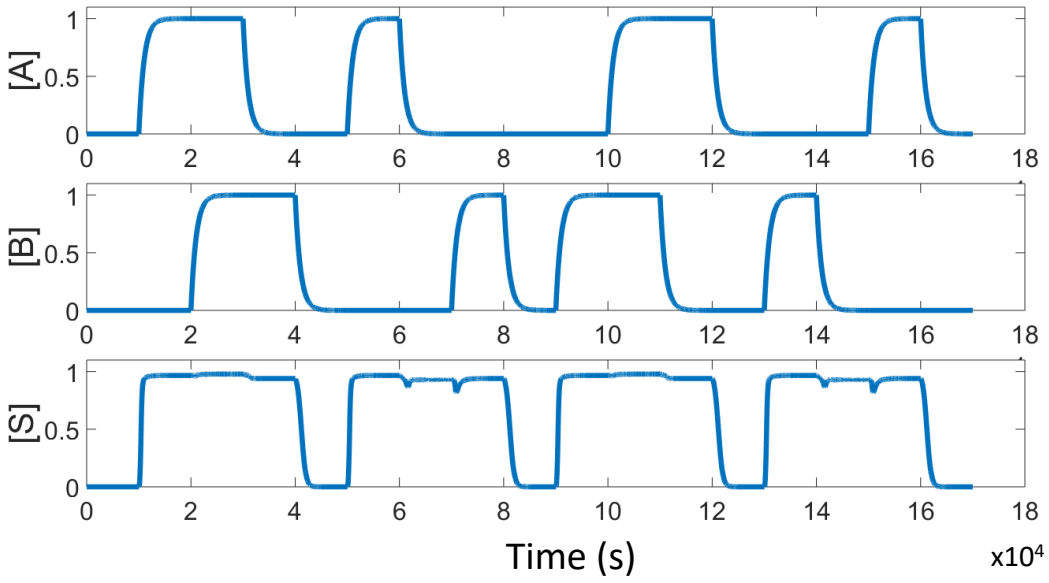


Fig. 11 MATLAB simulation of the crossing gate

3.4.3. Sensibility of the GRN parametrization

As mentioned, the parameters are set to the values given in Table 7. They are identical for each regulator.

In this paragraph, we investigate designed GRNs' sensibility towards the dispersion of the parameters associated to the regulators. These parameters are related to the affinity of the regulator to its promoter and the cooperativity of this binding, i.e. respectively the dissociation constants K_A and K_R and Hill's numbers n_A and n_R. To that end, these parameters are allowed to vary around their default value. Let Λ be the vector containing these parameters. A test consists in 100 random draws $(\Lambda_1, \dots, \Lambda_{100})$ in which each set is computed as following: $\Lambda_k = \Lambda_0 \cdot 10^{\sigma \cdot \Psi_k}$ for the dissociation constants and $\Lambda_k = \Lambda_0 \cdot (1 + \sigma \cdot \Psi_k)$ for the Hill's number, where Λ_0 is the default value of the parameter, Ψ_k is randomly drawn with a standard normal distribution and σ is the standard-deviation of the spread of the parameters. Results were compared to the expected ones (Fig. 9 and Fig. 11).

The success rate corresponds to the percentage of parameters sets producing the expected response: a threshold is fixed for the differentiation between a high and a low level. The success rate for different configurations as a function of σ (relative standard deviation of the perturbation given in %) is shown on Fig. 12. We observe that both systems are stable towards a dispersion of up to 10%. Above this value, malfunctions appear. As expected, the system with a double feedback loop (crossing system) is more sensitive than the system with a single one.

Biological noise, i.e. temporal fluctuation of the reaction rates around their deterministic value, was also investigated: we tested the influence of the variation of the species concentration on the performance of the circuits. Here, a Gaussian noise is added on each production and each degradation term of the differential equations of the system. The standard deviation of the noise is proportional to the each production and each degradation term to model a Poisson process.

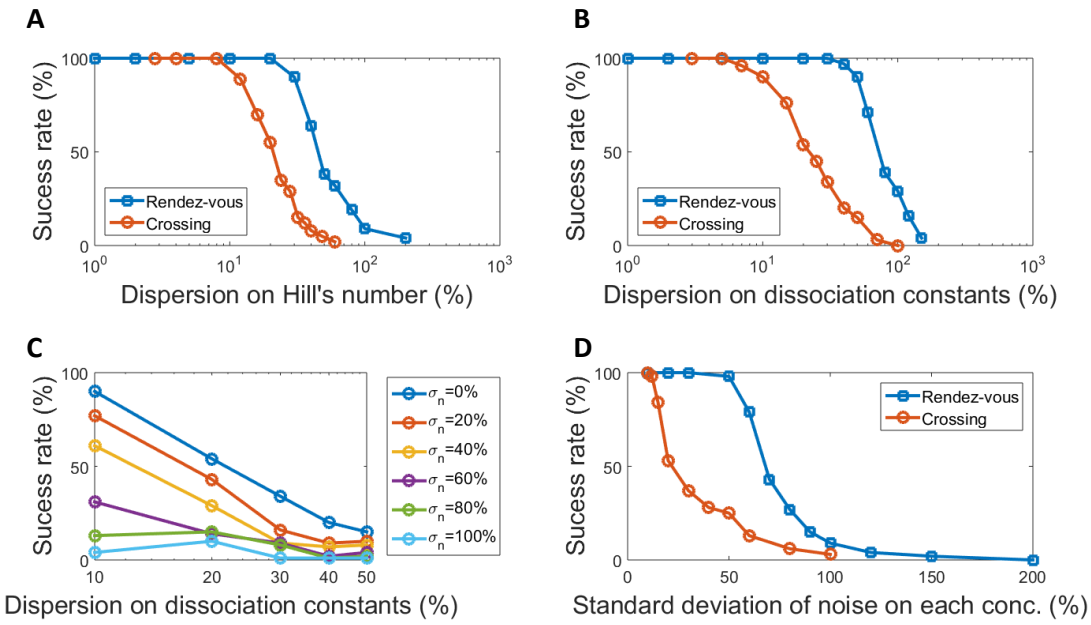


Fig. 12 Success rate for different perturbations. A and B: a fixed fluctuation is applied on parameters related to the regulators. C: success rate in function of the dispersion applied to the affinity constant, for different values of dispersion of Hill's number. D: a variation is added to the flux of species (production and degradation) in form of a Gaussian noise with a fixed standard deviation.

The last graph shows that the *rendez-vous* gate is very robust. Indeed this system is robust towards a noise relative standard deviation of over 50%. By opposition, the crossing system with the double feedback loop is much more sensitive. Malfunction appears above 7%.

The results show that feedback loops in a GRN reduces robustness towards biological noise in particular. We will therefore seek a solution to these malfunctions.

3.5. Synchronization of sequential GRN

A deeper analysis of malfunction in asynchronous sequential systems highlights that issues arise from the continued update of the internal variables. Indeed, calculated internal variables are fed back permanently to the input of the system, even if they are not yet stable or if the computation of their new value is still in progress. Thus, a solution to avoid malfunction could be to update the values of internal variable only at specific times, when we are sure that they are stable and consistent with the theory. The transition between states becomes synchronous.

The design of these circuits follows the same process as for asynchronous sequential systems with the addition of synchronization devices on feed-back loops. These devices, often called D-Flip-Flops (DFF), are memories that can be updated only on edges of a synchronization signal, namely the clock. Three conditions have to be met in order to ensure the proper functioning of synchronous systems:

- a DFF have to be added in the feed-back loop for each internal variable,
- the DFF have to share the same clock signal and the signal have to be distributed without delay to all the DFF,
- the interval between two consecutive clock edges have to be large enough to let the internal variable to update and stabilize.

Thus, the realization of synchronous systems with GRN implies first the ability to realize a GRN that mimics the behaviors of electronic DFF and second the ability to generate a clock signal and to distribute it synchronously to all the instantiated DFF. The first challenge is discussed hereafter. The second one is discussed in the third part of the manuscript.

4. Design of a biological D-Flip-Flop

The design of a biological synchronous memory is a tricky challenge. The first artificial biosystems exhibiting a non-combinatorial feature are oscillators which switch from a state to another at defined regular time steps (Elowitz and Leibler 2000) and a toggle switch (Gardner, Cantor, and Collins 2000) in 2000. Alternative constructs have also been described, most of them being based on a positive feedback loop in order to maintain state without external stimulus (Chang et al. 2010; Becskei, Séraphin, and Serrano 2001). All of these solutions are asynchronous (the memory can be updated at any time) and cannot be used in the GPL. In 2012, Hoteit *et al.* suggested a GRN achieving a D-flip-flop behavior (Hoteit, Kharma, and Varin 2012a) (Fig. 13). The solution is very expensive (7 promoters and about 10 involved proteins) and has been designed upon the standard structure used in microelectronics, *i.e.* two D-latches cascaded in a master-slave structure (Paul Horowitz and Hill 1989). In their DFF, light acts as the clock signal by modulation of the effectiveness of a transcription factor. Note that equivalent behavior has also been reached using non-genetic systems (enzymatic reactions) (MacVittie, Halámek, and Katz 2012).

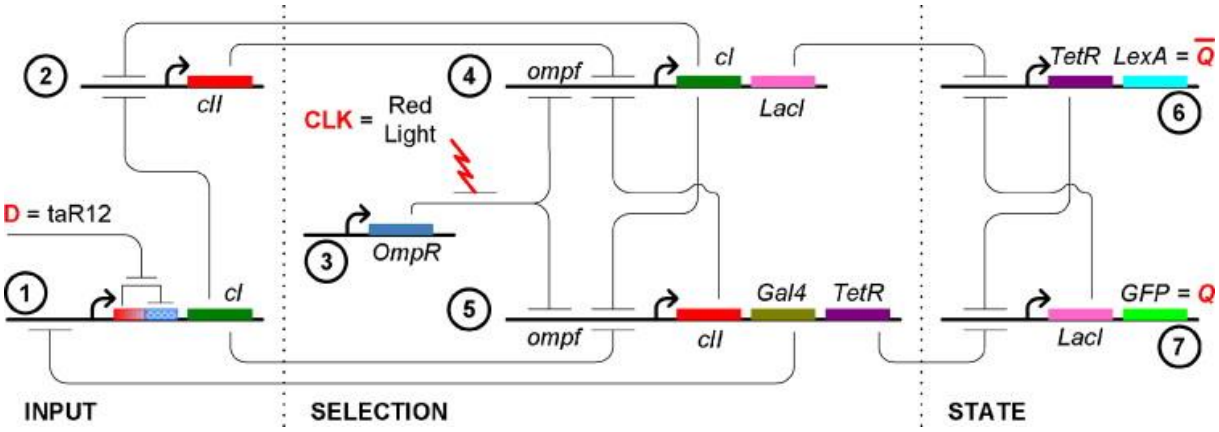


Fig. 13 The gene regulatory network of the BioD, a biological DFF with 7 promoters (Hoteit, Kharma, and Varin 2012b).

We propose a more compact implementation using only 3 promoters and a competitive cross-repression presented in Fig. 14. In the next paragraphs, we describe the system and exemplify how it can be used to construct sequential systems.

4.1. Description

A DFF is sensitive to two signals, namely Data (*D*) and Clock (*Clk*). By opposition with standard memories, the output of the DFF (*Q*), which should be bistable, may only change after rising (or falling) edges of the clock signal and according to the value of *D*: the output copies the state of D and memorizes it. Hence if *D* is high during the falling edge, *Q* turns (or stays) high whereas if *D* is low *Q* turns (or stays) low (Table 8). In the biological domain, a falling edge is a sharp decrease in the concentration of a protein (by enhanced degradation or by inhibition).

Table 8 Behavior of a flip-flop

Clk (PreQ1)	D (AHL)	Output (GFP)
L	H/L	$Output_{prev}$
↓	L	L
↓	H	H

The structure of the biological DFF (BioDFF) we developed for the bacteria *Escherichia coli* is composed of 3 operons and involves 8 proteins and molecules. In addition, the system possesses two input signals and one reporter (GFP in this case). In this example, the *D* input can be 3-oxo-C12-homoserine lactone (3OC12HSL, thereafter termed AHL), which activates the LasR protein, which can in turn bind the *luxI* promoter and activate expression of operon #1. The oscillating *Clk* signal can be realized with pre-queuosine$_1$ (PreQ1) and the adequate riboswitch (Winkler and Breaker 2005). The corresponding cartoon is given in Fig. 14.

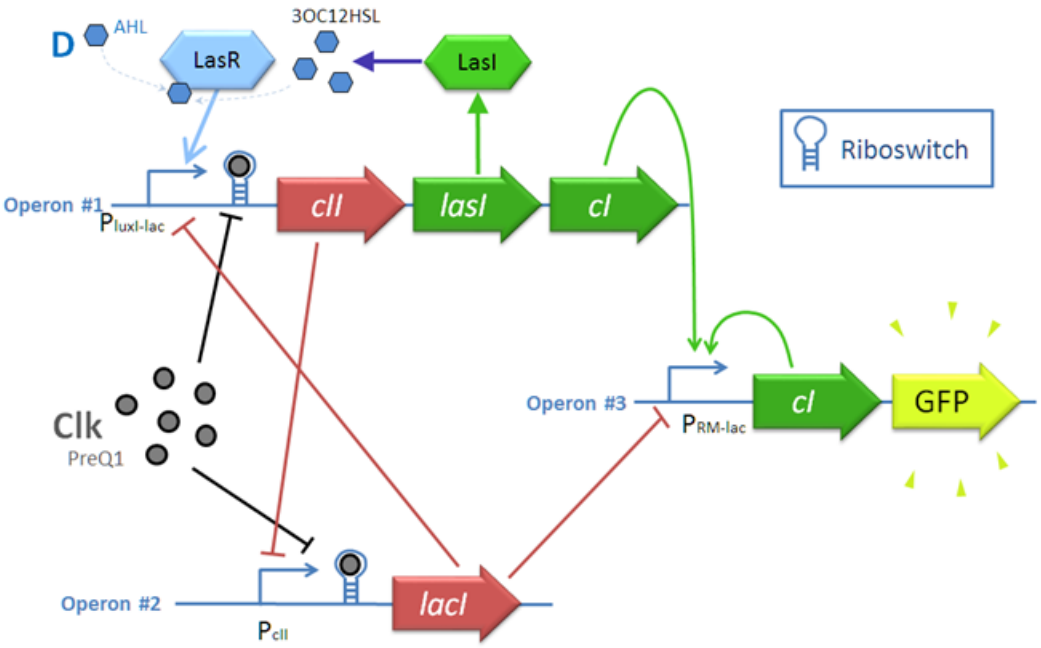


Fig. 14 A biologic D-flip-flop.

The operon #1 uses the *luxI* promoter (P_{luxI}). Three proteins are synthesized when this operon is expressed: a repressor (CII), an enzyme (LasI) and an activator (CI). The enzyme LasI produces AHL (3OC12HSL). Upon binding with AHL, the protein LasR (expressed constitutively on a separate operon) activates the transcription of the *luxI* promoter on operon #1 (Gray et al. 1994). LasI therefore plays the role of a self-activator. Direct addition of AHL, here Data signal, achieves the same effect. CII is a cross-repressor for operon #2 (Hoteit, Kharma, and Varin 2012b). The phage λ regulator protein CI activates the transcription of operon #3 by binding to the O_R domains of its P_{RM} promoter (Court, Oppenheim, and Adhya 2007). The transcript of operon #1 contains a so-called riboswitch, a short RNA sequence located on the mRNA and sensitive to the presence of a specific ligand (PreQ1 in our system). Upon addition of the ligand, formation of a premature terminator structure on the mRNA stops the

transcription process (Gong et al. 2012): the ligand acts as a repressor of the genes located downstream of the riboswitch. The promoter of operon #1 also contains a LacI binding site. Hence, its expression is turned down in the presence of the repressor LacI. The expression of the operon #1 is controlled by the concentration of AHL, from the direct input or from the enzyme LasI, the molecule PreQ1 and LacI synthesized by the operon #2. The operon #2 synthesizes LacI, a repressor inhibiting the expression of operons #1 and #3. It has the same riboswitch sequence as operon #1 and is therefore also inhibited by PreQ1. Furthermore, operon #2 is sensitive to the repressor CII synthesized by operon #1. The operon #3 has a positive linear feedback construct (Gardner, Cantor, and Collins 2000), that is to say that it synthesizes a self-activator CI. Expression of the operon #3 is controlled by two regulating proteins: an activator synthesized by operon #1 and a repressor synthesized by operon #2.

The most obvious case is when PreQ1 is present. Operons #1 and #2 are repressed by PreQ1. As a consequence, the operon #3 is "insulated" from the rest of the circuit and acts as a bistable memory: if active, this activity is maintained by the synthesis of CI. If not active, there is no activator, thus it remains inactive. Now, let us consider the case where PreQ1 drops from high to low. At this point we have to distinguish between two cases. If AHL is high, the activity of both operons #1 and #2 increases: of operon #1 because of AHL and of operon #2 because of its constitutive promoter which is no more repressed. But at the same time, they start to synthesize cross-repressors LacI and CII. Therefore, a race occurs between the two genes (*lacI* and *cII*). In order to reach the expected behavior, production of CII should be "boosted" in comparison with LacI. This could be achieved, for instance, by inserting two coding sequences for CII protein in the operon #1. This boost leads to the inhibition of the operon #2 and the activation of the operon #3 by CI. As a consequence, the output can rise to high. On the other hand, if AHL is low when PreQ1 drops from high to low, the scenario is more obvious because the operon #1, which does not have a constitutive promoter, remains inactive. A steady state in which the operon #2 is active and represses operons #1 and #3 is quickly reached. As a consequence, the output activity drops to low. When PreQ1 is low, activity of the operon #2 depends only on activity of operon #1. If the operon #1 is inhibited, CII is not synthesized, thus the operon #2 is active and the output is low. If the operon #1 is active, the operon #2 is repressed and the output is high. Nevertheless, AHL may change at any time and especially may fall to low while PreQ1 is on. If the output is already inactive, it does not matter. However, to prevent a change of output state while it is active, the activity of the operon #1 has to be maintained by the self-activation of LasI. Finally, when PreQ1 rises from low to high, activity of operons#1 and #2 decreases without any modification of the output gene activity.

4.2. Modeling and simulation results

The BioDFF is modeled with Matlab. We reuse the classical equations introduced in Chapter 2. For the three operons, all the 20 parameters are set to the default value (see Table 1) except K_{R2}, which is equal to the half of the default value (0.005) and models the fact that the repression of CII is boosted in comparison to one of LacI. Parameter values are chosen accordingly to (Alon 2006).

4.2.1. Global behavior

The behavior of the system is tested for a regular signal Clock (frequency of 10^{-5} Hz) and a signal Data which is alternatively on and off according to a pattern that covers all the possible use cases.

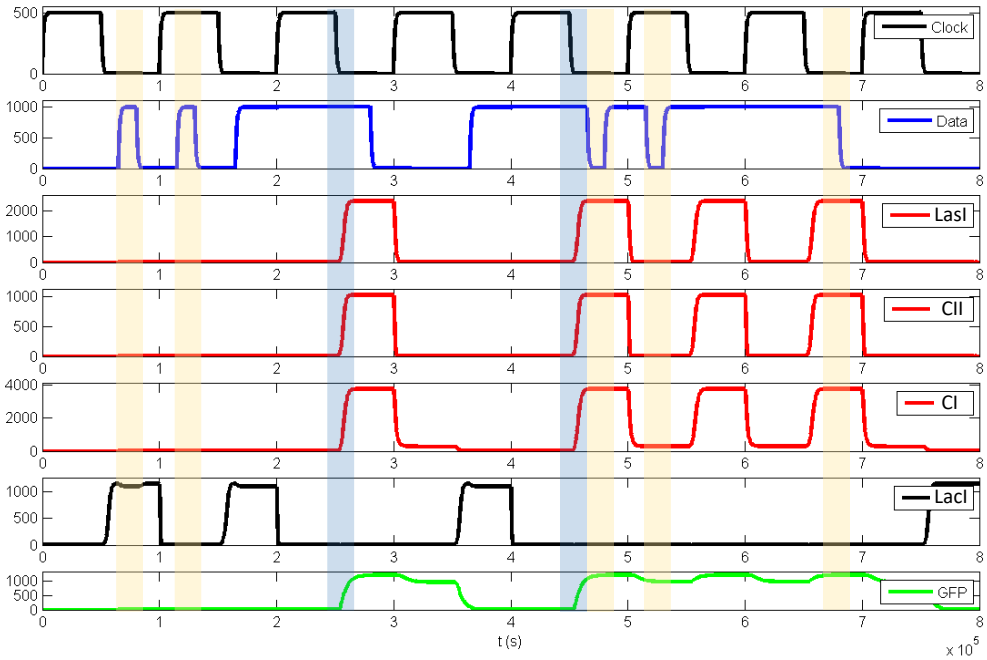


Fig. 15 Global behavior of the system. The y axis is in nM. The system is turned on (i.e. the output protein GFP is produced) only when the data is ON (high level) at a falling edge of the clock (blue rectangles). The output of the system does not vary if the data varies between falling edges of the clock (yellow rectangles).

First of all, the results of the simulation show that at each falling edge of the clock, the output (GFP) follows the input data: when data is low, the output is or turns low and when data is high, the output is or turns high. The modeled system correctly imitates the input at the falling edge of the clock, as expected. As for the memorization process, we can see that it is also correct. The output has to memorize its previous state whenever the clock is not falling from high to low. To see that we varied the input data between two falling edges of the clock (yellow rectangles on Fig. 15) and we see that the output does not vary.

We can also notice that the level of GFP, though being and staying high whenever it should, varies slightly (less than 20%). This phenomenon is due to the clock signal going high, meaning that the operon #1 no longer produces CI, activator of operon #3 and therefore enhancer of GFP expression. GFP high level only depends on its self-activation. Depending on the threshold needed to consider the output to be high, this could pose a problem. In our case, as the level of GFP stays in the order of magnitude of protein concentration considered high, we accept this difference. Increasing $Ktl3$ (or $Ktr3$) reduces this problem (data not shown). This can be achieved by using a stronger promoter.

4.2.2. Robustness towards leakiness of operon #3

As mentioned in the model description, the presence of a positive feed-back loop could render the system unstable. We therefore added leakiness to promoter #3: we added a term $\alpha \cdot Ktr3$ and the maximal value of the Hill function was modified to $(1-\alpha) \cdot Ktr3$.

The results show that for low values of leakiness ($\alpha < 5\%$) the system behaves normally (see Fig. 16). However, when reaching high values of leakiness, operon #3 is able to self-activate when the clock is on. If the data was previously off, operon #2 would previously have been on meaning that there would still be the repressor protein CII present in the environment. What the simulation shows is that this residual repressor is not enough to counteract the leakage of operon #3 when the said leakiness

becomes too high. A potential solution would be to increase the strength of the said repressor, or to ensure non-leaky promoter. The latter suggestion seems the more realizable.

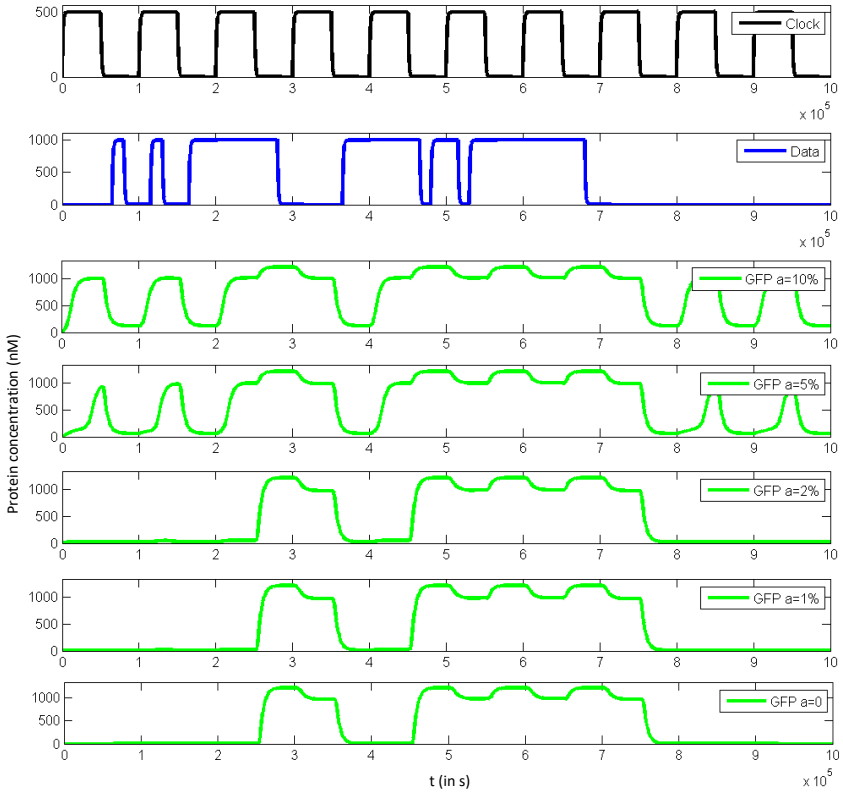


Fig. 16 Simulation with a leaky promoter #3. The relative value of leakiness (alpha) is indicated in the upper right corner of each diagram of the output, GFP.

4.2.3. Robustness towards noise during transcription

Another way to test the robustness of the circuit is to add noise during the transcription process. We add Poisson noise on the Ktr of the 3 operons, which in biology corresponds to natural noise concerning the transcription process. Results are shown in Fig. 17. Our system tolerates up to 10% of noise on the transcription process.

We then integrate our circuit in systems implementing sequential functions, namely a frequency divider by 2 also known as a 1-bit counter, and a 2-bits counter.

4.3. Design of biological counters

To design a 1-bit counter, we must first design a system that divides the frequency by 2. In microelectronics, this is realized by connecting the output of the flip-flop to its own data input through an inverter gate (Fig. 18A). To this aim, we need operon #3 (producing the output) to express a repressor, which in turns represses an activator for operon #1. Thereafter, we will name this repressor R4. In the model, we only modified the equations so that operon #3 synthesizes R4 and the data input is now negatively regulated by R4.

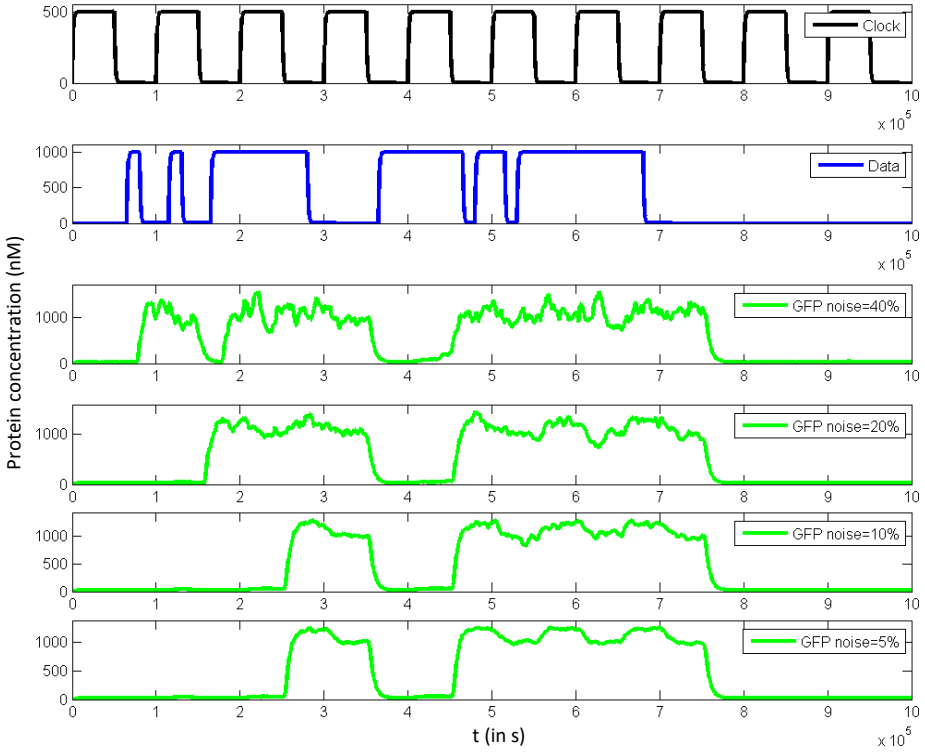


Fig. 17 Simulation of a noisy transcription. Relative value of noise attributed to the 3 Ktr is indicated in the upper right corner of each diagram of the output.

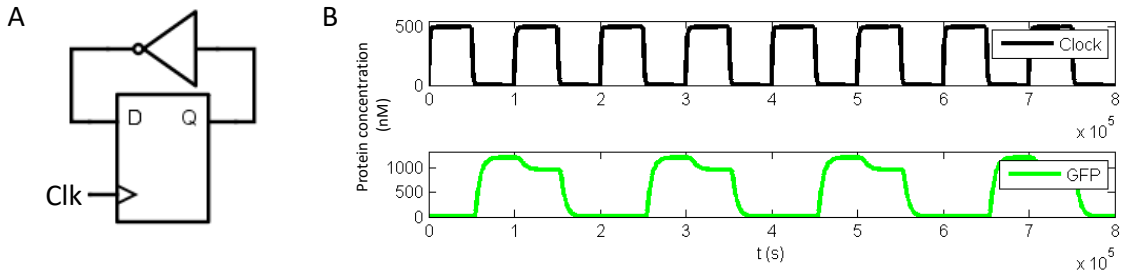


Fig. 18 Frequency divider. A shows the theoretical circuit (/Q is the complementary output of Q). B shows the result, GFP corresponds to the output Q.

We indeed obtain as output a signal with a frequency divided by 2 as shown by the simulation results on Fig. 18B.

To build a counter on 2 bits, we need to put 2 flip-flop in series, the clock of the second one being the output of the first (see Fig. 19). We simply copied the model of the first system. To avoid impedance problem, the output of the first flip-flop was buffered so that the clock of the second flip-flop was in the expected range. A buffer here is simply an activated gene. The activator is produced by operon #3 and act cooperatively with a Hill's number of 2. This first system did not work because of not sharp enough edges of the second clock. To solve this issue, two additional buffers were added, with the same cooperativity (Fig. 19). It can be noted that 2 buffers with a Hill's number of 3 or one buffer with a Hill's number of 5 also gives correct behavior. Because of biological delays, we observe 'glitch' state when the system transitions from the state corresponding to 01 to the one corresponding to 10 (the

system briefly goes to the state 00) as well as between 11 to 00 (the system briefly goes to the state 10). Depending on the application, this might be an issue or not. If required, a solution to this problem is found by using synchronous counters (Kuphaldt 2016).

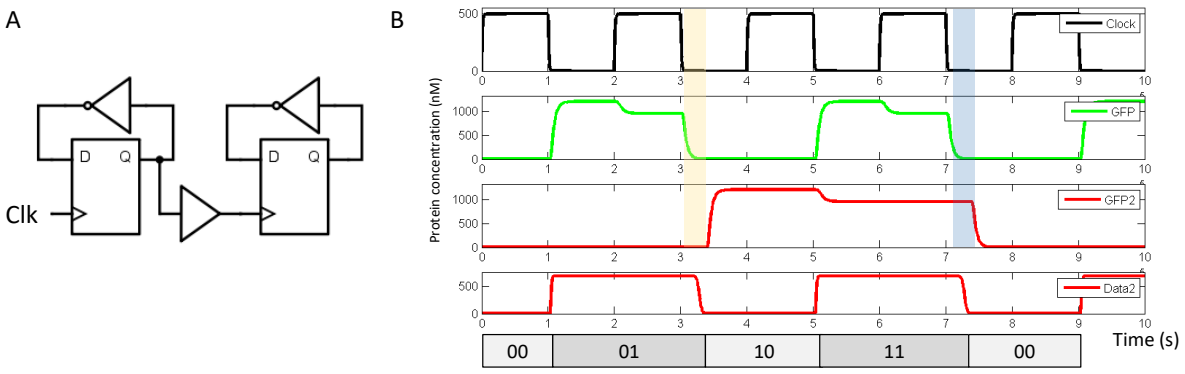


Fig. 19 A biological counter. It is composed of two flip-flop, the clock of the second one being the output of the first (A). The first (resp. second) digit is the output of the first (resp. second) flip-flop (B, last row). GFP (resp. GFP2) is the output of the first (resp. second) flip-flop (B). Data2 is the buffered output of the first flip-flop. Glitches can be observed during the transition between 01 to 10 (yellow rectangle), and between 11 to 00 (blue rectangle).

To conclude, a new genetic D flip-flop has been designed. It is an alternative of a structure previously introduced by Hoteit et al. in 2012. Its main advantage consists in being more compact (only three operons against seven for Hoteit). The other side of the coin is that our system relies on a competitive reaction between two repressors and a cognate promoter, which renders the system less robust than Hoteit's. Indeed, the simulations carried out in Matlab showed that system parameters should be controlled precisely enough to ensure proper functioning of the system, which can be easily done *in silico* but could be a tricky challenge when realized with actual genetic material.

4.4. Necessity to split the system

With this DFF, we can now use GeNeDA with GPLs that include DFF and model sequential systems. However, for large systems requiring more than one DFF, a drawback may arise. Indeed, this single component, the DFF, requires many genes. In biology, crosstalk between genes has to be avoided as there is no wire to connect a component to another one: the proteins are all in the same compartment, the cell, and can interact with every other protein. To avoid this crosstalk, we would need different implementations of the DFFs, which means a high number of genes, even though our construct is more compact. In particular, this fact has not been taken into account in the design of the 2-bit counter where it has been considered that it is possible to integrate two strictly identical D flip-flops within the same cell.

As introducing a high number of genes in a cell might prove to be challenging, a second solution is more often privileged nowadays: the system is split into several subsystems implemented into different populations of cells. Looking back on the example of the counter, this would consist in creating two populations of cells, one for the first bit and the other for the second. Each cell includes a single D-flip-flop whose clock signal is generated by the previous cell. Virtual prototyping of this kind of system requires a space simulator and the ability to synchronize signals (e.g. the clock signal) across multiple cells. This paradigm is discussed in details in Chapter 7.

5. Conclusion

In this chapter, we demonstrated a way to automate the design of combinatorial GRNs by reusing electronics tools. From a high level specification (truth table, Verilog file), the tool suite designs a GRN and provides associated SBML and SystemC-AMS models. A special care was given to the build of the generic part library, which is the cornerstone of the software. It has been made as realistic as possible regarding the possibilities offered by synthetic biology and has been validated on a benchmark of standard circuits.

We also address the question of the design of sequential GRN. For asynchronous sequential systems, Huffmann's method provides two sets of Boolean equations which can be given as input of GeNeDA to obtain the equivalent GRN. Nevertheless, simulation results highlight major shortcomings of asynchronous GRN. Indeed, the delay introduced by gene activation/inhibition in the feedback loops induces malfunction and/or instability.

For synchronous circuits, DFFs are required. Drawn from an analogy with electronics, Hoteit et al. demonstrated a GRN having the behavior of a DFF but involving many genes and promoters. However, its feasibility with actual biological material has not been demonstrated. Thus, we proposed a more compact biological DFF. The robustness of this circuit, which also involves feedback loops, has been validated. As the GeNeDA's digital synthesizer can directly handle Verilog files that describe a synchronous system, the introduction of our DFF in our Generic Part Library enables the design of any Boolean GRN, at least from a theoretical point of view.

In practice, the realization of such GRN might be a little bit trickier. The first bottleneck that we will encounter is the number of part required to build a synchronous system. An example is given in (Madec et al. 2013). A simple 3-states synchronous system was generated with GeNeDA and required 11 promoters and 2 DFF, which is a large GRN in comparison with the technological know-how. Concerning the DFF itself, another problem may arise. In synthetic biology, all the parts are put together in the same compartment, the cell. Crosstalk between genes is thus unavoidable (there is no wire to connect a component to another one, as in electronics) and may lead to the same issues that we highlighted for asynchronous systems.

For a system with multiple DFFs, a solution consists in using, for each instance of the DFF, a different set of promoters and regulators that are uncoupled from the other. As this solution might be challenging, a privileged alternative consists in splitting the system into several subsystems that are implemented into different populations of cells. The study of this kind of system is more complicated and requires a simulator that takes spatial location into account. In particular, the issue of the synchronous distribution of the clock signal to each cell may arise.

These points are discussed in much more details in the third part of this manuscript. Before that, the question of the design automation of GRN that cannot be represented at the Boolean level of abstraction (typically, a system exhibiting a bell-shaped response) is tackled in the next chapter.

6. References

Alon, Uri. 2006. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. http://books.google.fr/books/about/An_Introduction_to_Systems_Biology.html?id=pAUdPQlCZ54C&pgis=1.

Ashenden, Peter J. 2010. *The Designer's Guide to VHDL*. Morgan Kaufmann.

Becskei, A, B Séraphin, and L Serrano. 2001. "Positive Feedback in Eukaryotic Gene Networks: Cell Differentiation by Graded to Binary Response Conversion." *The EMBO Journal* 20 (10): 2528–35. doi:10.1093/emboj/20.10.2528.

Bhatia, Swapnil, and Douglas Densmore. 2013. "Pigeon: A Design Visualizer for Synthetic Biology." *ACS Synthetic Biology* 2 (6): 348–50.

Brayton, Robert K, Gary D Hachtel, Alberto Sangiovanni-Vincentelli, Fabio Somenzi, Adnan Aziz, Szu-Tsung Cheng, Stephen Edwards, et al. 1996. "VIS: A System for Verification and Synthesis." In *Computer Aided Verification*, 428–32.

Chang, Dong-Eun, Shelly Leung, Mariette R Atkinson, Aaron Reifler, Daniel Forger, and Alexander J Ninfa. 2010. "Building Biological Memory by Linking Positive Feedback Loops." *Proceedings of the National Academy of Sciences of the United States of America* 107 (1): 175–80. doi:10.1073/pnas.0908314107.

Court, Donald L, Amos B Oppenheim, and Sankar L Adhya. 2007. "A New Look at Bacteriophage Lambda Genetic Networks." *Journal of Bacteriology* 189 (2): 298–304. doi:10.1128/JB.01215-06.

Elowitz, Michael B, and S Leibler. 2000. "A Synthetic Oscillatory Network of Transcriptional Regulators." *Nature* 403 (6767): 335–38. doi:10.1038/35002125.

Gardner, T S, C R Cantor, and James J Collins. 2000. "Construction of a Genetic Toggle Switch in Escherichia Coli." *Nature* 403 (6767): 339–42. doi:10.1038/35002131.

Giovanni De Micheli. 1994. "Synthesis and Optimization of Digital Circuits." In *Synthesis and Optimization of Digital Circuits*, 576. McGraw-Hill. https://dl.acm.org/citation.cfm?id=541643.

Gong, Zhou, Yunjie Zhao, Changjun Chen, and Yi Xiao. 2012. "Computational Study of Unfolding and Regulation Mechanism of preQ1 Riboswitches." Edited by Vladimir N. Uversky. *PloS One* 7 (9). Public Library of Science: e45239. doi:10.1371/journal.pone.0045239.

Gray, K M, L Passador, B H Iglewski, and E P Greenberg. 1994. "Interchangeability and Specificity of Components from the Quorum-Sensing Regulatory Systems of Vibrio Fischeri and Pseudomonas Aeruginosa." *Journal of Bacteriology* 176 (10): 3076–80. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=205467&tool=pmcentrez&rendertype=abstract.

Hoops, Stefan, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. 2006. "COPASI—a Complex Pathway Simulator." *Bioinformatics* 22 (24). Oxford Univ Press: 3067–74.

Hoteit, Imad, Nawwaf Kharma, and Luc Varin. 2012a. "Computational Simulation of a Gene Regulatory Network Implementing an Extendable Synchronous Single-Input Delay Flip-Flop." *BioSystems*. Elsevier Ireland Ltd, 1–15. doi:10.1016/j.biosystems.2012.01.004.

———. 2012b. "Computational Simulation of a Gene Regulatory Network Implementing an Extendable Synchronous Single-Input Delay Flip-Flop." *Bio Systems* 109 (1). Elsevier Ireland Ltd: 57–71. doi:10.1016/j.biosystems.2012.01.004.

Hucka, M., A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, et al. 2003. "The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models." *Bioinformatics* 19 (4): 524–31. doi:10.1093/bioinformatics/btg015.

Ismail, Mohammed, and José E Franca. 2012. *Introduction to Analog VLSI Design Automation*. Vol. 95. Springer Science & Business Media.

Jamieson, Peter, Kenneth B Kent, Farnaz Gharibian, and Lesley Shannon. 2010. "Odin II-an Open-Source Verilog HDL Synthesis Tool for CAD Research." In *Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on*, 149–56.

Jerke, Goeran, and Jens Lienig. 2009. "Constraint-Driven Design: The next Step towards Analog Design

Automation." In *Proceedings of the 2009 International Symposium on Physical Design*, 75–82.

Koza, John R, Forrest H Bennett, David Andre, Martin A Keane, and Frank Dunlap. 1997. "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming." *Evolutionary Computation, IEEE Transactions on* 1 (2). IEEE: 109–28.

Kuphaldt, Tony R. 2016. "Asynchronous Counters : Sequential Circuits - Electronics Textbook." http://www.allaboutcircuits.com/textbook/digital/chpt-11/asynchronous-counters/.

MacVittie, Kevin, Jan Halámek, and Evgeny Katz. 2012. "Enzyme-Based D-Flip-Flop Memory System." *Chemical Communications (Cambridge, England)* 48 (96): 11742–44. doi:10.1039/c2cc37075a.

Madec, Morgan, Francois Pecheux, Yves Gendrault, Lujo Bauer, Jacques Haiech, and Christophe Lallement. 2013. "EDA Inspired Open-Source Framework for Synthetic Biology." In *Biomedical Circuits and Systems Conference (BioCAS), 2013 IEEE*, 374–77.

Micheli, Giovanni De. 1994. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education.

Mishchenko, Alan. 2015. "ABC Website." *Berkeley Website*.

Moon, Tae Seok, Chunbo Lou, Alvin Tamsir, Brynne C Stanton, and Christopher A Voigt. 2012. "Genetic Programs Constructed from Layered Logic Gates in Single Cells." *Nature* 491 (7423): 249–53. doi:10.1038/nature11516.

Paul Horowitz, and Winfield Hill. 1989. *The Art of Electronics*. Cambridge University Press.

Pêcheux, Francois, Morgan Madec, and Christophe Lallement. 2010. "Is SystemC-AMS an Appropriate " Promoter " for the Modeling and Simulation of Bio-Compatible Systems ?" In *IEEE International Symposium on Circuit and Systems (ISCAC)*.

Rabaey, Jan M, Anantha P Chandrakasan, and Borivoje Nikolic. 2002. *Digital Integrated Circuits*. Vol. 2. Prentice hall Englewood Cliffs.

Rutenbar, Rob. 1993. "Analog Design Automation: Where Are We? Where Are We Going?" In *Custom Integrated Circuits Conference, 1993., Proceedings of the IEEE 1993*, 11–13.

Thomas, Donald E., and Philip R. Moorby. 2002. *The Verilog® Hardware Description Language*.

Vachoux, Alain, Christoph Grimm, and Karsten Einwich. 2003. "Analog and Mixed Signal Modelling with SystemC-AMS." In *Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on*, 3:III--914.

Winkler, Wade C, and Ronald R Breaker. 2005. "Regulation of Bacterial Gene Expression by Riboswitches." *Annual Review of Microbiology* 59 (January): 487–517. doi:10.1146/annurev.micro.59.030804.121336.

Chapter 5
Design Automation at Analog Level

1. Introduction

In this chapter, we will focus on the automatic design of biological systems that are described by an analog behavior. This approach is based on two motivations. First, several systems cannot be described by a Boolean function. An example of such a system is Basu's band-pass (Basu et al. 2005) for which the output reporter protein is synthesized only for an intermediate concentration of input protein (more details about this system are given in the Section 2.3). Second, by opposition to digital

electronics, the gap between the Boolean abstraction of a gene regulatory network and its actual behavior can be important. In particular, connections between genes are not so obvious. For instance, let gene #1 produce an activator for gene #2. Even if gene #1 is active, the amount of synthesized protein may be not sufficient to activate gene #2. Thus, it is often necessary to supplement the high-level design by a design at a lower level of abstraction. Another example is the amplification function described in (Xie et al. 2011) where two layers of regulation are needed to generate a sufficiently sharp transition between states.

In the Chapter 2, we defined two levels of abstraction below the Boolean one, namely multi-valued logic and analog. Designing at intermediate levels of abstraction has already been demonstrated. René Thomas' investigations deserve to be highlighted (Thomas, Thieffry, and Kaufman 1995). Indeed, he developed a formalism for the modeling of dynamical behaviour of biological regulatory network through multivalued logic variables, rules, graphs and graphical representation of the state space. A few years later, Gilles Bernot extended this approach to include temporal properties of GRN (Bernot et al. 2004). An alternative has been recently investigated based on fuzzy logic which is used to describe the rules that govern the relations between protein concentration and gene states (Gendrault et al. 2014). The main asset of fuzzy logic in comparison with standard multivalued logic is that the link between fuzzy value and actual concentration of protein is never lost. Fuzzy logic can be used to describe systems at an intermediate level of abstraction, but also for design purposes, as it has been shown in (Gendrault et al. 2014). In this case, an algorithm tests all the possible combinations of rule matrices (picked up from a library) and finds the one that best meets the expected behavior. Each rule matrix corresponds to a gene-protein interaction and its content provides the designer with important clues about the choice of the gene-protein interaction to implement.

Design automation of GRN at the analog level of abstraction is very close to the analog synthesis in miocroelectronics. This topic has been widely investigated since the beginning of the 80's and is still an open question. Several tools and methods have been demonstrated using specific formalisms and formal computation (Doboli and Vemuri 2003; Lohn and Colombano 1998). Nevertheless, these developments have not led to a generic tool which would have been widespread in the analog designers community. The main reason is that they were too complex and too specific to be used for a large range of circuits. In addition, they require libraries and/or artificial learning methods or the translation of designer experience to formal rules, which is not straightforward. In a more general way, it was observed that the ratio between the implementation complexity of such algorithms and the complexity of circuits that can be synthesized were poor in comparison to a hand-made design.

One of the most outstanding breakthrough in the domain of analog synthesis has been made by Koza in 1997 (Koza et al. 1997) based on genetic programming algorithms. He demonstrates the potential of his method on a large set of electronic circuits (filters, amplifiers, controllers) for which genetic algorithms provided solutions (circuit topology and component dimensioning) very competitive in comparison with human intelligence (Koza 2003). At the time, the main shortcoming of evolutionary algorithms was that they required computing power that could only be provided by supercomputers. This is no longer true with current technologies: the exploitation of the parallelized computation over small networks and/or the exploitation of the performance of the Graphical Processor Units (GPU) (Maitre et al. 2009) makes it possible to obtain such results with reasonably priced computing systems.

Generally speaking, analog synthetis is a subset of inverse problems. Inverse problems consist in searching the functions (in this case the model of the system) which give a response as close as possible

to the expected one (in this case, the specification). Several methods exist to tackle this kind of problems. Most of them require the topology of the system to be known *a priori* and optimize the system's parameters to converge toward a target. They are briefly overviewed in the first Section of this chapter. Emphasis is put on evolutionary algorithms which seems to be promising in our context. The application of different families of evolutionary algorithms in the context of synthetic biology is demonstrated. First, we consider GRN for which the topology (i.e. the assembly of elementary mechanisms) has been previously fixed and for which we aim at finding the most appropriate building block to realize each mechanism. For this purpose, two approaches are possible. On the one hand, combinatorial optimization can be used to try different combinations of building blocks and find those exhibiting the best response. On the other hand, continuous optimization can be used to find the sets of parameters for each model of each mechanism that give the best response. These sets of parameters then guides the designer in the choice of the building blocks that will be used to perform each mechanism. This second method is illustrated in the context of a GRN in Section 2. Finally, we tackle the much more complex problem of the design automation of a GRN for which the topology of the network is not defined *a priori*. To do that, genetic programming methods can be applied, as discussed in Section 3.

2. Parameter optimization for a given GRN

The first task at hand is to optimize the components of a given system. For a GRN of a given topology, many implementations can exist. Indeed, many activators and repressors of different affinity towards their promoter can be used. However, not all kind of regulators are suited to realize a given function.

We will first see which method is best suited to solve this issue, then we present our approach with evolutionary algorithms and finally show our results obtained with this algorithm on two different problems.

2.1. A brief overview of inverse problem

Let f be a function (in our case, the GRN model) that depends on a set of parameters x (model's biochemical parameters). Applying f on x leads to y, the response of the system. Inverse problem consists in defining a response $\tilde{y}$ (the target response) and exploring the search space to find a value of x such as $f(x) = \tilde{y}$. In other words, it is equivalent to look for the value of x that minimizes the function $h(x) = f(x) - \tilde{y}$ (the subtraction here corresponds to an operator able to evaluate the difference between two responses). Most of the time, $h(x)$ (also called the score function) is multimodal function, i.e. it may have many local optima. The research of the global minimum among all local ones is not straightforward.

2.1.1. Local algorithms

Greedy algorithms like hill climbing and gradient descent allow to find a local minimum. For a discrete function, hill climbing starts with a random solution (x_0 on Fig. 1A). This algorithm visits its neighbors (other solutions in the search space) and only changes to the new solution if it is a better one (lower in case of minimization). Gradient descent can be applied to continuous derivable functions and also starts with a random solution x_0. This algorithm computes the gradient of the point and selects a new solution in the direction of the negative gradient (arrows on Fig. 1B). The distance of the newly visited point depends on the absolute value of the gradient: the greater the farther. This algorithm will

therefore achieve « big steps » when on a steep slope and gradually reduce the size of these steps when close to a gradient equal to 0, representing a local (if not global) minimum.

Both of these algorithms can be stuck in the first valley (local minimum) they find, therefore the position of the starting point x_0 is crucial. Moreover, if the discrete function is generated from a continuous one, the increment between each step is crucial not to miss the global minimum: we see on Fig. 1A that the global minimum (red cross) is only sampled by two solutions (the two blue lines flanking it) that are both above the lowest local minimum (red arrow). Thus, the computation time of greedy algorithms increases with the number of dimensions (more neighbors to evaluate).

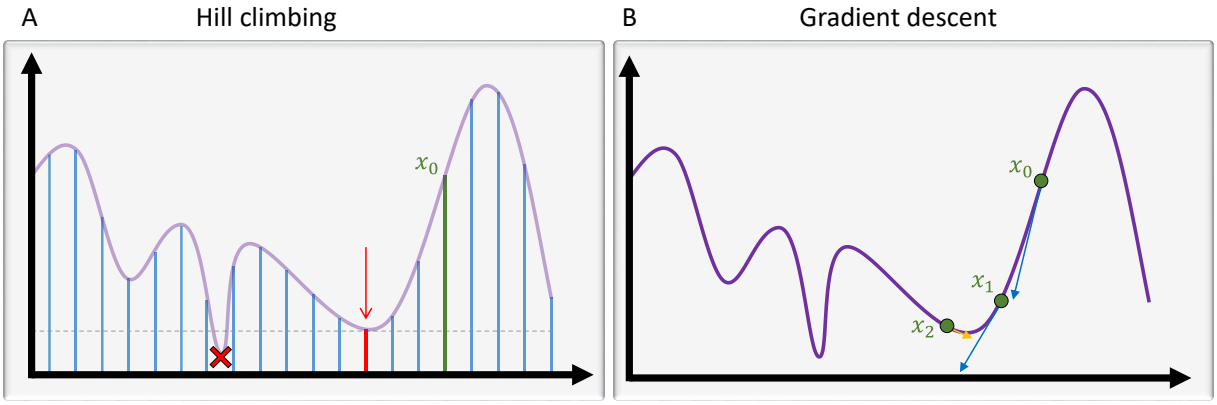


Fig. 1 Local optimization algorithms. A: Hill climbing used on a discrete function. x_0 is the starting point of the algorithm, the red arrow shows the local minimum found by the algorithm while the red cross shows the expected global minimum. B: gradient descent for a continuous function. x_i are the successive points visited by the algorithm, the blue and yellow arrows represent the associated negative gradients.

An alternative to local methods are global algorithms which search the global optimum.

2.1.2. Global algorithms

We can distinguish two categories of global algorithms: enumerative and stochastic. Enumerative algorithms are applied to problems that have a finite number of solutions (like combinatorial problems). They explore all solutions, hence guaranteeing to find the global optimum. However this exploration can be extremely long. Furthermore, not all problems at hand possess a finite number of solutions. We therefore focus here on stochastic algorithms.

On their lowest degree, stochastic algorithms such as Monte Carlo (Metropolis and Ulam 1949) evaluate a random set of solutions and select the best. Simulated annealing (Kirkpatrick, Gelatt, and Vecchi 1983) uses Monte Carlo. This algorithm introduces a variable called the temperature. The starting point is a random solution x_0 with a high temperature T. At each iteration, the algorithm evaluates a new solution. If this solution is better, it is selected as the next point. Otherwise, the algorithm selects the new points instead of keeping the initial one with a probability depending on the difference between the evaluation of both points and the temperature T (Metropolis et al. 1953). Then the temperature T decreases. With this probability, there is a chance not to always search in the current valley: the algorithm can also explore other points that might appear at first evaluation worse but lead to more promising valleys. Under certain conditions and an infinite time, this algorithm guarantees to find the global optimum. Obviously, an infinite convergence time is not practical.

In practice, global methods are often used with local methods. First, the global algorithm roughly scans every valley and selects the most promising one. Then the local algorithm is used to search the selected valley and find its minimum. These are called hybrid methods.

2.1.3. Configurable algorithms

Some algorithms allow to tune the convergence speed by adjusting two parameters: the exploration rate and the exploitation rate. Exploration relates to the scanning of the search space: the more exploration, the more the algorithm will evaluate unknown solutions. On the other hand, exploitation encourages the deepening of an explored region to find a better optimum and enhance convergence. A good trade-off between exploration and exploitation is important to explore all promising valleys and not miss an important one but also converge in a reasonable time. An example of such a tunable algorithm is the ant colony optimization (Dorigo and Stützle 2004). However, to adapt this kind of algorithm to a given problem, two parameters are often insufficient.

In a nutshell, we want an algorithm able to handle a wide variety of functions (discrete, continuous, derivable, multimodal…), that can find a global optimum in a reasonable time and that has enough parameters to adapt it to a given problem. Evolutionary algorithms are good candidates to these requirements and we chose to focus on them.

2.2. Evolutionary algorithms

2.2.1. Presentation

Evolutionary algorithms are composed of several adjustable operators. By specializing our algorithm to our problem, we have an algorithm that will converge faster than a simple stochastic search. However, we lose the guarantee to find the global optimum.

The concept of evolutionary algorithms appeared earlier than the other algorithms mentioned previously, with the work of Fraser in 1957 (reprinted in (Fogel and B. 1998)) and Bremermann in 1962. The foundations laid by this early work were left aside for a few years because the computing power could not keep up with the demands of such algorithms. A renaissance happened in the 1990s with the work and funding of super computers of Koza (Koza 1992).

To solve a problem with evolutionary algorithms, we need to be able to represent solutions to it, by a single value, a vector of integer or real numbers, a bit string, etc. Evolutionary algorithms can be classified into categories according to their representation among others (*e.g.* the operators): genetic algorithms (Kalyanmoy Deb and Goldberg 1989; Holland 1975) work with bit strings, evolutionary strategies (Rechenberg 1965) use real-valued vectors and genetic programming (Cramer 1985; Koza 1992) represents the solution as a program. In general, implemented algorithms use representations and operators from different paradigms so that this classification is less accurate.

As for other algorithms, we need a way to evaluate a solution. In evolutionary algorithm, the evaluator is called the fitness function. The evaluation can be absolute or relative.

Evolutionary algorithms take their inspiration from Darwinian evolution: the two main principles are stochastic modifications to an individual and selection among a population. An evolutionary algorithm therefore manipulates a population of solutions called individuals and applies operators to them.

2.2.2. Description of an iteration

Evolutionary algorithms proceed through different steps described on Fig. 2.

The first step of an evolutionary algorithm is the initialization of the population. Random solutions are generated. These solutions need to be evaluated with the help of the user-defined evaluation function. Each individual of the population therefore receives a fitness score. These individuals constitute the parents.

Here we can set an early stop condition. If the algorithm fulfills this condition, it stops and retrieves the best individual.

The next step is the creation of new individuals called children. For each child, the process is the following:

i) a given number (depending on the selection operator) of parents is selected in the population with a probability depending on their fitness score,
ii) a crossover operator is applied to create a child with a recombination of the parameters of the parents,
iii) a mutation operator is applied, leading to small alteration of the child's parameters.

Then, the children are evaluated and the population (parents + children) can be reduced using a selection operator. The surviving individuals constitute the next generation. The way the selection is performed and, more precisely, the way parents and children are treated is referred to as elitism. A strong elitism means that the next generation is only composed of children (every parent dies). Conversely, a weak elitism means that the next generation is only composed of best individuals, should they be parents or children.

A new iteration with the same steps is then carried out until the stop criterion (if one has been defined) is met or until the algorithm reaches a given number of generations.

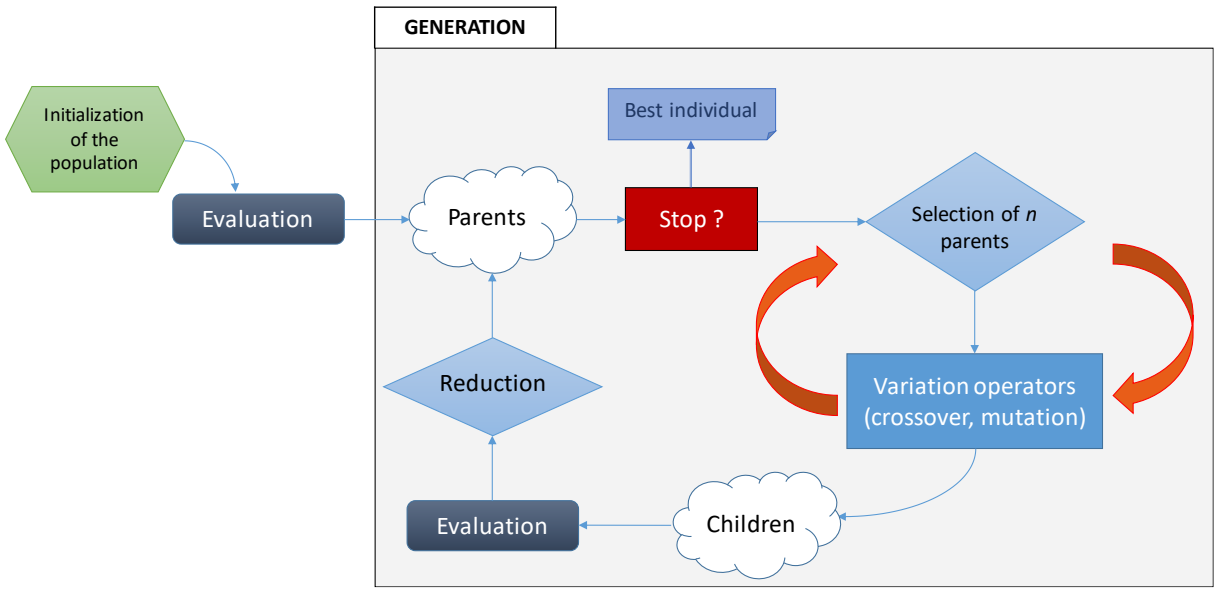


Fig. 2 Diagram of an evolutionary algorithm (adapted from MOOC Pierre Collet).

In comparison with other methods, evolutionary algorithms have the advantage to be highly configurable, especially through their different operators (crossover, mutation, selection). There are many ways to select, recombine and mutate individuals. The best way to have an efficient algorithm is to customize these operators to adapt them to the problem. During this tuning phase, one has to keep in mind that, for evolutionary algorithms, convergence speed is not a guarantee of quality. An

algorithm converging too fast will probably converge to a local optimum. Conversly, a long converging time is neither practical nor competitive. We therefore tinker our operators so that the selection pressure (increasing the convergence speed) and the stochasticity (decreasing the convergence speed) are in a good balance.

2.2.3. Implementation: EASEA platform

In evolutionary algorithms, a major limiting factor in the execution time of an iteration is the evaluation. However, it is possible to evaluate each individual in parallel, as they have no influence on the evaluation of others. We therefore decided to implement our algorithm on the EASEA (Easy Specification of Evolutionary Algorithms) platform, developed in 1999 by Collet et al. (Collet et al. 2000). This platform allows an easy implementation of evolutionary algorithms and an automatic parallelization over GPGPUs.

2.3. Results on the band-pass

Continuous optimization with genetic algorithm can be used to select the most appropriate actual promoter and regulator to use in an abstracted GRN once its topology has been fixed (for instance, by a first design automation stage at a higher level of abstraction). An ODE-based model of the network is first written. Features of promoters and transcription factors (strength of a repression, leakiness of a promoter) correspond to the parameters employed in the model (Hill function notably). Then, an evolutionary algorithm is used. The individuals that evolve are represented by a vector of the parameters of the model. This vector is referred to as the genome of the individual. In this context, a gene is therefore an element of this vector. However, to avoid confusion, we will employ the term *algogene* to refer to an element of this vector. The best individual resulting from the computation guides the designer in the choice of the promoters and regulators to implement.

This principle is applied to the design of an existing GRN (a band-pass detector) which performs a function that cannot be modeled by Boolean equations. The system and its model is first described. The algorithm is then described with an emphasis put on the choice we made for the different operators. Finally results are discussed. At the end of this section, results obtained on another system (GRN also involving protein-protein interaction) are also described.

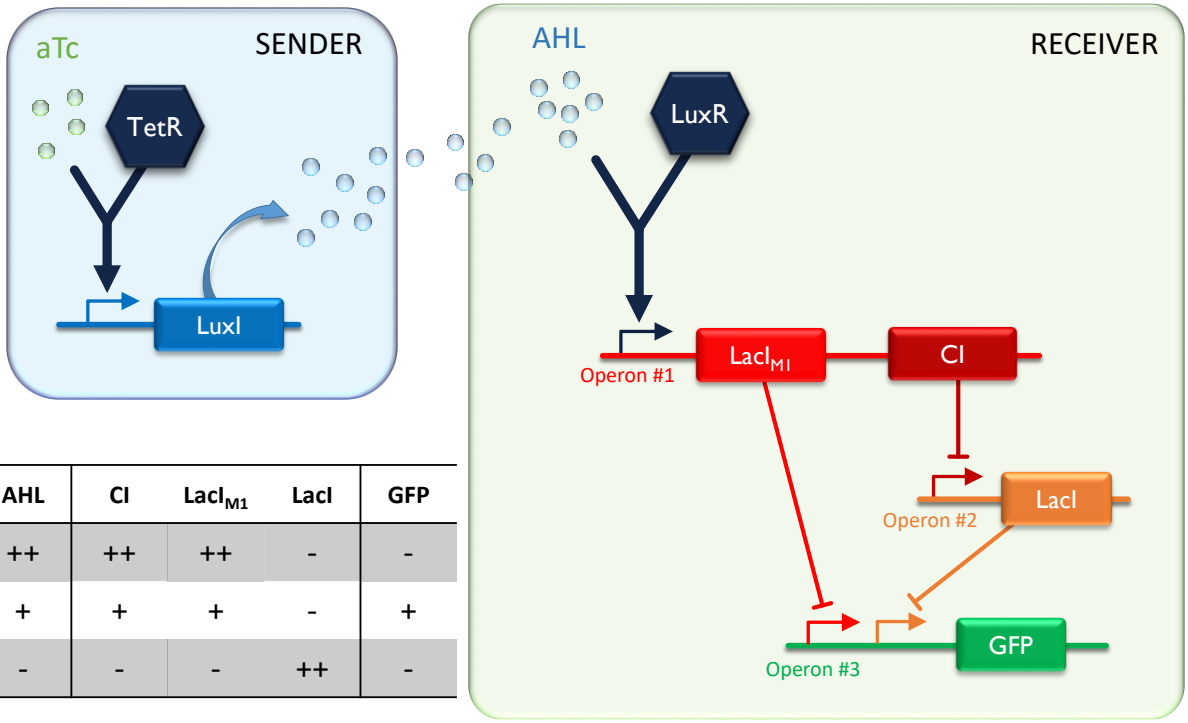
2.3.1. Description of the system

The biological band-pass system developed by Basu *et al.* (Basu et al. 2005) is depicted in Fig. 3. This system allows the detection of an intermediate concentration of acyl-homoserine lactone (AHL). The complete system consists in two populations of cells: senders and receivers. Senders synthesize and emit isotropic AHL (the signal) inside a Petri dish. AHL is a small molecule able to diffuse in the gelose of the Petri dish and to enter cells. Receivers react to the concentration of AHL and produce GFP, a Green Fluorescent Protein (the output in this system) if the concentration of AHL ([AHL]) is comprised within a specific interval (around 5.10^{-2} µM). In practice, one or many groups of senders are laid on specific spots on a Petri dish covered with receiver cells.

Behavior

Our focus was put on the biological core that is computing the output, namely the receiver cells. They are composed of three operons (Fig. 3). Operon #1 is positively regulated by AHL (via LuxR, a constitutively expressed protein) and expresses a modified LacI ($LacI_{M1}$). Operon #1 also produces CI, which in turn inhibits the expression of operon #2 which produces LacI. Both $LacI_{M1}$ and LacI inhibits the expression of GFP via the operon #3. As mentionned, $LacI_{M1}$ is a modified version of the LacI

repressor: it is a weaker repressor, meaning that its dissociation constant with its promoter is higher than LacI. A higher concentration of $LacI_{M1}$ (with regards to LacI) is required to shut down GFP expression.



AHL	CI	$LacI_{M1}$	LacI	GFP
++	++	++	-	-
+	+	+	-	+
-	-	-	++	-

Fig. 3 The band-pass system described by Basu *et al.* (Basu et al. 2005) and the table describing its behavior for various concentrations of AHL. ++, + and – correspond respectively to a high, medium and low concentration.

Let us consider that the sender cells are constantly sending AHL. Like every molecule, AHL degrades over time. Hence its concentration is higher near the sender cells than far away from them. The receiver cells near a group of sender cells therefore receive more AHL than those further away. Depending on this AHL concentration, GFP is or is not expressed.

Indeed, the level of $LacI_{M1}$ and CI depends on the concentration of AHL: the more AHL, the higher the concentration of $LacI_{M1}$ and CI (see Chapter 2 for the Hill equation governing this correlation).

With the fact that $LacI_{M1}$ is a weaker promoter, we can know understand the different scenarios summarized in the table on Fig. 3. When AHL concentration is high (++), $LacI_{M1}$ concentration is also high and GFP is not expressed. When AHL concentration is low (-), $LacI_{M1}$ is low but also CI hence LacI is high. A high concentration of LacI results in no expression of GFP. When AHL concentration is medium, $LacI_{M1}$ concentration is medium and as it is a weak repressor, it is not sufficient to repress GFP. However, CI is a strong repressor: even at a medium concentration, it can repress LacI expression. With no LacI and not enough $LacI_{M1}$, GFP is expressed.

Model

Each gene transcription into mRNA and subsequent translation into protein can be modeled by ordinary differential equations similar to the one described in Chapter 2. As the system is evaluated at the steady state, ODEs lead to direct analytic relationship giving the GFP concentration as a function of AHL concentration (Equation 1 to Equation 4) and 22 parameters summarized in Table 1. For the evolutionary algorithm, these parameters constitute the genome of the individuals to be evolved.

Equation 1

$$[LacI_{M1}] = \frac{k_{TL1}}{d_{LacIM1}} \cdot \frac{k_{TR1}}{d_{mRNA1}} \cdot \frac{1}{1 + \left(\frac{K_{A1}}{[AHL]}\right)^{n_{A1}}}$$

Equation 2

$$[CI] = \frac{k'_{TL1}}{d_{CI}} \cdot \frac{k_{TR1}}{d_{mRNA1}} \cdot \frac{1}{1 + \left(\frac{K_{A1}}{[AHL]}\right)^{n_{A1}}}$$

Equation 3

$$[LacI] = \frac{k_{TL2}}{d_{LacI}} \cdot \frac{k_{TR2}}{d_{mRNA2}} \cdot \frac{1}{1 + \left(\frac{[CI]}{K_{R2}}\right)^{n_{R2}}}$$

Equation 4

$$[GFP] = \frac{k_{TL3}}{d_{GFP}} \cdot \frac{k_{TR3}}{d_{mRNA3}} \cdot \frac{1}{1 + \left(\frac{[LacI_{M1}]}{K_{R3}}\right)^{n_{R3}} + \left(\frac{[LacI]}{K'_{R3}}\right)^{n'_{R3}}}$$

Table 1 Parameters table. The boundaries correspond to the limits in which the parameters were constrained during initialization (as well as crossover and mutation if relevant). Dissociation constants are given in µM and transcription constants in µM.s^{-1}.

Parameter	**Description**	**Boundaries**	
		min	max
$\boldsymbol{k_{TR1}}$	Transcription constant of operon #1	10^{-3}	10^{3}
$\boldsymbol{K_{A1}}$	Dissociation constant of AHL	10^{-4}	10^{3}
$\boldsymbol{n_{A1}}$	Hill's number of AHL	1	4
$\boldsymbol{d_{mRNA1}}$	Degradation constant of mRNA of operon #1	10^{-3}	10^{-1}
$\boldsymbol{k_{TL1}}$	Translation constant of $LacI_{M1}$ (op. #1)	10^{-7}	10^{-5}
$\boldsymbol{d_{LacIM1}}$	Degradation constant of $LacI_{M1}$	10^{-4}	10^{-2}
$\boldsymbol{k'_{TL1}}$	Translation constant of CI (op. #1)	10^{-7}	10^{-5}
$\boldsymbol{d_{CI}}$	Degradation constant of CI	10^{-4}	10^{-2}
$\boldsymbol{k_{TR2}}$	Transcription constant of operon #2	10^{-3}	10^{3}
$\boldsymbol{K_{R2}}$	Dissociation constant of CI	10^{-4}	10^{3}
$\boldsymbol{n_{R2}}$	Hill's number of CI	1	4
$\boldsymbol{d_{mRNA2}}$	Degradation constant of mRNA of operon #2	10^{-3}	10^{-1}
$\boldsymbol{k_{TL2}}$	Translation constant of operon #2	10^{-7}	10^{-5}
$\boldsymbol{d_{LacI}}$	Degradation constant of LacI	10^{-4}	10^{-2}
$\boldsymbol{k_{TR3}}$	Transcription constant of operon #3	10^{-3}	10^{3}
$\boldsymbol{K_{R3}}$	Dissociation constant of $LacI_{M1}$	10^{-4}	10^{3}
$\boldsymbol{n_{R3}}$	Hill's number of $LacI_{M1}$	1	4
$\boldsymbol{K'_{R3}}$	Dissociation constant of LacI	10^{-4}	10^{3}
$\boldsymbol{n'_{R3}}$	Hill's number of LacI	1	4
$\boldsymbol{d_{mRNA3}}$	Degradation constant of mRNA of operon #3	10^{-3}	10^{-1}
$\boldsymbol{k_{TL3}}$	Translation constant of operon #3	10^{-7}	10^{-5}
$\boldsymbol{d_{GFP}}$	Degradation constant of GFP	10^{-4}	10^{-2}

2.3.2. Setup of the evolutionary algorithm

Population size and selection operator

The population is composed of 1000 individuals. At each generation, 1000 offspring were generated and individuals for the next generation were selected *via* a binary tournament with weak elitism. A binary tournament is a stochastic operator defined with a probability p comprised between 0.5 and 1. The operator randomly selects 2 individuals in the population and returns the best with a probability p.

A deterministic tournament of size n randomly selects $n-1$ individuals in the population. The tournament gives a rank to an individual of this group based on the number of outperformed solutions in this group. Namely, the rank is $n - number\ of\ beaten\ solutions$. The algorithm then selects the solutions with the best rank. As the group is formed randomly, the obtained rank is stochastic.

Tournaments are easily parallelizable and configurable. Parents selection also uses a binary tournament.

Initializer

Each parameter is initialized randomly within its respective range (see Table 1). For the four Hill's numbers, a random number is drawn from a uniform distribution between 1.0 and 4.0. Because we are in a biological system, for all the other parameters, the random value is drawn from a logarithmic distribution between the two boundaries. In practice, for example, k_{TR1} is fixed to 10^x where x is random value drawn from a uniform distribution between -3 and 3.

Crossover function

Different crossover functions were tested: a simple replacement called the parental replacement (a tosscoin decides whether the child will be a copy of parent 1 or 2), a barycentric crossover, a BLX-α crossover (Eshelman and Schaffer 1992) and an SBXv crossover (K Deb and Agrawal 1995) (Table 2). The crossover function is called for all children creation and involves 2 parents for the creation of 1 child. The 3 crossovers are summarized on Fig. 4 and explained in the following.

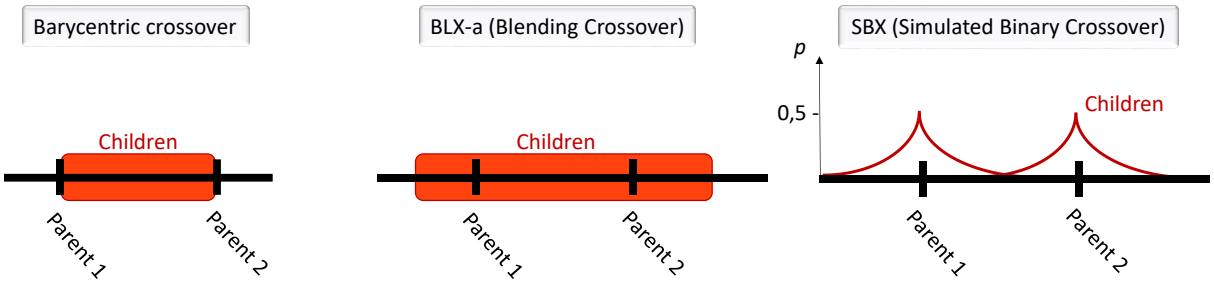


Fig. 4 Schematics of the different crossover operators used in this section. The red box shows the zone in which the child parameter can be created (with a uniform repartition). For the SBX crossover, the probability is represented on the vertical axis.

Barycentric crossover

For the barycentric crossover, a random value α is drawn between 0 and 1. The crossover is applied to every parameter x of the child to create as follows:

$$child_x = (1-\alpha) \cdot parent1_x - \alpha \cdot parent2_x$$

A new value for α is drawn for each parameter of the genome. The parameter of the child is therefore always comprised in its parents' parameters, increasing the convergence speed and reducing the explored space search.

BLX-α crossover

The BLX-α crossover allows to explore "beyond" the two parents, in both direction. BLX-α is parametrized by α. For each parameter, a random γ is created as follows:

$$\gamma = (1 + 2\alpha)\upsilon - \alpha$$

With υ a random value drawn between 0 and 1. The child parameter is created as follows:

$$child_x = (1 - \gamma) \cdot parent1_x - \gamma \cdot parent2_x$$

The value of α determines how far out of the limits of the interval $[parent1_x, parent2_x]$ the child parameter can be created. If $\alpha = 0$, we have a barycentric crossover.

SBX crossover

The SBX crossover is inspired from an operator applied to binary strings. This operator generates two children. A random value u is drawn between 0 and 1. According to this value, β is calculated as follows:

$$\beta = \begin{cases} (2u)^{\frac{1}{1+\eta}} & if\ u \leq 0.5 \\ \left(2(1-u)\right)^{\frac{1}{1+\eta}} & otherwise \end{cases}$$

Two parameters $child_{x1}$ and $child_{x2}$ are then created:

$$child_{x1} = 0.5 \cdot \left((1 + \beta) \cdot parent1_x + (1 - \beta) \cdot parent2_x\right)$$

$$child_{x1} = 0.5 \cdot \left((1 - \beta) \cdot parent1_x + (1 + \beta) \cdot parent2_x\right)$$

In our case, as we only create one child, one of the two parameters is randomly chosen. This operator gives a solution that has a high probability to be close to either one of the parents.

Mutation function

Different mutation functions were tested: a simple replacement (a parameter is replaced by a new value randomly drawn in the parameter's range), relative mutation (addition of a gaussian variation with a dispersion based on the nominal value) and self-adaptive mutation (from Evolutionary Strategies, see (Rechenberg 1974)). Self-adpative consists in letting the amplitude of the variation evolve as well. The mutation operator is applied to each *algogene* with a probability of 0.05%. If involved, the parameter sigma of the self-adaptive mutation is mutated whenever its corresponding *algogene* is.

Evaluator

To evaluate the individuals, a first step was to select 40 values of AHL concentration spread uniformly in the logarithmic scale between 10^{-4} to 10^{1}. Then for each of these values, the simulated GFP concentration is computed according to equations 1, 2, 3 and 4 fed with the parameters of the *algogene*. The simulation results are compared to the target response, which is approximated as a Gaussian (in the logarithmic space) centered on $\mu = -2$, of maximal height $GFP_{max} = 20$ and of standard deviation $\sigma = 0.2$ as follows:

$$f(x) = GFP_{max} e^{-\frac{(\log(AHL)-\mu)^2}{2\sigma^2}}$$

The evaluator computes the Mean Square Error (MSE) between all computed and expected points. The aim of the algorithm is to minimize the error.

In general, if any of the operators allowed the parameters to cross the boundaries used for initialization, an additional limiting step was applied to the child parameter.

2.3.3. Results

Table 2 shows the operators that were used in the results analysis: barycentric crossover operator, auto-adaptative mutation or gaussian noise, gaussian curve with 40 points for the evaluator. Stopping criterion was the number of generations. Analysis of the biological parameters is performed on the genome of the best individual after 5000 generations.

Table 2 Summary of the preliminary tests. Par.Replacement is a parental replacement. When not indicated, mutator operator was applied with a probability of 0.05%. Each score corresponds to the mean score of three to six runs after 1000 generations, with their standard deviation value. BLXα-x (respectively SBXv-x): x corresponds to the value of α (respectively v). Evaluation was performed with 40 samples taken uniformly in the log domain.

Crossover		Mutator	Boundaries	Average time (s)	Average score of the best
Operator	Domain				
Par.Replacement	lin	simple bounded	No	35	0.02 ± 0.007
Par.Replacement	lin	simple bounded 0.1%	No	34.8	0.04 ± 0.013
Par.Replacement	lin	relative (sig=0.2)	Yes	30.3	0.012 ± 0.005
Par.Replacement	lin	relative (sig=0.1)	Yes	32.4	0.006 ± 0.001
Par.Replacement	lin	relative (sig=0.3)	Yes	30.8	0.011 ± 0.003
Par.Replacement	lin	mut_autoad noise 1.0	Yes	32.5	0.05 ± 0.031
Par.Replacement	lin	mut_autoad noise sqrt(L)	Yes	28.5	0.02 ± 0.008
Barycentric	log	simple bounded 0.01%	No	39.9	0.08 ± 0.004
Barycentric	log	simple bounded	No	40.1	0.07 ± 0.002
Barycentric	log	simple bounded 0.1%	No	40.5	0.10 ± 0.009
Barycentric	log	simple bounded 0.2%	No	40.6	1.27 ± 0.555
Barycentric	log	mut_autoad noise sqrt(L)	Yes	41.8	0.04 ± 0.002
BLXα-0.1	lin	simple bounded	No	36.4	12.23 ± 10.56
BLXα-0.1	log	simple bounded	No	40.2	0.05 ± 0.005
BLXα-0.1	log	simple bounded	Yes	40.3	0.05 ± 0.006
BLXα-0.2	log	simple bounded	No	40.3	0.03 ± 0.001
BLXα-0.2	log	simple bounded	Yes	40	0.04 ± 0.005
SBXv-1	log	simple bounded 0.005%	Yes	35.2	4.15 ± 0.057
SBXv-1	log	simple bounded	Yes	35.3	2.66 ± 0.609
SBXv-2	log	simple bounded 0.005%	Yes	38.7	1.00 ± 0.867
SBXv-2	log	relative (sig = 0.1)	Yes	39.1	1.19 ± 1.309

The evolutionary algorithm has been implemented on the EASEA platform (P. Collet F. Krüger 2013; Collet et al. 2000). Computations have been performed on a standard computer without GPU acceleration. Several combinations of crossover operator and mutator have been tested. The main results are summarized in Table 2. Stopping criterion is the number of generations. Analysis of the biological parameters is performed on the genome of the best individual after 5000 generations.

The score reflects how far the function fed by the 22 parameters found by the algorithm is from the target function. Scores and computing time correspond to the mean of these values over 30 runs if not indicated otherwise. For clarity, only the first four runs are shown on Fig. 5 and Fig. 6.

Relevance of the parental replacement on a biological point of view

When using parental replacement as crossover operator, the results are the best in terms of computing time (158s) and score of the best individual ($9.91.10^{-3}$ corresponding to a relative error (score of the best individual over the height *h* of the peak) of 0.05%). Out of five runs, four parameters (k_{TR1}, d_{CI}, K_{R2}, n'_{R3}) systematically reached their lower boundary and one (n_{A1}) reached its upper one. Decreasing the mutator's Gaussian standard deviation did not alter significantly this tendency (out of four runs, the same parameters but k_{TR1} reached systematically one of their boundary).

Validity of the results

As shown on Fig. 5, the algorithm solutions always requires for K_{R3} (the repression constant of LacIM1) to be lower than K_{R2} by at least one order of magnitude. In the original work (Basu et al. 2005), this is also a requirement, which they solve by using CI as a strong repressor (to have a low K_{R2} constant), and as a weaker repressor. Similarly, we observe that the constants related to LacI and LacIM1 (K_{R3} and d_{LacIM1} and respectively K'_{R3} and d_{LacI}) are close for the two proteins. This is also the case in the original work. It is to be noted that the degradation constants of the proteins obtained by our algorithm are in accordance with the parameters used by Basu *et al.* to represent their actual system. As for the global synthesis rates (for a given protein, it corresponds to $\frac{k_{TR}k_{TL}}{d_{mRNA}}$), they are in the range of 0.1 µM.min^{-1} in our results and of 1 µM.min^{-1} in the original work.

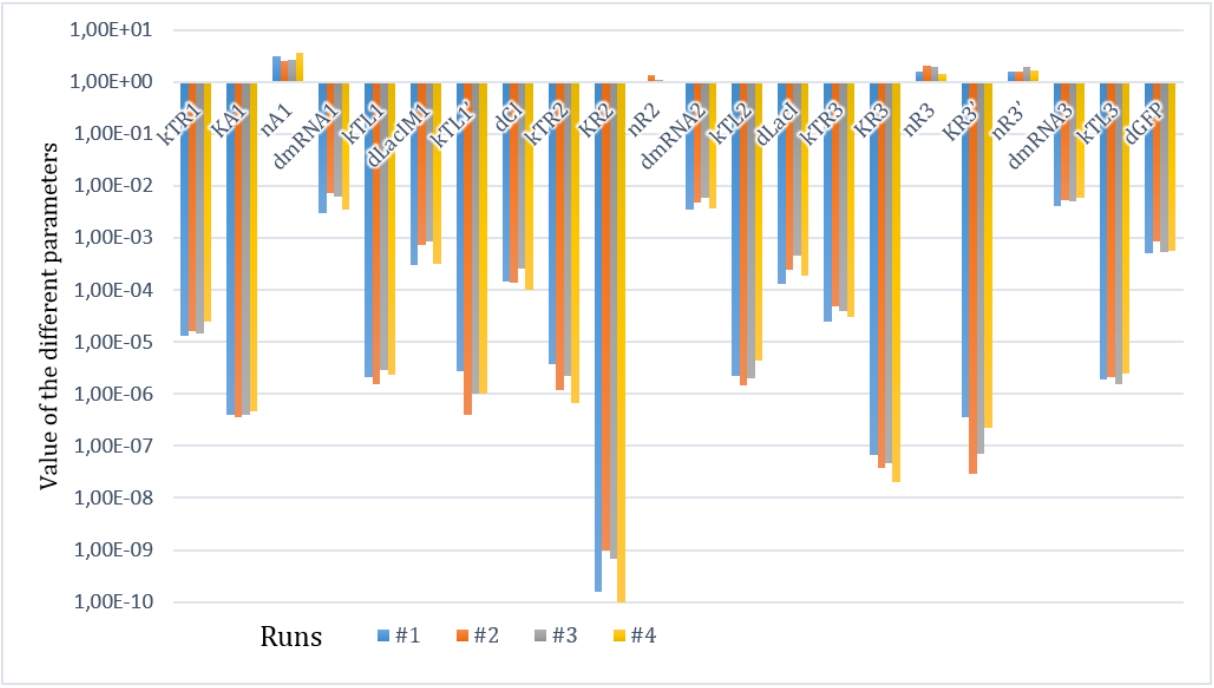


Fig. 5 Value of the *algogenes* for four runs after 5000 generations. The dissociation constants are given in M.s^{-1}, the transcription constants in , the translation and the degradation constants in s^{-1}.

The assets of self-adaptive mutation

Self-adaptive mutation offers the possibility to retrieve information about key parameters of the system. After the algorithm reached its stopping criteria, we retrieved the sigma values of the best individual. Out of 30 runs, less than 5% *algogenes* showed a sigma value lower than 0.2 whereas more

than 17% had sigma values greater than 0.3 (see Fig. 6). We observed that no *algogene* has a particularly low value of sigma. On the contrary, parameter k'_{TL1} has an average sigma value of 0.38, showing that it is less sensitive to variation.

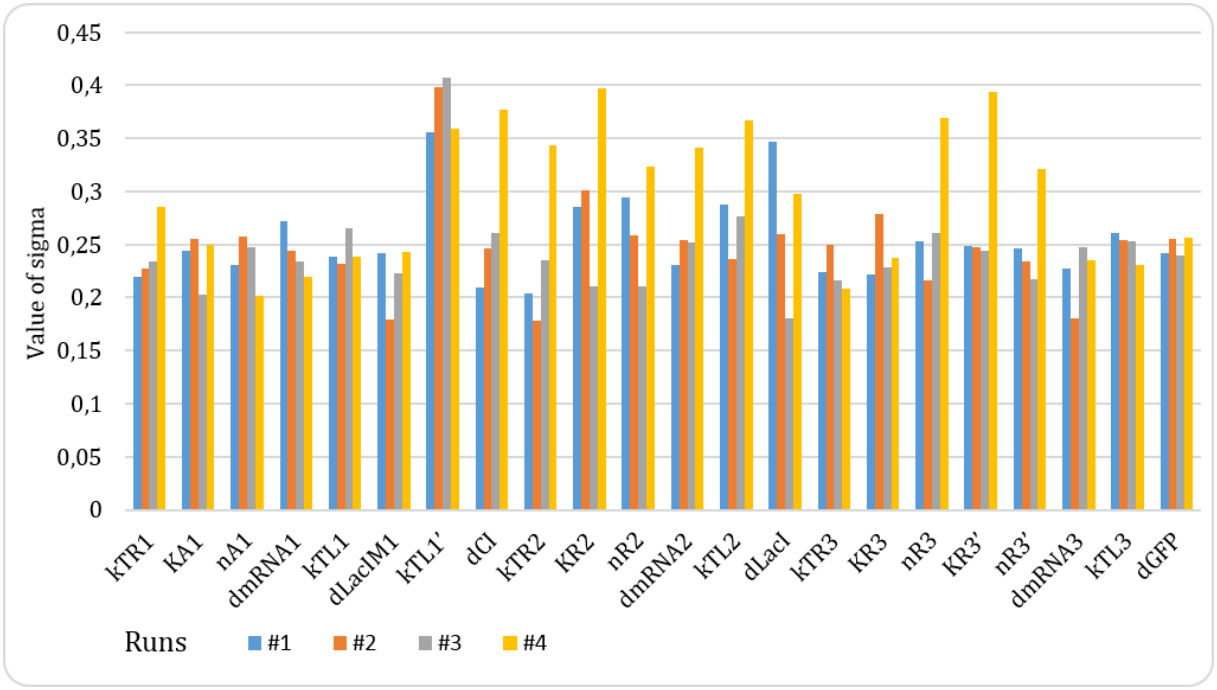


Fig. 6 Values of the self-adaptive mutator sigma values for four runs after 5000 generations.

Validation on different targeted responses

We tried to see whether the algorithm could solve similar problems with the same set-up by shifting the target function and increasing its peak (Fig. 7). Three other gaussian curves were tested: one with $\mu = 1$ and $GFP_{max} = 5$ µM, another one with $\mu = 10^{-1}$ and $GFP_{max} = 10$ µM and a last one with $\mu = 10^{-3}$ and $GFP_{max} = 40$ µM. In each case, relative error was under 0.3%, in 200s on average (see Fig. 7). For each peak, we took the average of each parameter out of three runs. It appears that most parameters are similar. Major differences can be observed for the dissociation constants of the regulators, namely activation constant K_{A1} and repression constants K_{R2}, K_{R3} and K'_{R3}. Apart for K'_{R3} where the tendency is opposed, they decrease with the height of the peak. The same is observed for n_{R3}. To observe whether this difference was due to the positional shift or the height difference, a new set of runs were performed with the same position for each peak (10^{-1} µM). The results showed that the tendency observed previously is absent. Moreover, the differences between the dissociation constants of each setup are negligible (they are non-existent or spread over less than one order of magnitude). No difference could be observed for the values of k_{TL3}, the translation constant of GFP. Only minor differences could be observed (when observed).

We also wanted to see whether this biological function was able to return a step-shaped response. The target function was changed to a step function with a value of 20 µM between $10^{-2.5}$ and $10^{-1.5}$ and 0 µM everywhere else. The algorithm found decent solutions (average relative error inferior to 2%) in 200s on average (see Fig. 7).

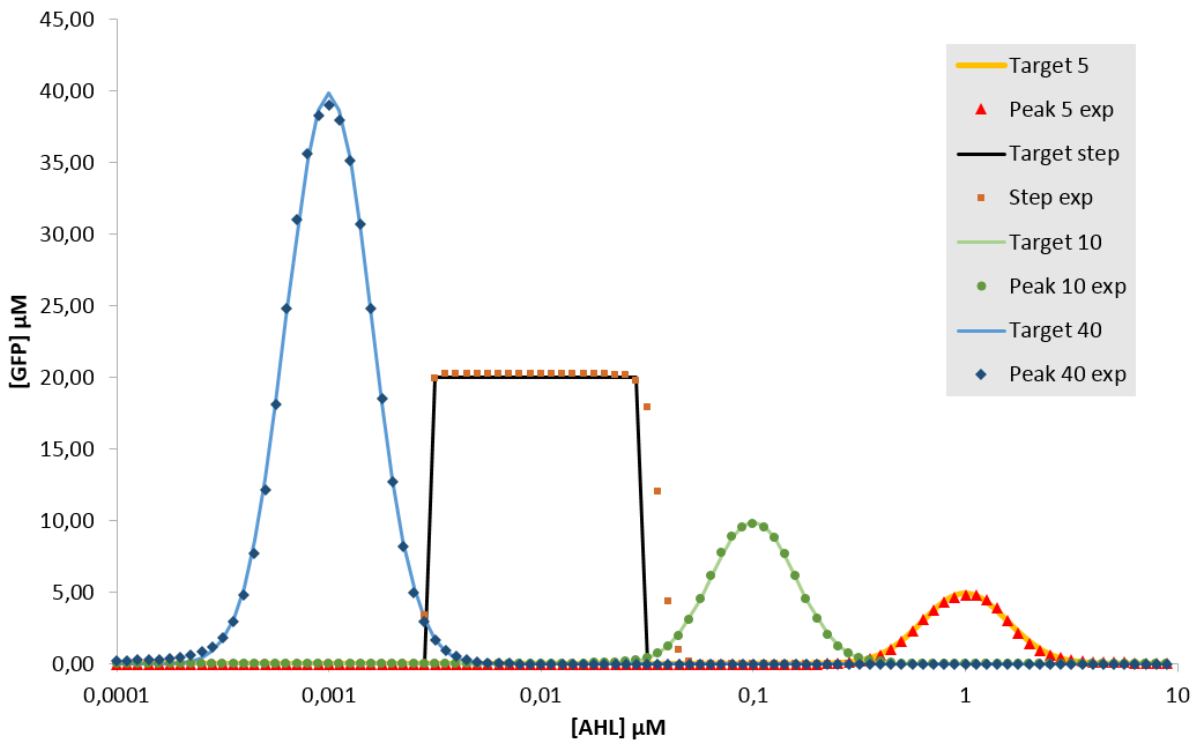


Fig. 7 Representation of the best individual's GFP response in function of [AHL] for different targets. The continuous lines show the targets and the points the corresponding results for three runs. "Target x" corresponds to a gaussian curve with height and "Peak x exp" the cognate results. For h = 40 (resp. 10, 5), the gaussian is centered on µ=10^{-3} (resp. 10^{-1}, 10^{0}). "Target step" corresponds to a step function centered on 10^{-2} with a plateau value of 20µM. The algorithm was run for 5000 generations with barycentric crossover and gaussian noise mutator.

Grouped evolution of algogenes

In nature, some *biogene*s coevolve: they are constrained by the same rules during evolution (Ehrlich and Raven 1964; Goh et al. 2000). We tried to reproduce this phenomenon by allowing the algorithm to divide the genome in *n* groups of *algogene*s (*n* varying from 1 to 22). In a group, the *algogene*s are crossed with the same parameters. From two to five groups, the scores are higher than the one group-algorithm. The algorithm is converging prematurely. Above five groups, compared to the one group-algorithm the scores are similar. However, the algorithm requires 1.15 more time when using groups.

We also tried to run the algorithm with autoadaptive groups. Each individual had a random number of groups, composed of *algogene*s randomly picked. Group setup of each individual was allowed to mutate. Crossover of two individual creates a child with a number of groups of one of his two parents. Results were neither conclusive in terms of score, nor in terms of biological interest: it was not observed that specific *algogenes* had a higher tendency to form a group together.

2.3.4. Discussion

The obtained results show that the algorithm can find coherent solutions to a biological problem. Indeed, the original work emphasizes some specificity of their system (significant differences in the repressor constants) which the algorithm successfully shows. As the process contains modifying operators (crossover, mutation), the parameters could have virtually taken any value in their respective allowed range. Interestingly enough, all the values obtained were biologically relevant and consistent with the original paper.

Self-adaptive mutation was expected to be an additional hint towards which elements of the system are of key importance in the correct realization of the expected function. However, no *algogene* showed a particular constraint (low sigma value). The general tendency for sigma values to be rather above average than below is due to the way the sigma is mutated. Indeed, its value is multiplied by the exponential of a random number drawn from a Gaussian distribution centered on 0.0 with a variance of 1.0. The high value of k'_{TL1} (translation of CI) sigma suggests that this parameter is allowed to vary from the returned optimized value. This would give the biologist more freedom regarding the promoter he needs to use.

Running the algorithm with other targets revealed key parameters to set the desired position of the peak. These parameters are dissociation constants of regulators, which makes sense since these parameters control the sensitivity of a promoter towards its regulator. Two groups can be distinguished. K_{A1} and K_{R2} allow to shift the peak, and K_{R3} and K'_{R3} to sharpen it. Indeed, a decrease on K_{A1} will lead to a higher sensitivity of operon #1 promoters towards AHL, so that a lower [AHL] will be required to initiate $LacI_{M1}$ and CI expression. This leads to an increase of GFP at lower [AHL], namely a shift of the rising edge of the peak to the lower concentrations. Similarly, a decrease on K_{R2} will increase the sensitivity of operon #2 towards CI, resulting in a stronger repression of LacI by CI: the falling edge of the peak will shift to lower [AHL]. Finally, K_{R3} and K'_{R3} control directly GFP expression and therefore act on the thinness of the peak. Indeed an increase on K'_{R3} decreases the sensitivity of GFP's promoter towards CI. The rising edge of the peak will therefore be shifted towards lower [AHL]. Seemingly, a decrease on K_{R3} will result in a shift of the falling edge of the peak towards lower [AHL] as well. This can be verified by simulating GFP levels in function of [AHL] while varying the above mentionned constants (data not shown).

We can expect that a change in GFP peak height only would involve a similar change in GFP translating constant. As it is not the case, we investigated the product $\frac{k_{TL3}k_{TR3}}{d_{mRNA}d_{GFP}}$ (see Equation 4), which indeed grows proportionally to the height of the peak. The algorithm is therefore capable of finding non trivial solutions regarding this biological problem.

With the possibility for the *algogene*s to evolve in groups, we expected to observe the appearance of groups of linked parameters. Indeed, as described above, the maximal $[GFP]$ depends of the constants k_{TL3}, k_{TR3}, d_{GFP} and d_{mRNA3}. When the algorithm found a good individual, if k_{TL3} and k_{TR3} (respectively d_{GFP} and d_{mRNA3}) evolve in the same direction (or not at all), the individual should keep its good score. We therefore expected for such parameters to tend to gather in the same group. This tendency was not observed.

To further validate the strength of this approach, a comparison with other methods (such as particle swarm or simulated annealing) could be carried out.

2.4. Results obtained on a XOR gate

To see whether this approach could be generalized to other types of genetic networks, we tried to optimize a bio-logic XOR gate with a similar algorithm. The complete description of the biological system can be found here (Ausländer et al. 2012). A simplified illustration is given on Fig. 8. Phloretin (Ph) and Erythromycin (Er) are the input of the system while YFP (Yellow Fluorescent Protein) is the output of the system. In the presence of both inputs, mRNA are not synthesized. Thus YFP is not produced. If there is no Ph nor Er, both mRNA are synthesized but inhibited because they are bound

with L7 (for mRNA1) and MS2 (for mRNA2). Finally, when Ph (respectively Er) is present alone, mRNA2 (resp. mRNA1) is produced but not MS2 (resp. L7). As a consequence, mRNA2 or mRNA1 can be translated and YFP is synthesized.

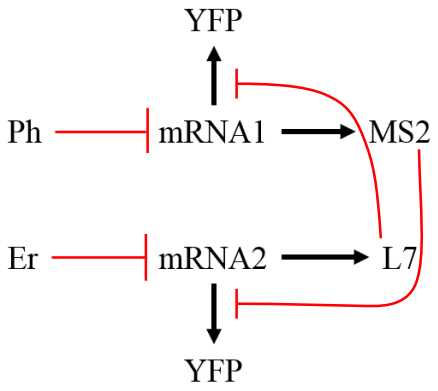


Fig. 8 Simplified XOR bio-logic gate. Barred red lines indicate a repression on the operon coding for the corresponding mRNA; red dashed arrows indicate a binding reaction; black heavy arrows indicate production. Ph: Phloretin. Er: Erythromycin. YFP: Yellow Fluorescent Protein.

The equation set that models this system is composed of 15 parameters (2 transcription rates, 2 dissociation constant and 2 Hill's numbers for Ph and Er repression, 2 dissociation constants for MS2-mRNA2 and L7-mRNA1 binding reaction and 3 translation rate and 4 decay rates).

Evaluation is performed by taking the MSE between the value of this function and the target function on 10 values of $[Ph]$ times 10 values of $[Er]$, spread uniformly on the log domain between 10^{-9} and 10^{-3} µM. The target function is a classic binary XOR, with low plateaus being set at 10^{-12} µM and high plateaus at 10^{-8} µM. Threshold value is set at 10^{-6} µM for both inputs (namely Er and Ph). The algorithm was able to produce a set of parameter with a score around 10^{-17} (Fig. 9). With the same network topology we also use the algorithm to obtain alternative Boolean functions, namely an NAND gate (no YFP when both Er and Ph are present) and an INH gate (YFP is produced when Er is present and Ph is absent).

2.5. From an evolution strategy to genetic programming

This work demonstrates the relevance of evolutionary algorithms in synthetic biology, in particular in the field of biological networks. Being able to optimize the parameters of a defined system before going to the bench is an important speedup in the process of biological networks design. Indeed, biologists typically have to test various sets of components before finding the most suitable. Such an algorithm gives hints on which parameters are of key importance and which kind of components should be used (*e.g.* a strong or weak repressor). Moreover, as this tool is capable of generating sets of parameters realizing a large variety of functions, it can also help to understand the specifics of a system, by varying the target function used in evaluation and analyzing the returned parameters. This modularity enlarges designers' horizon by giving them the possibility to preliminary test in silico any (crazy) idea they might have.

Our initial goal is to design an algorithm able to create a relevant biological system (a GRN here) from scratch to answer the biologist's problem. Two tasks are still not adressed. First, the research into a device library (typically the Biobricks library ("Registry of Standard Biological Parts" 2015)) of the closest components to the returned parameters has always to be done by hand. Second, to use our

algorithm, the biologist still has to imagine the biological system's architecture before 'feeding' it to the algorithm. Genetic Programming could be used to automate this step.

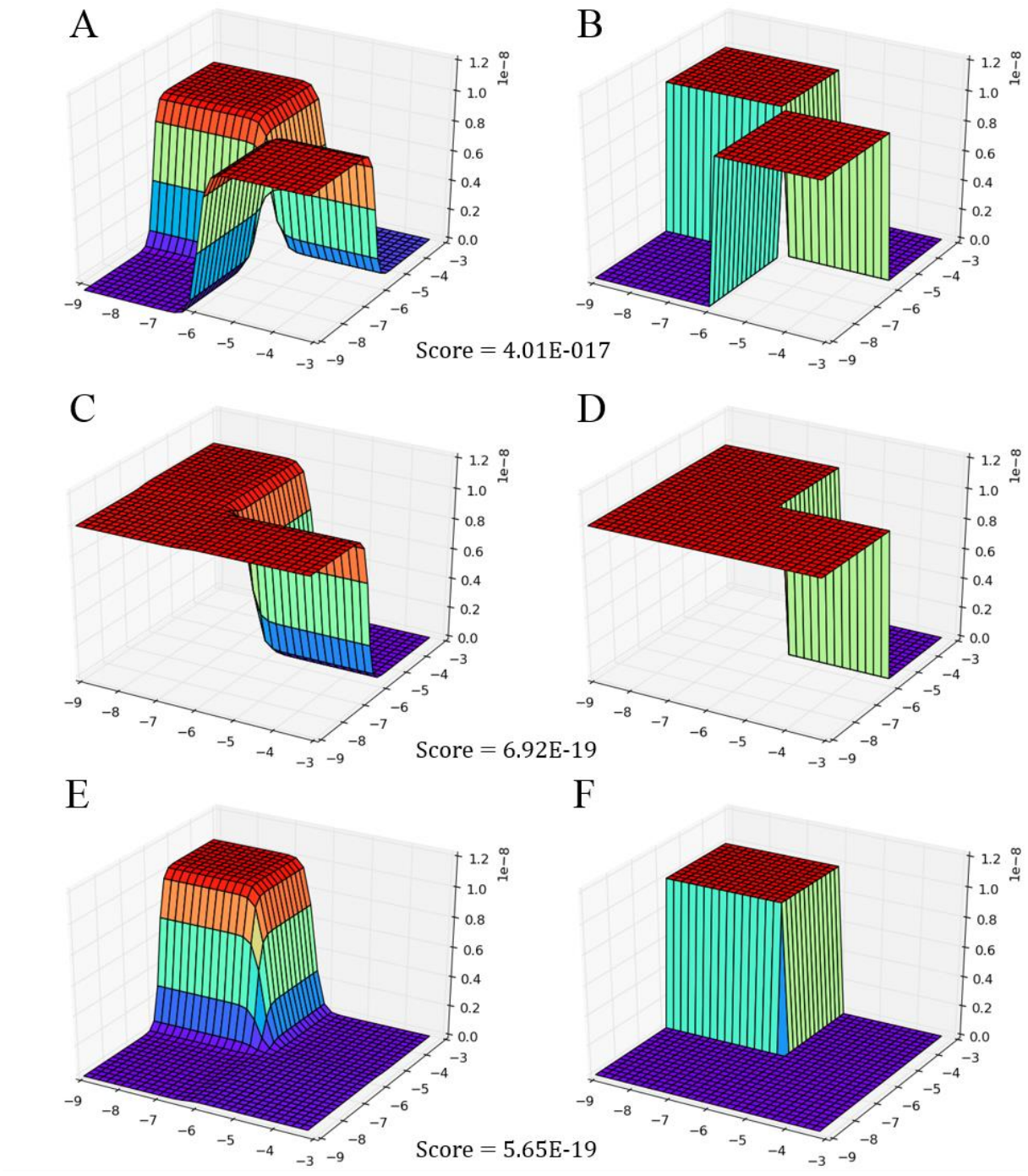


Fig. 9 [YFP] in function of [Er] and [Ph] for three different targets. B (resp. D and F) shows the target function for a XOR (resp. NAND and INH) gate. A (resp. C and E) corresponds to the results for the target B (resp. D and F). Score corresponds to the fitness score of the best individual for each target.

3. Evolving the GRN topology

3.1. Introduction to genetic programming

Genetic programming (GP) is a form of artificial intelligence as it is able to find solutions humans would not be able to find (Hornby et al. 2018). They were initially designed to evolve functions. More generally, they are able to manipulate genomes of variable sizes. In comparison to inverse problem (see Section 2), GP evolve both the function f and the parameters x in order to find f_0 and x_0 such as $f_0(x_0) = \tilde{y}$. Attributing a fitness score to an individual is done by executing (running) the individual. In our case, this refers to simulation. GP can be used on constructions, like GRNs.

The first apparition of GP dates back to 1958 with Friedberg's self-modifying algorithm. Pioneering work was published by Cramer (Cramer 1985) in an article that deals with most elements used in GP algorithms. His algorithm manipulates a variable-size genome in form of a graph and applies different operators like mutation, crossover and self-recombination. The major step in this domain was accomplished by Koza and his massive computer power in the 1990s (Koza 1992; Koza 2003; Koza and R. 1989; Koza 1994; Koza, Bennet III, and Andre 1999). His books on GP are foundational and his algorithms led to the submission of patents on electronic analog circuits. In a biological context, François and Hakim used GP to design a bistable switch and an oscillating GRN (François and Hakim 2004). An algorithm based on their work is described later in this chapter. The work of Patil *et al.* also deserves to be highlighted (Patil et al. 2005). They applied an evolutionary programming based method on metabolic pathways of the whole yeast Saccharomyces cerevisiae. Their method suggested non-intuitive metabolic engineering strategies for optimizing an industrial fermentation process.

Our ultimate goal here is to design the topology of a GRN and find the optimal parameters of each of its components (according to a set of requirements).

3.2. Genetic programming with generic functions

GP algorithms can create and optimize mathematical functions represented as trees. We first try to evolve a mathematical function and then map its elements to biological functions. We apply this approach on a XOR gate and a band pass system. Then, we changed the genome representation and opted for graphs (to allow feedback loops) and fed the algorithm with prepared biological functions only. This work is presented on the example of a band pass target function.

In what follows, tournament *x* refers to a tournament selection operator with a selection pressure of *x* (when the selection pressure is an integer over 0, the best individual from the *x* will be selected; when the selection pressure is a real number between 0.0 and 1.0, the best individual of 2 will be selected with the probability *x*).

3.2.1. XOR gate

In this first example, we only used Boolean operators. This simple example is a good entry point for testing GP. Moreover, it is interesting to evaluate GP as an alternative to GeNeDA. Indeed, the stochastic nature of GP algorithms enables them to find different solutions for a design where GeNeDA has a deterministic solution based on the library used for the design.

We therefore use a NOT, an AND and an OR operator. The target is set to a 2-inputs XOR gate.

Algorithm parametrization

A generation is composed of 5000 individuals. A strong elitism with 1 elite (number of surviving parent) is used to select with a tournament 2 after the creation of 5000 children. The mutation and crossover probability are 0.1 and 0.9 respectively. The selection operator is a tournament 7. The stop criterion is the number of generation which is fixed to 50. Initialization of the tree is performed with a ramped half-and-half method with a depth of tree comprised between 2 and 4. That is to say that half of the initial trees are fully grown (with a depth of 4) whereas the others are grown with a depth between minimal and maximal depth value (Koza 1992). The maximum tree depth authorized during the iterations is 8. The individuals' functions were evaluated on the 4 cases of the truth table. The fitness score is the number of cases where the individual does not provide the right output.

Results

The different equations obtained for 6 runs are summarized in Table 3.

Table 3 Different versions of a 2-input XOR gate.

Run #	Equation	NOT	AND	OR	Total
1	$(A+B)\cdot\overline{A\cdot B}$	1	2	1	4
2	$\overline{B\cdot A\cdot B}\cdot(A+B)$	1	3	1	5
3	$(\bar{B}\cdot A)+(B\cdot\bar{A})$	2	2	1	5
4	$(\bar{B}+\bar{A})\cdot(A+B)$	2	1	2	5
5	$\overline{\bar{\bar{B}}\cdot A}\cdot(A+B)$	3	2	1	6
6	$\overline{B*A*A}*(A+B)$	1	3	1	5

All runs achieved a score of 0 for the best and also the whole population. Indeed, all solutions correspond to a XOR gate. We note the variety of solutions found by the algorithm. No pressure was put on the number of operators used (apart from the maximal tree depth), so that some equations employ useless terms ($\overline{B\cdot A\cdot A}$ is the same as $\overline{B\cdot A}$).

We also note that the term $(A+B)$ appears in all but one solution. As the OR (+) operator is in itself very close to the XOR gate (3 of the 4 inputs states show the same result), it is expected that this term stands out.

The result of the run #1 is also very interesting. XOR expression suggested by GP used 4 operators, which is one less that the number of operator used in the standard XOR formulation ($\bar{A}\cdot B+\bar{B}\cdot A$) which would be almost systematically suggested by humans, even the specialists of digital electronics.

Such results are easily readable and implementable. Compared to GeNeDA, this method has the advantage of offering several possible variations of the same GRN function, sometimes even very complex or redundant, instead of offering only the simplest one. On the other hand, the stochastic search of the solution is much less efficient than the formal research carried out by GeNeDA in terms of computing power. We now want to go further, i.e. evolving GRN by taking into account their analog behavior and try to match targets described by an analog behavior.

3.2.2. Band-pass system

To that end, we use as a target function the Gaussian target of the Basu system studied in Section 2. Here we allow the algorithm to use the classical mathematical operators (addition, subtraction, multiplication, division and power) used in real-valued equations. The parametrization of the algorithm is the same as for the Boolean approach except that the initial maximal tree depth is 6 and the maximal one is 12, the number of generations is increased to 100 and the population size to 20,000. Evaluation is performed by comparing the response of the generated system to Basu Gaussian target on 1024 random AHL concentrations which decimal logarithms are uniformly spread between -4 and 1. Taking the evaluation points randomly ensures that the solutions found by the algorithm do not match specific points in particular but the whole target in general.

The algorithm is able to find a function that fits almost perfectly. However, the evolved functions (data not shown) involve a large number of arithmetic operators and do not relate to any known biological function: it is therefore impossible to build a GRN accordingly.

GP algorithms allow the implementation of Automatically Defined Functions (ADF). To populate a node, GP can either choose between an operator and a terminal (a value or a variable). With ADFs, the user can define a custom function that the algorithm can choose as a third option.

This concept is reused in the following work. Indeed, it appeared that trees were not the ideal way to represent GRNs, as GRNs often imply feedback loops.

3.3. Genetic programming with ADF

Another way to represent the structure of a GRN is to use graphs, with each node being a molecule and each edge a reaction. Predefined functions, namely the ADFs, are embedded in each reaction. The parameters of the reaction however may be modified. The algorithm we want to elaborate would therefore be able to build a network according to a list of predefined biological reactions and optimize its structure and parameters to fit the target response. The target response is a dose-response curve in the continuous domain.

3.3.1. Rules of construction of the network

The network build by the algorithm follows the following guidelines:

1. there is at least one input, and all inputs have to be directly or indirectly connected to the only output;
2. a gene or a set of genes cannot be strictly independent from the main network of the output;
3. a gene produces only one protein and a protein has always at least one function, such as an interaction with a gene or with another protein to form a complex;
4. a complex always has two predecessors;
5. if a complex has no function, one of its predecessors has at least 2 functions (the complex formation and another). This complex has a role similar to a negative regulator in that its formation sequesters its predecessor (the binding consumes the reactants), thereby reducing their concentration.
6. two nodes can only be connected by one regulation and a node cannot have an interaction on itself (as DNA is differentiated from the protein it produces, a protein may still influence its own production).

7. Other limitations can be fixed: the number of proteins interacting with a same gene and maximal number of genes in a network.

During the course of the algorithm, addition or deletion of a node or an edge is also subject to specific rules:

1. If a node producing a protein is added, the protein has to have a function;
2. after the addition of a complex, the choice of the targeted node (the destination of the edge from the complex) cannot be another complex to avoid the formation of a grid of complexes in one round;
3. a protein may only be deleted if its coding gene is deleted;
4. an input or an output cannot be added or deleted by a mutation;
5. if a gene or a complex is removed, every predecessor of the node connects with every direct successor of the node and the parameters of these new interactions are copied from the old node-children interactions;
6. if a sub-network becomes independent after a deletion, it is removed if it is not possible to make a link with the successors of the deleted node or if this last has no children;
7. if a complex is removed because one of its predecessors is removed, the remaining predecessor forms a link with the children, even if they are complexes;
8. If there are creations or modifications of a node during a deleting mutation, the parameters which have to be defined will be copied from the deleted node or edge.

All functions must have these behaviors. It is important to understand that the nodes called "complexes" are not biological complexes but a representation of the interaction of two other nodes. So a "complex" could be a complex or the modification of a protein by another, like a phosphorylation or an ubiquitination.

3.3.2. Reaction parameters

As usual, a differential equation solved at the steady state is computed for each species. In this work, mRNA was not taken into account. This differential equation comprises several parameters that are held in the different reactions involving this node. Hence each node and each edge is represented by the following according to its type:

- each node but genes has a quantity in µMol. This quantity is constant over time for the inputs;
- for genes and the output: the maximal and minimal productions of a protein by its gene, in µMol.s^{-1};
- for edges from proteins, complexes and inputs: the dissociation constant (in µMol) and Hill's number of its binding to associated promoters;
- for complexes: the stoichiometric coefficients `a` and `b` for the formation of a complex such as `a*A + b*B = AB`;
- for complexes: the reaction constants `kon` and `koff` for the complex binding and dissociation respectively such as `A + B -> AB (kon) -> A + B (koff)`;
- for proteins, complexes and the output: the degradation constant, in s^{-1}.

These parameters are optimizable by the algorithm. However, we limit this variation to a given domain (see Table 4). P_{max} corresponds to the maximal expression rate of the protein (if mRNA was integrated in the model, it would correspond to the ratio $k_{tr} * k_{tl}/d_{mRNA}$). α is a number between 0 and 1 and

corresponds to the leakiness of the promoter: this allows a reduced expression of the gene even if the promoter should be off (inhibited or in the absence of activator).

Table 4 Biological reactions parameters and their range of variation.

Parameters	Variation domains
P_{max}	*[10e-7, 10e-1]*
α	*[0,1]*
K	*[10e-4, 10e+3]*
Hill's number	*[1, 4]*
Coeff. Stoech.	*[1, 5]*
k_{on}	*[10e+4, 10e+8]*
k_{off}	*[10e-6, 10e+5]*
d_Y	*[10e-3, 10e-1]*

The differential equations are the classical expression used for representing protein expression and complex binding (see Chapter 2) with the following modification: as mRNA is not represented, it is considered at its steady state and parameter P_{max} represents the ratio $\frac{K_{tl}K_{tr}}{d_{mRNA}}$.

3.3.3. GP algorithm set up

We adapted our algorithm from the work of P. François and V. Hakim (François and Hakim 2004). We consider a population of N networks. At the initialization step, there are N identical networks with one or more inputs acting on a gene, producing a protein which activates the output. The latter is a gene coding for a protein which is not represented in the graph and so the output node must be considered like a set of two nodes (the gene and its protein).

Mutation

At each generation, a copy of each network is mutated and added into the pool with the other networks, giving a population of 2N elements.

Mutations can be of six different types:

1. **m1:** random change of a parameter;
2. **m21:** removal of a gene or a complex;
3. **m22:** removal of an interaction of a drawn node;
4. **m31:** addition of a gene and with a link to a chosen node;
5. **m32:** creation of a new interaction of a protein/complex on a gene;
6. **m33:** creation of a new complex between two proteins and/or complexes.

Each mutation has a different probability to be chosen, with `P(m1) > P(m21) > … > P(m33)`. The sum of all probabilities is 1. During the last generations of a run, the algorithm focused on optimizing the parameters only and sets the probabilities of m31 to m32 to 0 (the other probabilities are updated so that the sum still equals 1).

The network is mutated a number of times proportional to the number of nodes in the network. The proportion is fixed by the user but must be above 1. A maximal number of mutations per networks can be fixed.

For each mutation, a node is chosen according to its age. The age of a node corresponds to the number of generations since its creation. The goal is to mutate aged nodes preferably so that young nodes that

were just created are not deleted right away. This release in selection pressure allows the network to evolve around this new node before being pressured. To that end, the nodes are sorted by age and divided into 5 groups. A group is randomly drawn according to an exponential probability law and a node from this group is randomly picked. The equation of probability for a group G_k is:

$$P(G_k) = \frac{N_k}{\sum_{i=1}^{M} N_i} \cdot \gamma \cdot e^{\gamma \cdot k}$$

With M the total number of groups (5 in our case), N_k the number of elements belonging to G_k and γ a parameter of the exponential law. $\gamma = 0.8$ was a good compromise between a very high probability for the oldest group to be chosen and a uniform probability; with this value, the probability of the oldest is 0.5 and second oldest 0.25. This probability $P(G_k)$ is then divided by the sum of all probabilities.

Fitness score

Currently, the following two scores are computed:

- **The output score**, which is the mean squared error for each specified test point of the dose-response curve (which can be of more than 2 dimensions if there are many inputs).
- **The complexity score**, derived from the cost function used in GeNeDA. This score equals to $\sum_{i=1}^{N} 1.6^{A_i+R_i-1} \cdot 1.25^{A_i} \cdot 1.25^{C_i}$ for N nodes, with A, B and C the number of activating edges, repressing edges and protein-protein interactions from the node *i* respectively. The genes and the output are not considered because they don't have these types of edges.

The networks are then ranked with Pareto's algorithm. Pareto's algorithm is a way to find the optimal solutions of individuals evaluated on more than one criterion. To that end, for a set of solutions, we find the Pareto front. This Pareto front regroups the solutions that are not dominated by any other on all criteria. On the example on Fig. 10, we want to minimize scores $f1$ and $f2$. We see that all solutions that have a better (lower) $f1$ than solution A (all solutions below A, including solution B) are dominated by A on $f2$. Moreover, all solutions that dominate A on $f2$ (solutions on its left) are dominated by A on $f1$. There is no solution that dominates A on both criteria at the same time, A belongs to the Pareto front. The symmetric reasoning can be applied to B, which also belongs to the Pareto front. For C however, we can find at least on point that dominates C on both $f1$ and $f2$ (A for example). C does not belong to the Pareto front.

In our case, the two scores used for the Pareto front are the output score (or MSE) and the complexity score. Rank 1 is attributed to the solution in the Pareto front which are then removed from the pool of solutions to evaluate. These steps are repeated for the remaining solutions (the new Pareto front is computed on this reduced pool of solution and rank 2 is attributed to the newly found Pareto front).

Selection for the next generation

The population is reduced until reaching the initial size N of the population. Two phases are distinguished.

During the first phase, the networks are allowed to grow because their complexity is not taken into account in the selection process. The networks are selected with a tournament of the size of half of the population based only on the output score.

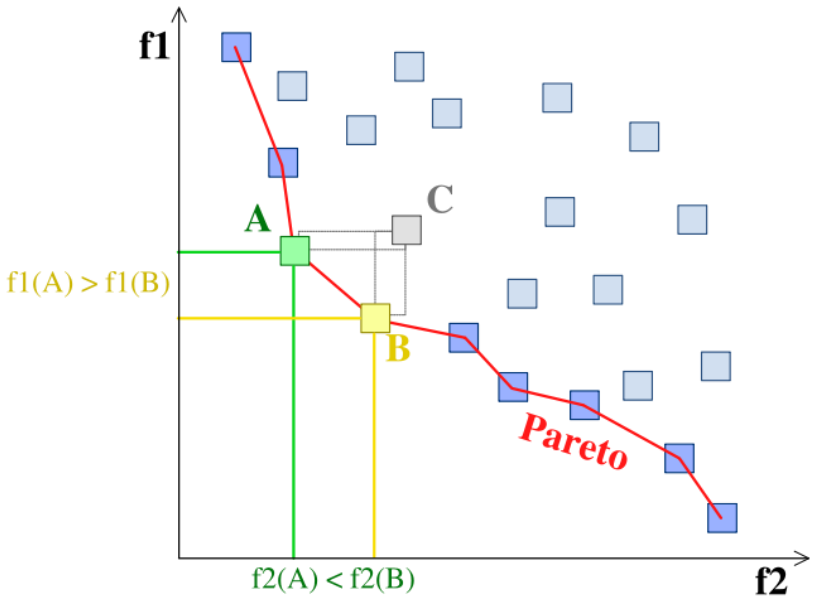


Fig. 10 Pareto front (https://commons.wikimedia.org/wiki/File:Front_pareto.svg)

During the second phase, the networks are deterministically selected based on their rank obtained with the Pareto algorithm. Beginning with rank 1 and ascending, all solutions belonging to a rank are selected for the next generation. If adding a rank contains too many solutions (the size of the selected group would exceed N), solutions from this rank are randomly selected. The corollary to that is that the higher the number of ranks, the lower the number of individuals chosen randomly because the ranks contain less solutions. To achieve a higher number of ranks, it is possible to introduce a bias in the selection operator. For each Pareto front, the solutions are ranked according to the MSE score: the first half has a rank that is higher than the second half, resulting in twice as more ranks than without the bias.

3.3.4. Results with a 4-points target

For the mutations probability, two steps are distinguished. In the first step, mutation probabilities are set to their default values. In the second step called the pruning step, these probabilities are altered so that only deletion of a node or modification of the value of a genetic parameter is authorized. Without the possibility to add a new node as a mutation, the network cannot grow in this pruning step. Combined with the use of the complexity score, the idea is to only keep the core nodes of a network in the final rounds of the algorithm. In this example, complexes are not authorized (the probability m33 is set to 0 in the first step already). By default, we used the biased selection operator.

Default parametrization for the algorithm is described in Table 5.For each case described thereafter, 5 repetitions of each run are carried out and the best solutions are described.

The target was defined with 4 points: two low points at input values of 1e-05 and 1000.0 where the expected output is 0 and two high points at input values of 0.01 and 0.5 where the expected output is 30. The idea is to obtain a network that behaves like a band-pass.

End of the growth phase and start of the pruning step

We first assess the effect of the growth phase and the pruning step. The tuning of the length of these periods influences the pressure put on the networks to have a low complexity. If this pressure is too high, we might obtain very simple networks at the expense of the MSE score. On the other hand, without limit the networks might grow to the maximal authorized size to minimize the MSE score. We

therefore try different combinations of growth phase stop time and pruning step start time (in generations).

Table 5 Default set-up of the algorithm

Parameter	Default value
Number of generations	50
Population size	20
Start of the pruning step	45
End of the growth phase	45
Number of mutations per node	2
Max. number of mutation per network	2
Default probabilities : m1 m21 m22 m31 m32 m33	 0.5 0.2 0.1 0.1 0.1 0
Pruning step probabilities : m1 m21 m22 m31 m32 m33	 0.8 0.1 0.1 0 0 0

Default values

First of all, we run the algorithm with the default values (Fig. 11). The end of the growth phase corresponds to the start of the pruning step at generation 45. For most of the runs, the average complexity increases until the start of the pruning phase, as expected. In some rare cases, the average complexity increased (see the column on the right on Fig. 11). This could mean that the pruning step mainly produced less performing networks that were not selected, increasing the concentration in complex networks.

In all the cases, we noted a global decrease in the MSE score starting around generation 40. This decrease was not impaired by the end of the growth phase and start of the pruning phase, meaning that these steps are not deleterious. The five best networks of each run were mostly very simple, with a number of nodes and edges rarely exceeding 10. The MSE scores of these networks however spanned across a large range: between 0.038 and 15. The complexity score did not exceed 10. For each run, these five best networks are very similar with one another: if at least one network shows a band-pass behavior, the other 4 networks also did. This shows that when the algorithm found a good solution, it converged toward it. However, due to the stochastic nature of the algorithm, in some runs no band-pass behavior was observed: in these cases, the algorithm converged toward a simple activation of the output by the input.

Most of the networks performing a band-pass showed the same core features: an incoherent feed-forward loop. This means that the input was both leading to activation of the output and inhibition of it (see the two schematics on Fig. 11). This behavior is very similar to the Basu system described previously (Basu et al. 2005). However, our algorithm proposes here a system with less genes.

When looking at the parameters, we first observe that the P_{min} are all very low compared to the P_{max} (more than 3 orders of magnitude). The leakiness is not to account for in the behavior of the system. Moreover, in both runs shown on Fig. 11 (and in most cases) the sharp increase noted in the concentration of output at around 10^{-3}-10^{-4} input is due to the high Hill coefficient of the output activation by protein 1.1 (above 3). In both cases, the threshold (dissociation constants K) for 1.1 activation and the output inhibition by the input differ by at least 2 orders of magnitude, with the activation being realized for lower values of inputs. This difference in parameter values is key for enabling the band-pass behavior as without it, both activation and repression would proceed for the same values of inputs and the output would never be expressed.

Noteworthy is the self-inhibition of protein 1.1 in the run on the left on Fig. 11. First of all, as its dissociation constant is 20, it only serves to prematurely lower the concentration of output, resulting in a lower MSE score of this network. Moreover, this protein is both an activator and a repressor. As an improvement on the current version of the algorithm, a new constraint could be added, similar to the one existing in GeNeDA: a protein cannot be both an activator and a repressor. For the time being, we can consider that in such cases, a buffer protein would be added to split the roles of activators and repressors on two different proteins.

Premature end of growth phase

The end of the growth phase was set at generation 40 (instead of 45). In most of the 5 runs (4 out of 5) the end of this phase coincided with a decrease in the average complexity and MSE score of the population (see the orange triangle on left diagram of Fig. 12). The results are comparable to the default runs in the number of band-pass system generated by the algorithm, the MSE score span (3.8 to 20) and the GRN performing a band-pass (both in terms of topology and parameters scale).

Premature start of the pruning step

The algorithm was run with the default values but the start of the pruning step was step to generation 40. Overall, the average complexity score of the population shows a slight tendency to decrease at generation 45 (orange triangle on the right diagram of Fig. 12) which corresponds to the end of the growth phase. The average MSE score however consistently decreased after generation 45.

The runs generated a similar number of band-pass systems. However, many of these band-pass did not match the high target points (at 30) and were below, hence these systems had a high MSE score. Moreover, the generated band-pass GRN were in general more complicated (with at least on more gene) than for the previous cases (Fig. 13). These networks show the same core networks with the intermediate gene 1.1 than for the previous runs. The self-inhibition is however replaced by a self-activation. As the core function does not rely on the supplementary genes (shown in grey on Fig. 13), this set-up proves to be less efficient in reducing the complexity of the generated solutions.

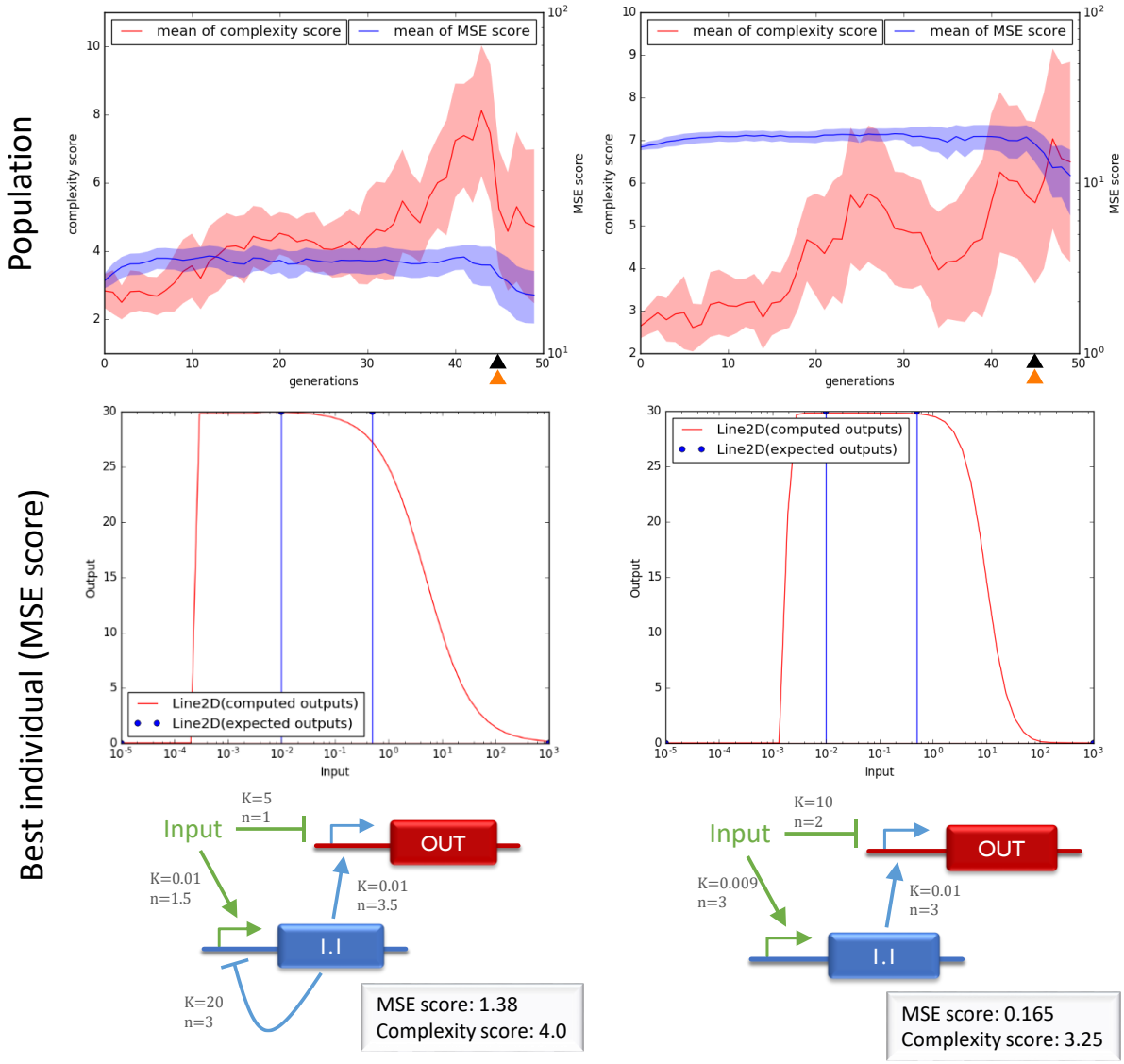


Fig. 11 Results of default runs. Each column corresponds to a run performed with default values. First row shows the evolution of the average MSE (blue) and complexity (red) score with the standard deviation across the population. Second row shows the expected outputs (blue dots) versus the simulation of the best network of the run (red curve). The best network is the network in rank 1 with the lowest MSE. In the third row, the schematics of the best network and its scores. Orange triangle: end of the growth phase. Black triangle: start of the pruning step. Near each regulation (activation or inhibition) the dissociation constant K and the Hill number are shown.

Considering the results, we find that the end of the growth phase is the most critical parameter. As both the default run and the premature end of the growth phase provided good results, both set-ups were reused.

Increasing the number of generations

To see whether evolving the networks for a higher number of generation would give better results (at the expense of computing time) we increased the number of generations to 150. The start of the pruning step was set to 135 and the end of the growth phase to 120.

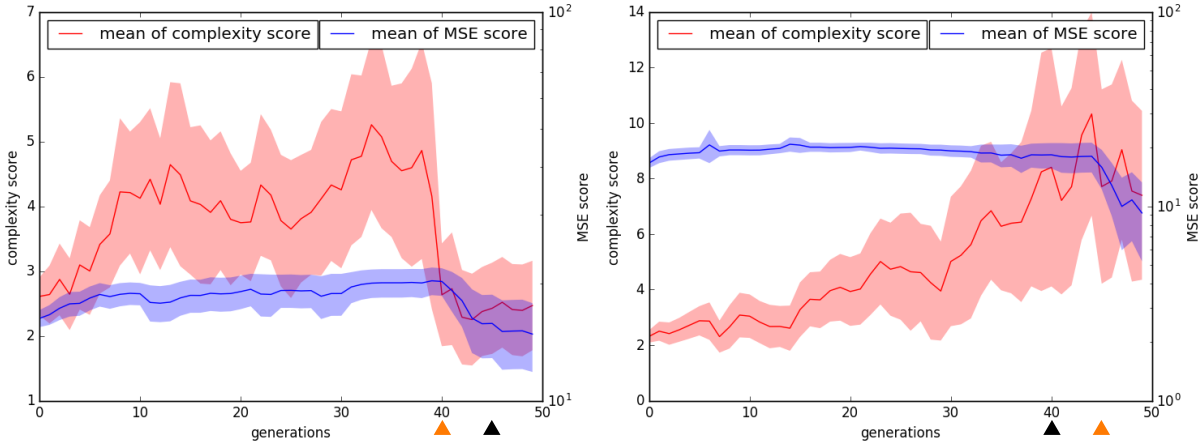


Fig. 12 Evolution of the population scores (MSE in blue and complexity in red) of two runs. The run on the left has a pruning step starting at generation 45 whereas its growth phase ends at generation 40. The run on the right has a pruning step starting at generation 40 whereas its growth phase ends at generation 45. Black triangle: start of the pruning step.

Similar observations can be made for the MSE score and the complexity score (a decrease coinciding with generation 120), confirming the previous results.Most of the 5 best networks of each run performed a band-pass behavior. Most of these networks were however more complicated, as an overall higher complexity score reflects. The topology of core GRN generated by the default run was found in almost all networks, though the parameters similarity was not evaluated.

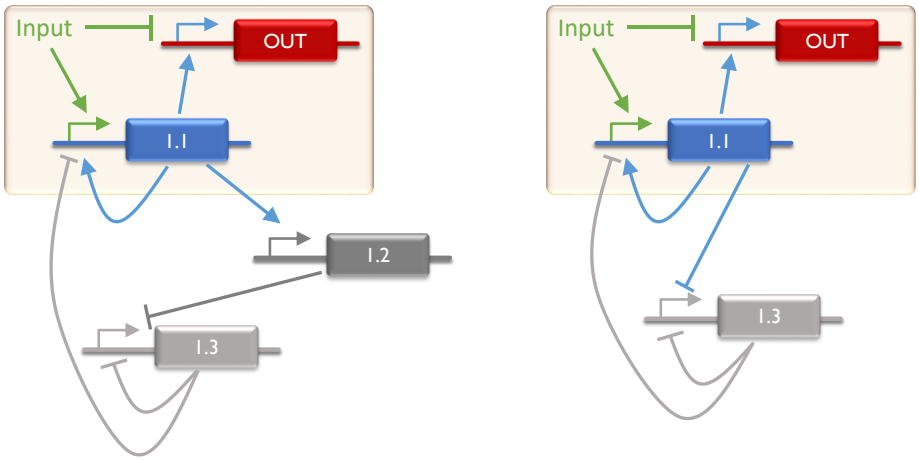


Fig. 13 Two band-pass generated by our GP algorithm with a premature start of the pruning step. Over the orange background, the fundamental core of the networks, also found in default runs.

When compared to the default run; the MSE score span was lower but the best MSE score was in the same range (around 0.02). As the run demanded a higher computing power but provided far more complicated solutions for a similar score, a high number of generation is not necessarily required.

A less demanding solution to virtually increase the number of generations would be to increase the number of mutations. As our algorithm has no crossover, the only operations that are added by an increased number of generations are the mutations and the selections. We therefore used the default

set-up and augmented the maximal number of mutation to 6. This solution gave a better tradeoff between results and demanded computing power.

Conclusion on the 4-points target

Most of the obtained GRN performed very wide band-pass. This was in accordance with a permissive target, which only included low-value output points at very low and very high values of input only. Not constraining the output allows to obtain a wide variety of band-pass, with exotic implementations. However, if a more defined response is to be obtained, the target has to possess more points.

3.3.5. Target with a higher number of evaluated points

For the following input values [1e-05, 0.01, 1, 10, 1000.0] the following output values were expected [0, 3, 20, 3, 0]. Using otherwise default parameters, except for the maximal number of mutations that was 6, the algorithm was able to find band-pass systems. None of them matched the 5 points, as reflected by a best MSE score of 5. The algorithm provided very simple networks with only a few exceeding 3 nodes.

As the algorithm found no solution toward which it could converge, we tried to increase the size of the population. With a higher number of individuals, more mutations are performed per generations so that the chances to find a good solution to direct convergence are increased. The results were however similar in all points but the computing time, which was higher.

In general, the band-passes generated were very flat. As our target function is relatively peaky, we tried to increase the weight of the central point so that its matching would favor the individual. The weight of the central point was therefore 5 times higher than the weight of the others points. This resulted in two types of behaviors only: a flat response (left diagram Fig. 14) and an activation response (right diagram Fig. 14).

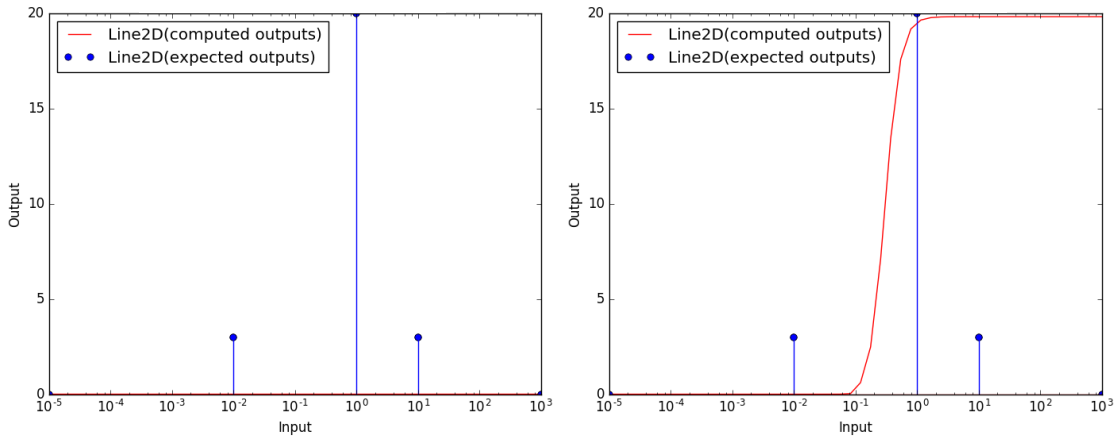


Fig. 14 Results for a 5-points target with 6 maximal mutations per networks and an increased weight for the central point. The blue dots are the expected outputs and the red curve is the simulation result of the network.

A 40 –points target was also tried. The algorithm also failed to provide satisfying results. We therefore see that for a finer description of the expected response, the fine-tuning of the algorithm is delicate and requires a deeper examination.

4. Conclusion

We evaluated our algorithm on a band-pass target. We see that with only 4 points directing the evolution of the networks, our algorithm provides a large variety of band-pass responses and possible implementations thereof. A sub-defined target has the advantage of computing solutions in a reduced time, as the evaluation is rather fast. Interestingly, we could obtain a system competing with the Basu system in terms of number of genes. A coarse analysis revealed the core features of the different networks evolved by the algorithm. A finer analysis supplemented with wet bench trials could lead to a very compact biological band-pass.

With a finer target having more points to match, we could not find the correct parametrization of the algorithm. Only few parameters were tweaked. A lead for investigation would be to give the algorithm the possibility to add complexes in the networks. The initial population could also be initialized randomly.

5. References

Ausländer, Simon, David Ausländer, Marius Müller, Markus Wieland, and Martin Fussenegger. 2012. "Programmable Single-Cell Mammalian Biocomputers." *Nature* 487 (7405). Nature Publishing Group: 123–27. doi:10.1038/nature11149.

Basu, Subhayu, Yoram Gerchman, CH Collins, FH Arnold, and R Weiss. 2005. "A Synthetic Multicellular System for Programmed Pattern Formation." *Nature* 434 (April).

Bernot, Gilles, Jean-Paul Comet, Adrien Richard, and Janine Guespin. 2004. "Application of Formal Methods to Biological Regulatory Networks: Extending Thomas' Asynchronous Logical Approach with Temporal Logic." *Journal of Theoretical Biology* 229 (3). Elsevier: 339–47.

Collet, Pierre, Evelyne Lutton, Marc Schoenauer, and Jean Louchet. 2000. "Take It EASEA." In *Parallel Problem Solving from Nature PPSN VI*, 1917:891–901. Springer, Berlin, Heidelberg. doi:10.1007/3-540-45356-3_87.

Cramer, Nichael Lynn. 1985. "A Representation for the Adaptive Generation of Simple Sequential Programs." In *Proceedings of the 1st International Conference on Genetic Algorithms*, edited by John J. Grefenstette, 183–87. Lawrence Erlbaum Associates.

Deb, K, and R W Agrawal. 1995. "Simulated Binary Crossover for Continuous Search Space." *Complex Systems* 9: 115–48.

Deb, Kalyanmoy, and David E. Goldberg. 1989. "An Investigation of Niche and Species Formation in Genetic Function Optimization." In *Proceedings of the 3rd International Conference on Genetic Algorithms*, 42–50. M. Kaufmann Publishers.

Doboli, Alex, and Ranga Vemuri. 2003. "Exploration-Based High-Level Synthesis of Linear Analog Systems Operating at Low/medium Frequencies." *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 22 (11). IEEE: 1556–68.

Dorigo, Marco., and Thomas. Stützle. 2004. *Ant Colony Optimization*. MIT Press. https://mitpress.mit.edu/books/ant-colony-optimization.

Ehrlich, P.R., and P.H. Raven. 1964. "Butterflies and Plants: A Study in Coevolution." *Evolution* 18 (4): 586–608. doi:10.2307/2406212.

Eshelman, Larry J, and J David Schaffer. 1992. "Real-Coded Genetic Algorithms and Interval-Schemata."

In *FOGA*, edited by L Darrell Whitley, 187–202. Morgan Kaufmann.

Fogel, David B., and David B. 1998. *Evolutionary Computation : The Fossil Record*. IEEE Press.

François, Paul, and Vincent Hakim. 2004. "Design of Genetic Networks with Specified Functions by Evolution in Silico." *Proceedings of the National Academy of Sciences of the United States of America* 101 (2): 580–85. doi:10.1073/pnas.0304532101.

Gendrault, Yves, Morgan Madec, Martin Lemaire, Christophe Lallement, and Jacques Haiech. 2014. "Automated Design of Artificial Biological Functions Based on Fuzzy Logic." In *Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE*, 85–88.

Goh, C S, A A Bogan, M Joachimiak, D Walther, and F E Cohen. 2000. "Co-Evolution of Proteins with Their Interaction Partners." *Journal of Molecular Biology* 299 (2): 283–93. doi:10.1006/jmbi.2000.3732.

Hansen, Nikolaus, and Andreas Ostermeier. 2001. "Completely Derandomized Self-Adaptation in Evolution Strategies." *Evol. Comput.* 9 (2). Cambridge, MA, USA: MIT Press: 159–95. doi:10.1162/106365601750190398.

Holland, John H. 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. *The Quarterly Review of Biology*. Vol. 1. MIT Press. doi:10.1086/418447.

Hornby, Gregory S, Al Globus, Derek S Linden, and Jason D Lohn. 2018. "Automated Antenna Design with Evolutionary Algorithms." Accessed February 15.

Kirkpatrick, S, C D Gelatt, and M P Vecchi. 1983. "Optimization by Simulated Annealing." *Science, New Series* 220 (4598): 671–80.

Koza, John R. 2003. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Norwell, MA, USA: Kluwer Academic Publishers.

Koza, John R. 1992. *Genetic Programming : On the Programming of Computers by Means of Natural Selection*. MIT Press.

Koza, John R. 1994. *Genetic Programming II : Automatic Discovery of Reusable Programs*. MIT Press.

Koza, John R, Forrest H Bennet III, and David Andre. 1999. "Method and Apparatus for Automated Design of Complex Structures Using Genetic Programming." https://patents.google.com/patent/US5867397A/en.

Koza, John R, Forrest H Bennett, David Andre, Martin A Keane, and Frank Dunlap. 1997. "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming." *Evolutionary Computation, IEEE Transactions on* 1 (2). IEEE: 109–28.

Koza, and John R. 1989. "Hierarchical Genetic Algorithms Operating on Populations of Computer Programs." *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*. Morgan Kaufmann Publishers Inc.

Lohn, Jason D, and Silvano P Colombano. 1998. "Automated Analog Circuit Synthesis Using a Linear Representation." In *Evolvable Systems: From Biology to Hardware*, 125–33. Springer.

Maitre, Ogier, Laurent A Baumes, Nicolas Lachiche, Avelino Corma, and Pierre Collet. 2009. "Coarse Grain Parallelization of Evolutionary Algorithms on GPGPU Cards with EASEA." In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, 1403–10.

Metropolis, Nicholas, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. "Equation of State Calculations by Fast Computing Machines." *J. Chem. Phys. J.*

Chem. Phys. Journal Homepage 21 (6). doi:10.1063/1.1699114.

Metropolis, Nicholas, and S Ulam. 1949. "The Monte Carlo Method." *Journal of the American Statistical Association* 44 (247): 335–41.

P. Collet F. Krüger, O Maitre. 2013. "Massively Parallel Evolutionary Computation on GPGPUs." In , edited by S Tsutsui and P Collet, 15–34. Springer.

Patil, Kiran, Isabel Rocha, Jochen Förster, and Jens Nielsen. 2005. "Evolutionary Programming as a Platform for in Silico Metabolic Engineering." *BMC Bioinformatics* 6 (1): 308. doi:10.1186/1471-2105-6-308.

Rechenberg, I. 1974. "Adaptive Mechanismen in Der Biologischen Evolution Und Ihr Einfluss Auf Die Evolutionsgeschwindigkeit: Arbeitsbericht."

Rechenberg, I. 1965. "Cybernetic Solution Path of an Experimental Problem." *Journal of Theoretical Biology* 215: 441–48.

"Registry of Standard Biological Parts." 2015. Accessed November 6. http://parts.igem.org/Main_Page.

Thomas, René, Denis Thieffry, and Marcelle Kaufman. 1995. "Dynamical Behaviour of Biological Regulatory networks—I. Biological Role of Feedback Loops and Practical Use of the Concept of the Loop-Characteristic State." *Bulletin of Mathematical Biology* 57 (2). Springer: 247–76.

Xie, Zhen, Liliana Wroblewska, Laura Prochazka, Ron Weiss, and Yaakov Benenson. 2011. "Multi-Input RNAi-Based Logic Circuit for Identification of Specific Cancer Cells." *Science (New York, N.Y.)* 333 (6047). American Association for the Advancement of Science: 1307–11. doi:10.1126/science.1205527.

Summary of Part Two

In this part dedicated to design automation of biological systems, we have developed and validated several tools. First, we have developed GeNeDA, a tool that takes as input a Boolean description of a GRN in a specific format (BLIF, Verilog, truth table, etc) and synthesizes, from this description, a combinatorial GRN. The tool involves a digital synthesizer from electronics and has been tuned in order to take into account the specificities of biology, as for instance the set of elementary digital functions that can be achieved by genetic regulation mechanisms.

Then we turned to the synthesis of sequential systems. For this, we first applied a design method borrowed from digital electronics (namely Huffmann method) to compute the Boolean equations of the system. Then, we used GeNeDA to obtain the GRN from these Boolean equations A deep analysis highlighted the limits of this approach. Two main difficulties have been identified: i) even for simple systems, the designed GRN are in general quite complex and are at the limit of what can be integrated in a single cell and ii) they are quite sensitive to the variations of the biochemical parameters (dissociation constants, translation and transcription rate, degradation coefficient, etc). Thus, these parameters have to be well-controlled and homogenized to guarantee the proper operation and the stability of the designed systems made.

Electronics tackled the problem of stability of sequential systems by deploying synchronous solutions. In this case, the feedback loops (source of malfunctions) are no longer calculated permanently but only at the tick given by a clock signal. This principle can be applied to synthetic biology as long as we can implement a D flip-flop. We have proposed a new GRN architecture to realize this function, composed of only 3 operons (versus 7 for the existing one). The biological D flip-flop has been validated in simulation, its robustness has been challenged by a set of Monte-Carlo simulation and it has been tested in some classic examples, such as counters. Again, the difficulty of integrating more than one flip-flop in a given cell has been highlighted. In addition, the question of the generation and the propagation of the clock signal, which is also a critical point in synchronous systems, has been raised.

In a second time, we were interested in the synthesis of circuits for which a Boolean description is not possible or is not sufficient to specify properly the system to realize. To tackle this question, we implemented genetic algorithms. In particular, we demonstrated that, using these algorithms, it is possible to find a set of parameters for a GRN that matches a response specified *a priori*. This is an important step toward design automation but the network structure has to be defined by a preliminary abstracted synthesis. Moreover, this approach still does not answer the question of the link between an abstracted GRN and an actual building block that can be found in part libraries. At the most, this tool gives clues to the designers on the key parameters of the system for which a particular care has to be paid when choosing these building blocks.

For this last limitation, we expected to bring a better solution with combinatorial search algorithms. Unfortunately, this study has not been successful. Besides, we tried to implement genetic programming to meet the challenge of the design of a GRN from scratch. We developed a Python tool than can evolve both the network topology and its parameters. The tuning of this algorithm in order to obtain consistent results is very subtle. In particular, we tinkered the selection pressure of the algorithm, the freedom that GRNs have in order to grow, the way the complexity of the network is taken in to account, the way the networks are pruned at the end of the algorithm, etc. Nevertheless,

we managed to find with the algorithm relevant results on some examples picked up from the literature.

The design approach we employ could, of course, be improved. The current tool allows the evolution of a GRN based on a set of requirements of the type dose-response with the possibility to have many inputs and outputs. The employed model however lacks the simulation of mRNA. Modeling mRNA as a species would give more precise results but also enable a new type of interactions: micro-RNA (miR) inhibition. This inhibition happens at the mRNA level (see Chapter 2). Having miR in our GRNs would allow a greater diversity of repressors and therefore a reduced risk of crosstalk. Moreover, the miR pattern of a cell (which miR are expressed in this cell) can be used to identify cancerous cells (Xie et al. 2011). A new version of the tool is an undergoing work. This new version includes mRNA and miR interactions as well as the possibility to have dynamic response as a target.

The tools that we developed in this thesis allows us to consider the design of GRN of an intermediate size (10 to 20 regulations). In this case, the main question will not be how to design the biological circuit but rather how to implement it in a cell. The solution to split the logical function in different cells that communicate with each other by chemical messengers has made its way in recent years. We have to be ready to answer it. The algorithms described in this part will be handle problems with such a level of complexity. However, since the system is distributed in several cells, spatialization plays a crucial role in the model. It will be necessary to couple the algorithm to an evaluation function that supports models depending on both space and time. This is the topic of the next part of the manuscript. In addition, computation time may be a bottleneck for this approach, combining an algorithm that is itself time-consuming and evaluating multiple models at each iteration, the simulation of each model being also time-consuming. In such a situation, we try to find a good trade-off that allows the algorithm to converge fast (but not too fast, because of the risks of being trapped into a local minimum) but that provides simulation results accurate enough to be trustable.

Introduction to Part Three

The Design Automation part highlighted one observation: as soon as we want to realize GRNs that perform a non-elementary function (e.g. sequential systems, finite state machines, counters, etc) the system cannot be integrated in a single cell due to technological issues. Thus, we have to divide it into several sub-networks, each of them being implemented in a different cell. This new way of making GRN has been suggested a few year ago and experimentally validated on a small-scale circuits (i.e. a XOR gate composed of 4 cells) (Brenner, You, and Arnold 2008; Li and You 2011; Tamsir, Tabor, and Voigt 2011). The design at a larger scale of such systems requires a tool that enables the combined simulation of intracellular biological mechanisms and map of concentration of the molecule involved in cell-to-cells communications. This is the issue we are addressing in this part.

A non-exhaustive list of the requirements for such a tool are given below:

- The tool should be able to combine local and global phenomena.
- For local phenomena, it should take as input common formalism, such as SBML, under which models of biological systems already exist.
- It should be fully integrated in our design environment
- It should be configurable in such a fashion that the user can play with the trade-off between accuracy and computation time in different ways.
- It should open-source.

Several solutions already exist to handle such models. They have been studied beforehand and some of them are described in this manuscript. However, this state of the art brings to light that no tool fulfils all of our requirements. On the other hand, members of our team have recently developed a simulator dedicated to tackle a similar issue, but in a very different domain. Their purpose was the electro-thermal simulation of integrated circuits (Krencker et al. 2010). The tool we developed during this thesis is an adaptation of this simulator to a biological context.

This part of the manuscript is divided into two chapters. The first one (Chapter 6) describes the way the tool has been developed. It begins with a state of the art of existing approaches and tools that can be used to take space location into account in biological models. An emphasis is put on three existing tools: HSIM, COMSOL and Virtual Cell. In the Section 2, the theoretical background on the reaction-diffusion equation (Fick's law used to model space- and time-dependent biological systems) and its counterpart, the heat equation, is reminded. Then, our tool is described. It is mainly composed of a mesher, a circuit generator and a model for an elementary mesh, on which focus is put in the Sections 4 and 5. Two discretization schemes have been evaluated to compute this model: finite differences and finite element methods.

The second chapter of this part (Chapter 7) is devoted to the validation of the tool and its application on several use cases:

- Basu's pattern generator, already presented in Part Two, but this time in its complete version including sender and receiver cells (Basu et al. 2005)
- A XOR realized with a consortium of 3 cells, each performing a NOR function and communicating with each other via AHLs (Tamsir, Tabor, and Voigt 2011)

- A simplified prey-predator system (Balagaddé et al. 2008)
- The study of the synchronization of biological oscillators (Garcia-Ojalvo, Elowitz, and Strogatz 2004)

References

Balagaddé, Frederick K, Hao Song, Jun Ozaki, Cynthia H Collins, Matthew Barnet, Frances H Arnold, Stephen R Quake, and Lingchong You. 2008. "A Synthetic Escherichia Coli Predator-Prey Ecosystem." *Molecular Systems Biology* 4 (1): 187. doi:10.1038/msb.2008.24.

Basu, Subhayu, Yoram Gerchman, Cynthia H Collins, Frances H Arnold, and Ron Weiss. 2005. "A Synthetic Multicellular System for Programmed Pattern Formation." *Nature* 434 (7037): 1130–34. doi:10.1038/nature03461.

Brenner, Katie, Lingchong You, and Frances H Arnold. 2008. "Engineering Microbial Consortia : A New Frontier in Synthetic Biology," no. July: 483–89. doi:10.1016/j.tibtech.2008.05.004.

Garcia-Ojalvo, J., M. B. Elowitz, and S. H. Strogatz. 2004. "Modeling a Synthetic Multicellular Clock: Repressilators Coupled by Quorum Sensing." *Proceedings of the National Academy of Sciences* 101 (30): 10955–60. doi:10.1073/pnas.0307095101.

Krencker, Jean-Christophe, Jean-Baptiste Kammerer, Yannick Hervé, and Luc Hébrard. 2010. "Direct Electro-Thermal Simulation of Integrated Circuits Using Standard CAD Tools." In *Thermal Investigations of ICs and Systems (THERMINIC), 2010 16th International Workshop on*, 1. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5636296.

Li, Bochong, and Lingchong You. 2011. "Synthetic Biology: Division of Logic Labour." *Nature* 469 (7329). Nature Publishing Group: 171–72. doi:10.1038/469171a.

Tamsir, Alvin, Jeffrey J Tabor, and Christopher A Voigt. 2011. "Robust Multicellular Computing Using Genetically Encoded NOR Gates and Chemical 'wires.'" *Nature* 469 (7329). Nature Publishing Group: 212–15. doi:10.1038/nature09565.

Chapter 6
Description of the Simulator

1. State of the art on spatial simulation in biology

1.1. Various simulation approaches

Space can be taken into account in biological models and simulators by different means (Takahashi, Arjunan, and Tomita 2005). Three main approaches exist: particle-centered modeling, concentration map and compartmental modeling.

In particle-centered modeling, each instance of each chemical species is modeled by an entity with a given position in a continuous space and with a given displacement vector. At every time step, the position of the particle is updated and a new displacement vector is computed according to a random walk which may depend on physico-chemical properties. When two particles are close enough, an interaction (binding, degradation, synthesis of another particle) may occur. HSim is an example of simulator that uses such an approach (Amar, Bernot, and Norris 2004). It is described in more details in the next section. The complexity of such algorithm grows linearly with the number of instances and the number of interactions. As a consequence, particle-centered modeling can be very useful for the study of elementary mechanisms with a low number of particle but it is not adapted to predict the behavior of large-scale systems.

Cellular automata are an alternative to perform particle-centered modeling. In this case, space is separated in elementary volumes (voxels). Voxel are in a given state depending on the number and the type of species occupying it. At each time step, the state of a voxel is updated according to rules depending on its current state and those of its neighbors (Ermentrout and Edelstein-Keshet 1993) . These rules may correspond to particle displacement (*e.g.* if a voxel is in an “empty” state and the neighbor on its right is in an “occupied” state, it turns to an “occupied” state and right neighbor on its right turns to an “empty” state) and/or chemical reactions (*e.g.* if the present voxel is in an “occupied by molecules A and B” state, it turns to an “occupied by AB” state). Rules might be either deterministic, stochastic or hybrid. The main advantage of this approach is the use of Boolean conditions instead of probability distributions that could be complex to describe and to calculate. However, simulation results may be very sensitive to the way space is divided, as it has been demonstrated in (Madec et al. 2012). Moreover, the computation complexity, which depends on the number of rules to evaluate and thus the number of states of the voxel, still grows exponentially with the number of potential interactions.

An alternative to particle-centered modeling is to compute concentration maps depending on both space and time for each species involved in the system. In this case, space is divided into a lattice of connected nodes. The concentration of chemical species is computed at every node and every time step as a function of the concentration at neighbor nodes. Many variants of this approach exist. The lattice can be regular or adaptive (the grid layout depends on the context), fluxes of particles between nodes can be deterministic or stochastic and, in both cases, computed by different ways.

For deterministic simulations, the system is described through partial differential equations (PDE) (Schaff et al. 1997). Most of the time, no analytical solution of these PDEs exists. Therefore, numerical resolutions methods such as finite differences, finite elements or finite volume are required (Johnson 2009; Evans, Blackledge, and Yardley 2000). Such methods are already implemented in several existing tools such as COMSOL™, which is one of the most used commercial generic PDE solver with an interactive graphical user interface (see next section) or FreeFEM++ (Hecht 2012), an open-source PDE solver written in C++ and a language using C++ idioms to describe systems. These methods can lead to

very accurate results when using very refined lattices and sophisticated methods but the price to pay is often a very high computation time.

For stochastic simulation, the computation process is the same as for stochastic simulations of dimensionless problems, except that it is executed in each voxel (Lecca et al. 2010). Diffusion between one voxel and each neighbor is put at the same level as reaction. The probabilities of the mechanisms are computed from the reaction rate on the one hand and from the diffusion law on the other. Gillespie's algorithms (Gillespie 1977) are then applied in each voxel in order to compute the transient evolution of the system.

Stochastic simulations are very accurate, especially for problems in which the concentrations are low (i.e., the computed concentrations correspond to quantities of molecule in the order of the unit). However, they are in general very computationally intensive, in comparison to deterministic methods. To take advantage of both methods, hybrid simulation algorithms have been developed(Spill et al. 2015; Harrison and Yates 2016). The main idea is to divide for each species space into two regions: one in which the concentrations are high and where a deterministic simulation is performed and one in which the concentrations are low and where a stochastic simulation is performed. The stochastic and deterministic regions overlap at an interface in which both simulation are coupled. This interface moves during the simulation according to conditions on concentrations.

The third alternative, which is the less expensive from a computation time perspective, is to compute the concentration of species only at several points of interest instead of over the whole space. In this case, space is divided into compartments and diffusion is modeled as fluxes between compartments. Compartments are therefore not located in space. The distance and properties of the inter-compartments medium are integrated as fixed parameters in the diffusion fluxes that connect compartments to each other. By this means, the space- and time-dependent problem is reduced to a time- only dependent problem. In most of the cases, even for simple diffusion problems, there are no analytic equation to compute these fluxes over complex geometrical shapes. Thus, approximation has to be made. Most of the biological simulators, as for instance COPASI (Hoops et al. 2006) or Virtual Cell (Loew and Schaff 2001) use this approach.

1.2. Existing tools

After reviewing different approaches for the spatio-temporal simulation of biological systems, focus is put on 3 existing tools: HSIM, COMSOL and Virtual Cell.

1.2.1. HSIM

HSIM is a biochemical simulator based on a stochastic automaton. HSIM was first released in 2004 (Amar, Bernot, and Norris 2004). As for every particle-centered simulator, the position and the type of each involved molecule are stored in a table. To describe a system, users have to write a configuration file according to a proprietary formalism. This file contains the following information: i) the geometry of the system (total length and diameter), ii) the name, the type, the size and the speed of the involved molecules, iii) a list of interaction with, for each, a probability of occurrence when the involved molecules collide and iv) the initial amount of each molecule. At each time step and for each molecule, the simulation loop depicted in Fig. 1 is computed.

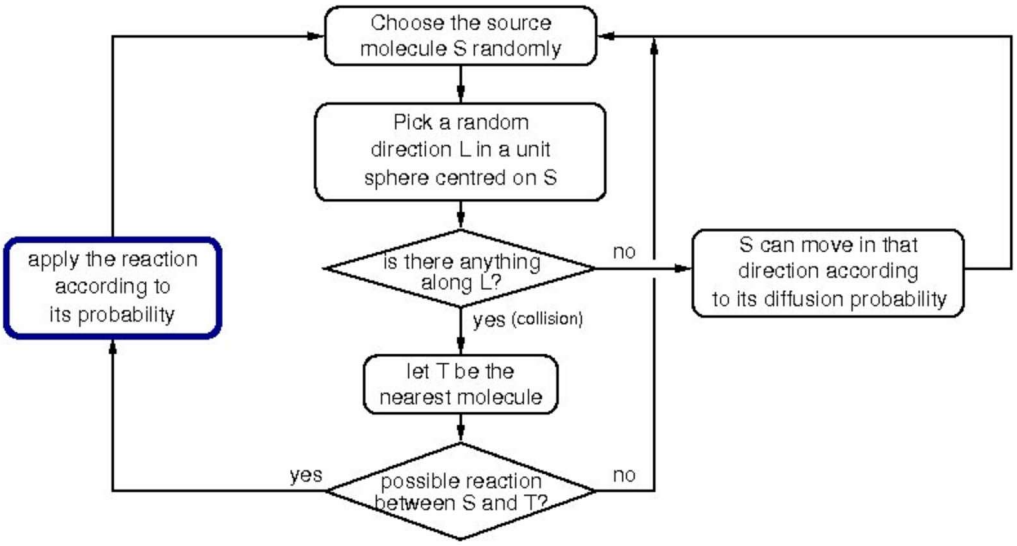


Fig. 1 Simulation loop of HSIM simulator. Extracted from (Amar and Paulevé 2015)

To develop a model with HSIM, the user first defines the geometry, i.e. a list of compartments. Then, he has to set the initial conditions, i.e. the number of each molecule and their initial position. Finally, he writes a list of rules that corresponds to every involved biochemical mechanisms according to a proprietary formalism. Parameters of the model are the following: the size of the compartments, the probability for each species to move during one time step and the probability for a reaction to occur when both reactants collide during a displacement. An example of the simulation of an enzymatic reaction is given on HSIM website. In our case, we are interested in the diffusion of molecules in a closed space and from a source point. To implement such model on HSIM, we start from the enzymatic reaction example. We set the number of enzyme to 1 and fix its position. Each time a substrate molecule collides with the enzyme, a product is synthesized. Simulation results are given on Fig. 2.

Recent versions of HSIM include more sophisticated features, as for instance the ability to couple particle-centered simulation with stochastic non-spatial simulations (Amar and Paulevé 2015). By this way, the simulation time decreases drastically for systems composed of species present in small amount and for which spatial localization is relevant as well as for systems composed of species present in larger amount and for which spatial localization is not relevant. In practice, there are two types of species: the one that are treated individually and the one that are treated globally. The previous algorithm (Fig. 1) has been modified as follows to take into account of all the possible interactions. Interactions between individually treated molecules are dealt with as in the original version of HSIM. Interaction between globally treated molecules are treated with a stochastic simulation algorithm such as Gillespie's algorithm (Gillespie 1977). Interaction between individually treated molecules and globally treated molecules are computed as follows. During the diffusion phase, the average numbers of collisions between both kinds of molecules is computed and the corresponding reaction rules are applied.

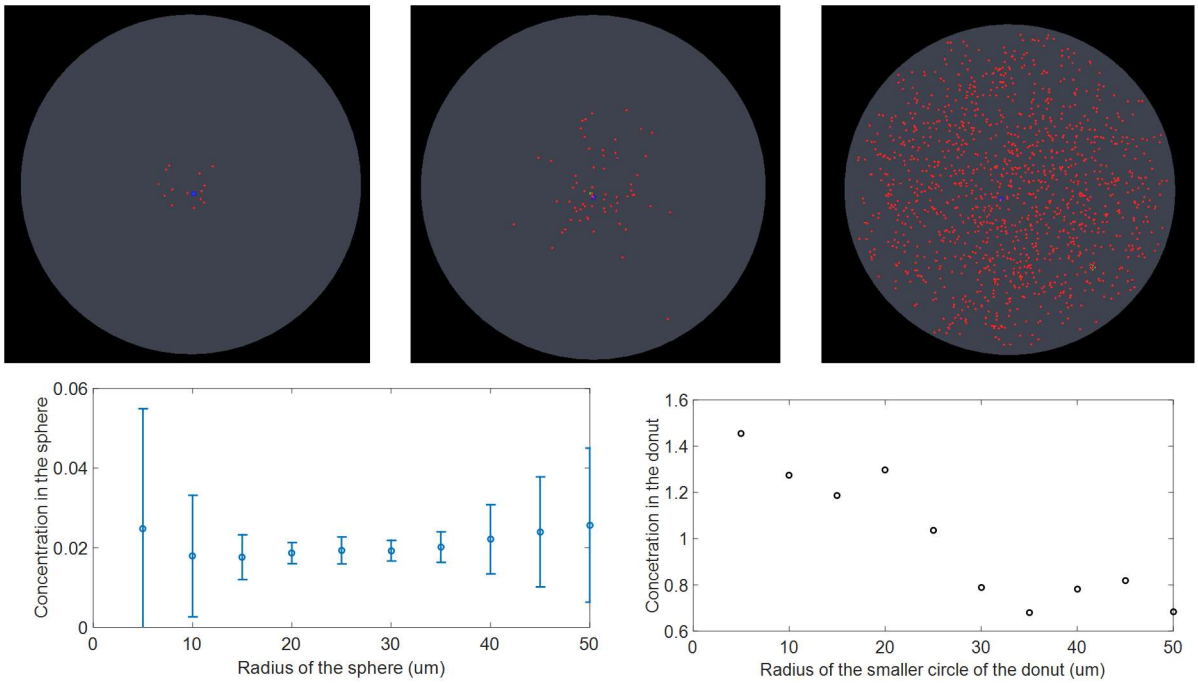


Fig. 2 Simulation results obtained with HSIM. Top three images correspond to the simulation of the diffusion of the product (red dots) synthesized at the enzyme (blue dot at the center). Bottom left diagram gives the distribution of the concentration of product into concentric spheres as a function of the radius of the sphere. Bottom right figure is equivalent except that the concentration is estimated in a 5µm donut instead of in a complete sphere.

1.2.2. COMSOL

COMSOL Multiphysics is a commercial finite element analysis, solver and multiphysics simulation software. It is composed of a tool for the specification of the geometry of the problem, several modules that correspond to the different domains of physics supported by the tool, an adaptive and physics-driven mesher, a computation core that handles and solves PDEs generated by the modules and tools for the analysis and the visualization of simulation results. Moreover, modules can be coupled together in order to perform multiphysics simulations. Different tools are accessible to the user through a Graphical User Interface or can be called in a MATLAB script.

Here is a step-by-step description of how to describe a model with COMSOL. The use case is the free diffusion of a molecule in a 100x100 square which is emitted by a cell at the center.

a) Selection of the module and the interface we would like to use. In our case, we use the "Chemical Species Transport" interface of the "Chemical Reaction Engineering" module. At this first step we also define the number and the name of the molecule under study. In our case, 1.
b) Definition of the geometry. In our case, a square of 100 mm with a point P at the center.
c) Definition of the boundary conditions. In our case, we set the concentration on the borders of the square to 0 and a flux is applied on the central point P.
d) Tuning of the physics. The master equation of the module we would like to use is the following:

$$\frac{dc}{dt} + \mathbf{u} \cdot \nabla c = \nabla \cdot (\mathrm{D} \cdot \nabla c) + R$$

where c is the space- and time-dependent concentration of the species (in mol/m^3), $\mathbf{u}$ is the velocity vector (in m/s), D is the fluid is diffusion coefficient (in m^2/s) and R is the reaction rate expression for the species (in mol/m^3/s). In our case, the diffusive medium is not in movement,

so $\mathbf{u} = 0$. Moreover, the only reaction that occurs in the diffusion medium is degradation of the molecule. Thus, R is set to $-d \cdot C$ (with d the degradation rate) over the whole surface of the square. The last step is to fix the values of D and d.

e) Generation of the mesh. The mesh is generated through the GUI with different options. The mesh generated for our study is given in Fig. 3. It has been obtained with the following options: "Physics-controlled meshing" and "Finer".

f) Start of the computation.

g) Analysis of the results. In our case, the output is a map of the concentration over the square (Fig. 3).

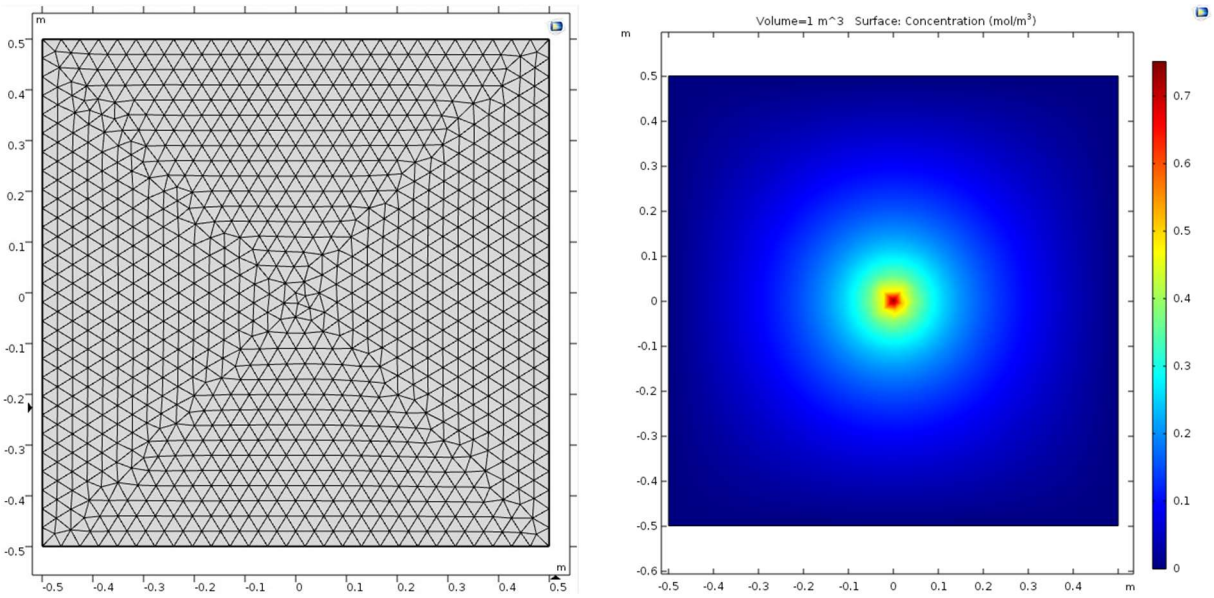


Fig. 3 Simulation results obtained with COMSOL. On the left, the lattice obtained with the COMSOL mesher in standard configuration. On the right, simulation results for the diffusion of molecules from a source at the center in a closed space.

1.2.3. Virtual Cell

Virtual Cell (VCell) is an open-source software platform dedicated to modeling and simulation of biochemical systems. Basically, a model is composed of different parts, described in an XML file: the compartments in which the reactions occur (cell nucleus, cytosol, membrane, etc), a list of species in each compartment, a list of fluxes between compartments and a list of chemical reactions with their equation rate and the compartment in which they occur. This information is used to set a list of ordinary differential equation that describes the system. These equations can be solved either with deterministic or stochastic methods.

Features introduced in recent versions of VCell enable spatial simulation. Geometries can be specified by analytic geometry equations, derived from combination of simple shapes or derived from imported images, such as 3D confocal microscope stacks.

Another specificity of VCell is that the computation is not performed on your own computer but deported on a server. This is both an advantage because the simulation does not consume local resources but also a drawback because it limits the coupling of the tool with others, or prevent the launch of simulation batches.

The simulation of a simple diffusion process on Virtual Cell can be carried out as follows.

a) Firstly, the user defines the system without taking localization into account. In particular, involved species, compartments, reactions inside compartments and fluxes between compartments have to be provided in the same way as if it was a non-spatial system. In our case, we define two compartments and a membrane between the two compartments. At the membrane, molecules *spec2* are produced with a constant rate. Inside the diffusion compartment, *spec2* is degraded with a standard linear model (Fig. 4A).

b) Secondly, the user has to describe the geometry of the system. In our case, we define a 100 mm x 100 mm square with a small 1 mm radius circle at the center (Fig. 4B).

c) Thirdly, the user has to map the non-spatial compartments and membranes defined in a) with actual compartments and membranes delimited by the geometry defined in b). In our case, we map the center compartment with the circle, the diffusion department with the square (excluding the circle) and the membrane with the perimeter of the circle (Fig. 4C).

d) Fourthly, the user specifies the type of simulation he would like to perform (spatial vs non-spatial, stochastic vs deterministic). The model is sent to the VCell server that performs the simulation. Results are available through a dedicated graphical user interface (Fig. 4D).

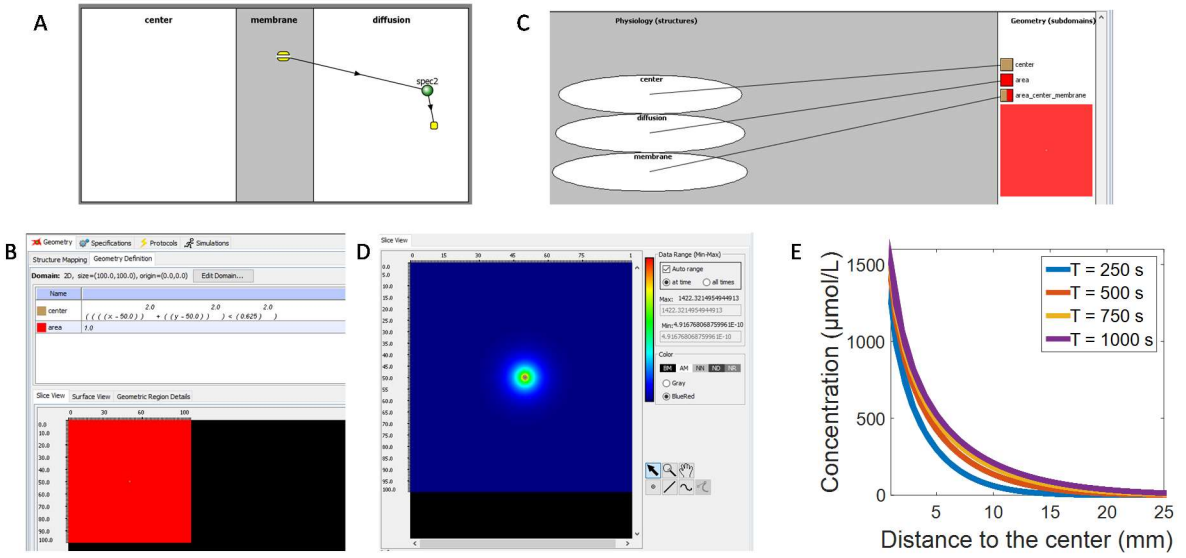


Fig. 4 Simulation results obtained with Virtual Cell. On the top left (A), the system is defined through the dedicated input graphical user interface. In our case, it is composed of 2 compartments and a membrane. A flux of the species under analysis is generated through the membrane and diffuse in the diffusion compartment, where it can be degraded. On the bottom left (B), the geometry of the system is defined. In our case, it is composed of 0.625 µm radius circle at the center and a 100x100 mm square. On the top right (C), the compartments defined in the system are mapped to the zones defined by the geometry. On the bottom middle (D), simulation results provided by Virtual Cell. On the bottom right (E), illustration of the results that can be obtained from the extracted simulated data: evolution of the concentration of the diffusing protein as a function of the time and the distance to the center.

1.3. Outcome on the state of the art

To meet the needs expressed in the introduction of part III, the ideal tool has to integrate an algorithm for the resolution of PDEs whose complexity can be controlled. The tool is intended to be used in design processes which would require a large number of simulations. Thus, simulation time has to be reduced, even if it leads to a loss of accuracy. Quantifying the tolerated accuracy loss is a very hazardous task but we are confident that targeting a very high precision equivalent to the one that would be provided by sophisticated meshes and solvers used by COMSOL is not necessary. We have to keep in mind that biological models (both for the diffusion process but also for involved biochemical reactions) also have

their limitations. Thus, it would be a waste of time to perform a simulation with a higher accuracy than the models themselves.

Moreover, we also have to keep in mind that our goal is to adapt design tools from electronics to biology. Thus, it will be necessary to build interfaces between the simulator and the other tools of the design flow. COMSOL would probably be the most suitable tool for that purpose. Unfortunately, it is a commercial tool. With HSIM, building interfaces seems to be difficult because the data structure resulting from HSIM simulation (particle positions) is very different from the one handled by electronics design tools of microelectronics (physical quantities such as voltages or current or concentrations in biological context). Finally, since the computation core of Virtual Cell is not local but is deported to a VCell server, interfacing VCell with tools running on a local machine also seems complex, or at least inefficient due to the large amount of data to transfer.

In the light of this state-of-the-art, it appears that existing solutions do not fully meet our needs and that the redevelopment of a PDE solver compatible with SPICE's electronic circuit simulator would be a better alternative. The use of SPICE ensures naturally the compatibility with a large number of tools used in electronics circuit design (e.g. circuit optimizers, performance analyzer…). Moreover, this choice is also motivated by the existence of another tool developed by our team and dedicated to the electro-thermal simulation of integrated circuits. This tool includes a PDE solver that can be easily adapted to our problem. It is described in the next chapter.

2. Theoretical background

In this section, we will first describe the analogy between electronics, thermics and biology on which our tool relies. More precisely, our tool is based on an electro-thermal simulator developed a few years ago in our team and which aim is to analyze the impact of temperature gradient inside an integrated circuit (Krencker et al. 2010; Garci, Kammerer, and Hebrard 2014). This tool is described in the second subsection. Finally, the finite difference discretization scheme, which is used in the first version of the model, is described.

2.1. Space and time modeling in biology

The generic model for a biological system that depends both on time and space is a set of PDEs (one per species) which can be written as follows:

Equation 1

$$\frac{\partial X_k}{\partial t} = f_k(X_k, t, x, y, z) + \sum_{j=1}^{R} v_{j,k}(\mathbf{X}, t, x, y, z) - d_k \cdot X_k(t, x, y, z) \qquad k = \{1 \dots N\}$$

where N is the number of involved species, $\mathbf{X}(t, x, y, z)$ is a vector with N elements $X_k(t, x, y, z)$, each being the concentration of the k-th species at time t and position (x, y, z), R the number of biochemical reactions which influence the local concentrations, $v_{j,k}(\mathbf{X}, t, x, y, z)$ is the rate at which the j-th reaction produces or consumes the k-th species, d_k is the degradation constant associated with the k-th species and $f_k(X_k, t, x, y, z)$ is a term that models the diffusion of the k-th species into space. Equation 1 is often called reaction-diffusion equation. Generally, $f_k(X_k, t, x, y, z)$ corresponds to Fick's law (Fick 1855):

Equation 2

$$f_k(X_k, t, x, y, z) = D_k(t, x, y, z) \cdot \nabla^2 X_k = D_k(t, x, y, z) \cdot \left(\frac{\partial^2 X_k}{\partial x^2} + \frac{\partial^2 X_k}{\partial y^2} + \frac{\partial^2 X_k}{\partial y^2} \right)$$

where $D_k(t, x, y, z)$ is the diffusion constant of the k-th species and ∇^2 is the Laplace operator. Sometimes, $f_k(X_k, t, x, y, z)$ can be more sophisticated (*e.g.* cell membrane, time- or space-dependence of the diffusion constant, obstacles, etc).

For the $v_{j,k}$ functions, standard models of biochemistry such as Hill's equations (Konkoli 2011), Michaelis-Menten's models (Michaelis and Maud Menten 1913) or protein-binding polynomial (Haiech, Gendrault, and Kilhoffer 2014) are implemented. Such models may depend on the concentration of all other involved species, the time and the position.

2.2. Analogy between biology, thermic and electronic

On the one hand, the analogy between electronics and biology at the device level has already been described in the chapter 2. With this analogy, it is possible to model $v_{j,k}(\mathbf{X}, t, x, y, z)$, the degradation term and the derivative term of Equation 1 with an equivalent electronic circuits.

On the other hand, an analogy can be drawn between biology and thermal physics when considering the diffusion phenomenon. The model of the diffusion of heat in a material is given by heat the equation derived from Fourier's law (Blundell and Blundell 2010).

Equation 3

$$C(t, x, y, z) \cdot \frac{\partial T}{\partial t} = \lambda_0(t, x, y, z) \cdot \nabla^2 T + Q(t, x, y, z) - h(t, x, y, z) \cdot T(t, x, y, z)$$

where $T(t, x, y, z)$ is the temperature, $C(t, x, y, z)$ is the local heat capacity, $\lambda_0(t, x, y, z)$ is the thermal conductivity, $Q(t, x, y, z)$ is the distribution of heat sources in space and time, and $h(t, x, y, z)$ is the convection constant. Similarities between heat equation (Equation 3) and the reaction-diffusion equation (Equation 1) are obvious.

Finally, the analogy between electronics and thermal physics is also well-known (Blundell and Blundell 2010). Electro-thermal simulations of integrated circuits are mostly based on this analogy.

2.3. Electro-thermal simulation of integrated circuits

The electro-thermal simulation of an integrated circuit is needed especially for high density integrated circuits, for mixed circuits with integrated power devices or for the next generation of 3D integrated circuits. Such a simulation requires two modeling layers, one dedicated to the computation of the thermal map and one dedicated the computation of the voltages and currents in the electronic circuit. Direct coupling on the one hand and relaxation on the other can be used to address this problem (Digele, Lindenkreuz, and Kasper 1997)

In the relaxation approach, a first electrical simulation is carried out with all the devices at room temperature. Then the power dissipated by each device is estimated and a 2D/3D thermal map of the chip is computed with an external tool. The mean temperature of each device is then computed from this map and fed back to the electrical simulator. This simulation loop is performed iteratively until convergence is obtained. This approach is straightforward but requires two separated tools. Moreover,

very fast changes cannot be considered as the simulation is therefore becoming highly time consuming.

The last generation of electro-thermal simulators use a direct coupling, i.e. both layers are computed together with the same solver. The principle is described in (Krencker et al. 2010; Garci, Kammerer, and Hebrard 2014). On the thermal layer, the heat equation is discretized and modeled with electronic equivalent circuits. Such circuits are composed of thermal capacitors on each node of the lattice to model heat accumulation by the chip, thermal resistors between nodes to model the heat transfer, thermal resistor between nodes and the thermal ground (ambient temperature) to model heat convection, and localized heat sources. The electronic layer is composed of temperature-dependent models of devices instantiated in the circuit. Each model is connected to the closest thermal node, uses the node temperature inside models to set temperature-dependent parameters and computes the localized heat source associated to the device.

Our tool, described in the next subsection, is based on an electro-thermal simulator developed in our lab and takes advantage of the aforementioned analogy between thermal and biological problems.

3. Overview of our simulator

The tool we developed is based on the SPICE environment. The ability of the language associated to SPICE to handle both biological systems (Madec, Lallement, and Haiech 2017) and thermal convection-diffusion problems (Garci, Kammerer, and Hebrard 2014) has already be proven. In this way, we take advantage of almost 50 years of experience and curation. Two SPICE distributions are used. Spectre MMSIM, a commercial simulator integrated in the Cadence Integrated Circuit Design Suite (https://www.cadence.com) that offers different types of analysis (operating point, transient, DC analysis, AC analysis, noise analysis, parameter sweep …). Moreover, it is parallelized and can thus simulate large-scale systems (with thousands of equation) in a very low computation time compared to other software. Beside, an open-source SPICE simulator, namely NgSpice (Nenzi and Vogt 2011) has also been tested and integrated in the following tool in order to keep the complete workflow free. The tool is composed of a suite of five main modules written in different languages (see Fig. 5): a mesher written in C++, a SPICE netlist generator written in Python, a generic model of an elementary mesh described in Verilog-A or directly in SPICE, a SPICE simulator and a Python script to read simulation output files and plot results. The complete workflow is detailed in the following subsections. Our tool supports 3D modeling but for clarity of explanation, the following will use 2D models to explain how our approach works.

3.1. Mesher

The role of the mesher is to discretize space into elements according to the problem topology and user-defined parameters. To reduce the computation time and the complexity of the model, we opted for a lattice composed of square meshes with variable degrees of refinement. Firstly, the user has to provide an input file (see the grey box on Fig. 5) that contains the list of the molecules that diffuse, general parameters concerning the lattice (its total size, the maximal size of an element) as well as information concerning the biological entities (cells) influencing the concentration. For each entity the description file must contain its position, its flux/reaction altering the concentration of molecule and its influence zone. Influence zones are ovoids (ellipse in 2D) characterized by their semi-axis and a

degree of refinement. Strong concentration gradients may be observed in these regions which thus must be finely meshed to obtain an accurate simulation.

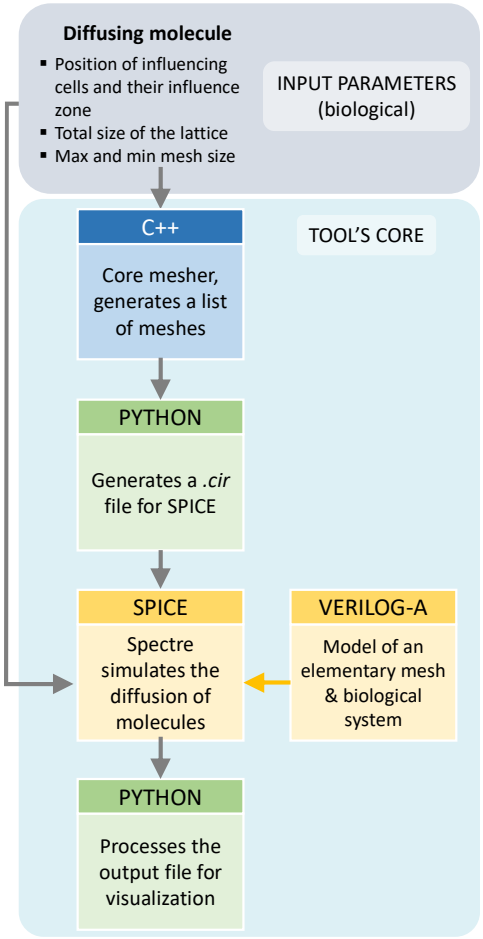


Fig. 5 Core modules of the designed tool. The label indicates the language in which the module was written.

The input file is read by the core module, namely the mesher, a C++ program that creates a list of elementary mesh and their associated nodes, as well as a list of all node coordinates. In a two-dimensional space, a first lattice made of rectangles of the maximal size defined by the user is generated. In a second step, each rectangle overlapping an influence zone (red circle in Fig. 6) is divided into 4 identical sub-rectangles if its size is larger than the maximal size allowed inside this zone. This operation is repeated until all the rectangles overlapping an influence zone are smaller than the maximal size. Finally, transitions between rough and refined lattice are smoothed: if the difference between the refinement levels of two adjacent rectangles is larger than 1, i.e. when the ratio of their sides is larger than 2, the largest rectangle is divided in 4 and this last operation is also repeated until each couple of adjacent rectangles have a difference of refinement level lower or equal to 1.

The mesher provides two output files. The netlist file is a CSV file composed of one line per mesh in the lattice and 27 columns per line. The first column is the net label. Columns 2 to 9 correspond to the node number of each corner. Columns 10 to 15 correspond to the node number of the center of each face. Columns 16 to 27 correspond to the number of the middle of each edge. The value -1 is affected

if the node number does not exist (i.e. the node of the given mesh is not connected to any corner node of any other mesh). The nodelist file is a CSV file composed of one line per node in the lattice and 4 columns. The first column is the node number. The three following are the node coordinates in x, y and z.

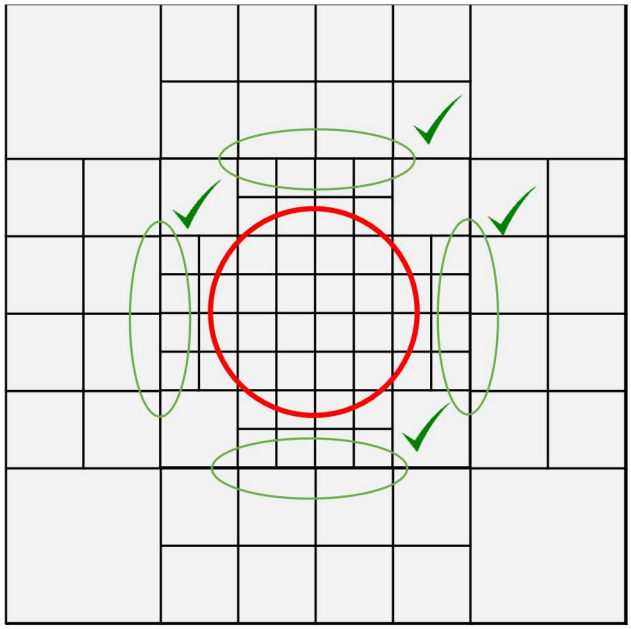


Fig. 6 Example of a lattice with 4 initial divisions and with a central refinement zone of n=2 (red circle). The algorithm checks whether the n value of each mesh in contact with the zone is superior or equal to expected value (here 2). If not, the relevant meshes are sub-divided. In order to keep a progressive refinement, some meshes have to be divided even if not in contact with the refining zone: green shapes illustrate this case.

3.2. Overview of the model of the elementary mesh

The model of the elementary mesh is composed of 9 terminals representing the concentration of molecules at each corner (C0, C1, C2 and C3) and at the middle of each edge (C01, C12, C23 and C30) as well as a reference node. It is also composed of 10 parameters described in Table 1. Corner nodes are always connected whereas the middles of each edge are connected only if the adjacent mesh is refined. Obviously, the model of the elementary mesh depends on the number and the position of the connected middles, which is described by parameters X01, X12, X23 and X30.

Two equivalent versions of the model have been developed, one in Verilog-A (more readable but not compatible with NGSPICE) and one directly in SPICE language. The Verilog-A model is given in Appendix III. More details on the content of the model (electronic equivalent circuit) are given in the next section.

1.3. Netlist generator

The SPICE simulator requires a description of the system as a netlist, i.e. a list of instantiated electrical devices (resistors, capacitor, current source, sub-circuit such as the model of the elementary mesh, …). This file contains the following elements: i) the definition of the global parameters of the model; ii) the instantiations of the elementary mesh models; iii) a map of initial concentrations; iv) boundary conditions; v) instantiations of all the local biological mechanisms and vi) simulation directives.

Table 1 Parameters of the model of a mesh

Parameter	Definition
ID	Id of the mesh (mostly for debugging purpose)
N	Refinement degree
X01, X12, X23 and X30	Boolean that indicates if a smaller neighbor mesh is connected at the middle of the corresponding edge
R_0	Local degradation constant
K_0	Scaling parameter (default = 1)
D_0	Local diffusion constant
MeshSize ($\boldsymbol{Ms}$)	Maximal allowed size of the edge of a mesh

3.2.1. Instantiation of the elementary mesh

Elementary mesh instantiations are created directly from the netlist delivered by the mesher. The Verilog-A model of the elementary mesh is encapsulated in a SPICE sub-circuit called `Mesh_unit`. A `Mesh_unit` is instantiated for each line in the netlist file. The *k*-th mesh is labelled `Mk` and a list of the 9 connections (8 nodes of the elementary mesh and the reference node which is always 0) is provided. Unconnected nodes are grounded (they have no influence on other nodes). Finally, the parameters that are not equal to their default value are specified.

Here is an example of instantiation of the elementary mesh #15 in the netlist. Bottom-left, bottom-right, top-right and top-left corners are respectively connected to node number 53, 54, 52 and 11. In addition, there is a connection at the middle of the right edge which is connected to the node 80. Local diffusion and degradation are set to their default value and the mesh is refined twice in comparison with the initial one.

```
M15 53 54 52 11 0 80 0 0 0 Mesh_unit X12 = 1 n=2.0 ID=15
```

For a system involving multiple diffusing species, a layer of elementary mesh is generated for each species. These layers are independent from each other but might be connected through reaction models (see next subsections).

3.2.2. Boundary conditions

The model allows us to consider different boundary conditions. By default, the space represented by the mesh is considered as closed. From a modeling point of view, it corresponds to open circuits after each boundary nodes. Space can also be considered as sufficiently large to have null concentrations at the border nodes: in this case, boundary nodes are grounded. Finally, it is also possible to model further diffusion outside of the mesh borders by adding a grounded resistor to all of the border nodes.

3.2.3. External models and sources

Reaction terms of Equation 1 are added by connecting associated Verilog-A or SPICE model to the electrical network generated. The path to the models and the position of the reactions are given in the input parameter file. The model of each cell must be described in Verilog-A or SPICE beforehand. SBML models can also be imported with the help of the BB-SPICE SBLML-to-SPICE translator. For each

reaction model, the netlist generator connects one instance of the model of a cell to every node in contact with a cell according to the input parameter file.

4. Model of the elementary mesh: finite difference approach

The external view of the model is described in Section 3.2 . In this section we describe in more details the content of the model of an elementary mesh. In the following, for simplicity's sake, the name of a node (e.g. C_1) will implicitly correspond to its potential, which in our analogy also corresponds to the local concentration of molecules on that node.

4.1. Model core

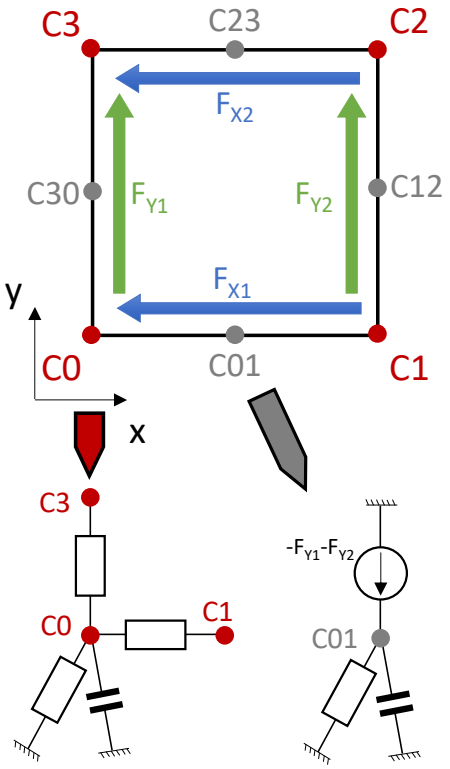


Fig. 7 Upper part: 2D elementary mesh with 8 potential connections: 4 corners (always connected, red dots) and 4 edges (not connected if the degree of refinement of the neighbor mesh is the same, grey dots). Plain arrows (blue and green) show the fluxes. Lower part: representation of the electronic devices connected to a corner node (left) or an edge node (right).

The model of the elementary mesh is given in Fig. 7. Four devices are connected to each corner node: two resistors connected at the adjacent node, a resistor R_n connected to the reference node, and a capacitor K_n also connected to the reference node. The two resistors represent the molecule flux F_{ij} between the two nodes they connect, which is proportional to the difference of concentration between these two nodes $F_{ij} = D_n \cdot (C_i - C_j)$. The resistor R_n represents the decay at a node. In the following, fluxes are renamed as follows, to identify the fluxes more easily and to avoid confusion between the nodes' numbers inside the mesh (0 to 3) and in the lattice (1 to the total number of nodes in the lattice): $F_{X1} = F_{10}$, $F_{X2} = F_{23}$, $F_{Y1} = F_{30}$ and $F_{Y2} = F_{21}$. For terminals C_{ij}, the model is slightly different with a Voltage-controlled Current Source (VCCS) instead of the two diffusion resistors. VCCS are instantiated only if the node is connected to another mesh, i.e. if the corresponding X_{ij} is True. The values of the resistance, capacitor and VCCS have been calculated in order to match the equations

obtained with the finite difference discretization scheme once all the instances of the elementary mesh are assembled.

4.2. Finite difference discretization scheme

Solving reaction-diffusion equation (or heat transfer equation) implies the resolution of PDEs. The easiest way to do that is to discretize space according to a finite difference scheme (Mazumder and Mazumder 2016). In this paragraph, we use this method to discretize Fick's equation. For simplicity's sake, a 2D system is considered.

Let $C(x, y, t)$ be the concentration of the molecules diffusing in a 2D space. In the following equations, this concentration is named C for a clearer demonstration. Fick's equation is given by:

Equation 4

$$\frac{\partial C}{\partial t} = D\Delta C - d_X C = D\left(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2}\right) - d_X C$$

with diffusion constant D and the degradation constant d_X of the diffusing molecule. We now consider a regular lattice. Let $C_{i,j}$ be the concentration at point with discrete coordinates (i, j), Δx and Δy the size of the mesh on both directions (Fig. 8). The second derivative term of C at (i, j) can be expressed as a function of the concentration at the neighbor nodes as following:

$$\frac{\partial^2 C}{\partial x^2} = \frac{\frac{\partial C}{\partial x}(x + \Delta x) - \frac{\partial C}{\partial x}(x - \Delta x)}{\Delta x} = \frac{\frac{C(x + \Delta x) - C(x)}{\Delta x} - \frac{C(x) - C(x - \Delta x)}{\Delta x}}{\Delta x}$$

Reusing the notation in discrete space established in the previous paragraph, we obtain:

$$\frac{\partial^2 C}{\partial x^2} = \frac{C_{i+1,j} - C_{i,j} - (C_{i,j} - C_{i-1,j})}{\Delta x^2}$$

We assume that our mesh is made of squares so that we have:

$$\Delta x = \Delta y = \Delta l$$

Hence:

$$\frac{\partial C}{\partial t} = \frac{D}{\Delta l^2}\left(C_{i+1,j} + C_{i-1,j} + C_{i,j+1} + C_{i,j-1} - 4C_{i,j}\right) - d_X C$$

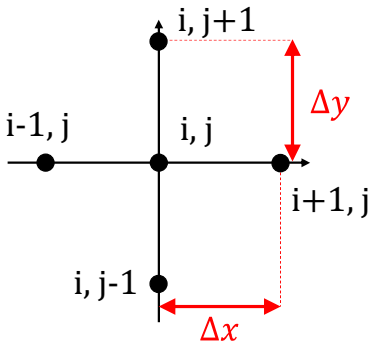


Fig. 8 Notations in a discrete space

4.3. Link with the elementary mesh model

4.3.1. Case of a regular lattice

In a regular lattice, a node (node #5 in this case) is surrounded by 4 squares of the same size. Fig. 9 defines the names of the neighbor nodes and meshes that are used in the following.

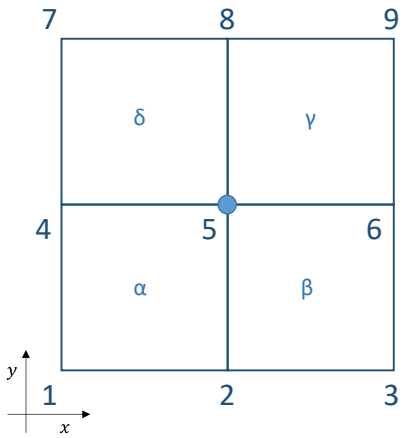


Fig. 9 A node surrounded by 4 squares of the same size

We designate as C_i the potential V_i on node i. Let write the equation governing the dynamic of the concentration at the node 5 (blue dot on Fig. 9) by applying Current Kirchhoff's Law (see Chapter 2). Four fluxes per adjacent lattice have to be taken into account, two from the diffusion phenomenon, one from the degradation resistance and one from the capacitor charge/discharge.

$$K_{eq,5} \cdot \frac{dC_5}{dt} = -F_{X2,\alpha} + F_{Y2,\alpha} + F_{Y1,\beta} + F_{X2,\beta} + F_{X1,\gamma} - F_{Y1,\gamma} - F_{X1,\delta} - F_{Y2,\delta} - \frac{C_5}{R_{eq,5}}$$

with the flux defined as in Section 4.1. The equivalent resistor connected to the ground seen at the node #5 is the parallel association of the resistor in each mesh:

Equation 5

$$R_{eq,5} = \frac{1}{\frac{1}{R_{n,\alpha}} + \frac{1}{R_{n,\beta}} + \frac{1}{R_{n,\gamma}} + \frac{1}{R_{n,\delta}}} = R_0$$

And the equivalent capacitor connected to the ground seen at the node #5 is the parallel association of the capacitor in each mesh:

Equation 6

$$K_{eq,5} = K_{n,\alpha} + K_{n,\beta} + K_{n,\gamma} + K_{n,\delta}$$

On the other hand, application of Equation 4 on node #5 leads to the following equation:

$$\frac{\partial C_5}{\partial t} = \frac{D}{\Delta l^2}(C_1 + C_3 + C_5 + C_7 - 4C_4) - dX \cdot C_5$$

Thus, by identification of terms, we obtain:

Equation 7

$$dX = \frac{1}{K_{eq,5} \cdot R_{eq,5}}$$

and:

Equation 8

$$D_n = K_{eq,5} \cdot \frac{D}{2 \cdot {Ms_n}^2}$$

with $Ms_n = \Delta l$, the size of the meshes.

Because the problem is symmetric, R_n and K_n should be equal in the four meshes. Finally, from the previous equations (Equation 5 to Equation 8) we extract the formula of R_n, K_n and Ms_n as a function of the maximal mesh size Ms, the degree of refinement n, the local decay rate R_0 and the local diffusion constant D_0 as follows:

$$R_n = \frac{4R_0}{\left(\frac{1}{2}\right)^n}$$

$$K_n = \frac{\left(\frac{1}{2}\right)^n}{4}$$

$$Ms_n = \frac{Ms}{2^n}$$

4.3.2. Case of an irregular lattice

To check the consistency of our definition, we repeat the same process for different configurations involving different refinement levels n. We also consider squares only. In this case, the discretization process involves more variables, as nodes in the middle of a segment also intervene. We therefore note $C_{i+\frac{1}{2},j}$ and $C_{i,j+\frac{1}{2}}$ the concentration of a node at a distance $\frac{\Delta l}{2}$ of $C_{i,j}$ in the x and y axis respectively. We summarize the demonstrations in Table 2 by giving the discretization partial equations, the model equation and the figure of the corresponding configuration.

In our model, Ms corresponds to the length of the undivided square, so that $\frac{Ms}{2^n}$ is the length of a square of refinement n. With that in mind, we see that the equations from the discretization in Table 2 are equal to the equations given by our model. The 4 cases presented in the table can be rotated to obtain all possible configurations (we remind here that two adjacent squares can only have a difference of 1 in their refinement level). By symmetry, we see that the equations are still equal. We therefore conclude that our model is correct according to the finite difference method.

4.3.3. Integration of external fluxes

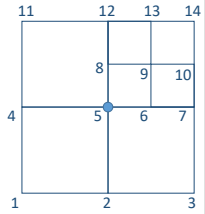
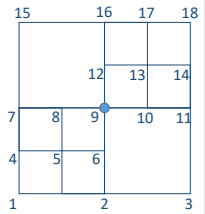
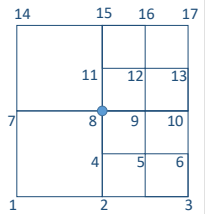
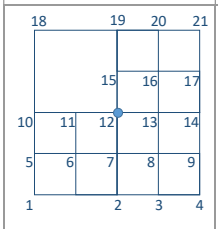
The user can add external fluxes on any node of the lattice. Let F_0 be the value of a constant external flux. This flux has to be given as a concentration per second. This value is then treated inside the python module to be adequately set as a current source on the node. Let F_{eff} be the value of this current source, corresponding to the flux F_0 set by the user on node (x_0, y_0). A flux F on a punctual zone (a node in our case) can be defined as follows:

$$F = \int_{-\infty}^{+\infty} F_0 \cdot \delta(x - x_0, y - y_0) \cdot dS$$

With $\delta(x-x_0, y-y_0)$ the Dirac function defined by its integral as follows:

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\delta(x-x_0, y-y_0)\cdot dx\cdot dy = 1$$

Table 2 Comparison between the model and the discretized diffusion equation for particular meshes. Equations are written for the central node (blue dot). The largest squares have a refinement level n and a length of $\Delta x = \Delta y = \Delta l = Ms_n$. The smaller squares have a refinement level $n+1$ and a length of $\frac{\Delta x}{2} = \frac{\Delta y}{2} = \frac{\Delta l}{2}$.

Configuration	Discretization	Model
	$\frac{\partial^2 C}{\partial x^2} = \frac{4}{7}\frac{2C_{i+\frac{1}{2},j} + C_{i+1,j} + 2C_{i-1,j} - 5C_{i,j}}{\Delta x^2}$ $\frac{\partial^2 C}{\partial y^2} = \frac{4}{7}\frac{2C_{i,j+\frac{1}{2}} + C_{i,j+1} + 2C_{i,j-1} - 5C_{i,j}}{\Delta y^2}$	$\frac{7}{8}\frac{K_0}{2^n}\frac{\partial C_5}{\partial t} = \frac{2^n D_0}{Ms^2}\frac{1}{2}(C_2 + 2C_4 + 2C_6 + C_7 + 2C_8 + C_{12} - 10C_5) - \frac{7}{8}\frac{C_5}{2^n R_0}$
	$\frac{\partial^2 C}{\partial x^2} = \frac{2}{3}\frac{2C_{i+\frac{1}{2},j} + 2C_{i-\frac{1}{2},j} + C_{i+1,j} + C_{i-1,j} - 6C_{i,j}}{\Delta x^2}$ $\frac{\partial^2 C}{\partial y^2} = \frac{2}{3}\frac{2C_{i,j+\frac{1}{2}} + 2C_{i,j-\frac{1}{2}} + C_{i,j+1} + C_{i,j-1} - 6C_{i,j}}{\Delta y^2}$	$\frac{3}{4}\frac{K_0}{2^n}\frac{\partial C_9}{\partial t} = \frac{2^n D_0}{Ms^2}\frac{1}{2}(C_2 + 2C_6 + C_7 + 2C_8 + 2C_{10} + C_{11} + 2C_{12} + C_{16} - 12C_9) - \frac{3}{4}\frac{C_9}{2^n R_0}$
	$\frac{\partial^2 C}{\partial x^2} = \frac{4}{3}\frac{2C_{i+\frac{1}{2},j} + C_{i-1,j} - 3C_{i,j}}{\Delta x^2}$ $\frac{\partial^2 C}{\partial y^2} = \frac{2}{3}\frac{2C_{i,j+\frac{1}{2}} + 2C_{i,j-\frac{1}{2}} + C_{i,j+1} + C_{i,j-1} - 6C_{i,j}}{\Delta y^2}$	$\frac{3}{4}\frac{K_0}{2^n}\frac{\partial C_8}{\partial t} = \frac{2^n D_0}{Ms^2}\frac{1}{2}(C_2 + 2C_4 + 2C_7 + 4C_9 + 4C_{11} + C_{15} - 12C_8) - \frac{3}{4}\frac{C_8}{2^n R_0}$
	$\frac{\partial^2 C}{\partial x^2} = \frac{4}{5}\frac{4C_{i+\frac{1}{2},j} + 2C_{i-\frac{1}{2},j} + C_{i-1,j} - 7C_{i,j}}{\Delta x^2}$ $\frac{\partial^2 C}{\partial y^2} = \frac{4}{5}\frac{2C_{i,j+\frac{1}{2}} + 4C_{i,j-\frac{1}{2}} + C_{i,j+1} - 7C_{i,j}}{\Delta y^2}$	$\frac{5}{8}\frac{K_0}{2^n}\frac{\partial C_{12}}{\partial t} = \frac{2^n D_0}{Ms^2}\frac{1}{2}(4C_7 + C_{10} + 2C_{11} + 4C_{13} + 2C_{15} + C_{19} - 14C_{12}) - \frac{5}{8}\frac{C_{12}}{2^n R_0}$

In our model, a node represents a surface, a part of the whole space to model. F_{eff} therefore corresponds to the flux through this "node" surface. To obtain the flux F we need to integrate this

constant flux over the surface S represented by the node (this surface depends on the mesh refinement). We therefore have:

$$F = \int_S F_{eff} \cdot dS = F_{eff} \cdot S = F_0 \int_{-\infty}^{+\infty} \delta(x - x_0, y - y_0) \cdot dS = F_0$$

Hence:

$$F_{eff} = \frac{F_0}{S}$$

Moreover, in our model a node is connected to 4 capacitors, one per neighboring mesh. These 4 capacitors are connected to the ground and a correspondence can be made to one equivalent grounded capacitor of value K_{eq} (see an example with Equation 6). We see clearly with the equation of the model that the fluxes are divided by this capacitance. Hence we have to multiply the value of the current source to implement by the equivalent capacitance of the node.

To summarize, the effective value F'_{eff} of the current source implemented in the model is defined as follows:

$$F'_{eff} = F_{eff} \cdot K_{eq} = \frac{F_0}{S} \cdot K_{eq}$$

4.4. Verilog-A implementation

In the Verilog-A model, the voltage between nX and the ground is noted as follows: `V(nX,nref)`. It corresponds to the concentration of molecules at node nX. The sources modeling the fluxes between nodes are written as follows: `I(nref,nX) <+ Fy1/2` with `I(nref,nX)` being the value of the current incoming node `nX`.

The capacitor and the resistor correspond to the following line:

```
I(nX ,nref) <+ + (ddt(V(nX ,nref))*Kn  + V(nX ,nref)/Rn);
```

with `I(nX ,nref)` the current exiting node nX and `ddt` the time derivative function. The first element represents the capacitor whereas the second represent the node connected to the reference node (i.e. the grounded reference). These elements are the same for all 8 nodes.

5. Model of the elementary mesh: finite element approach

In this part, we study an alternative to finite difference to model the elementary mesh: the finite elements approach. First, we study the finite element discretization scheme on the convection-diffusion problem in thermal physics. Then, by analogy, we build the model of the elementary mesh adapted to our problem. The complete demonstration in thermal domain is given in Appendix I. In this section, only the main results and their adaptation to the biological context are described.

5.1. Model of the elementary mesh in regular lattices

The model of the elementary mesh in regular lattices is straightforward. We use the analogy between temperature, molecule concentration and voltage on the one hand and heat flux, molecule flux and

current on the other hand. Results from Appendix I can be used. The electrical equivalent model of the elementary mesh is composed of 6 resistors (Fig. 10). The values of the resistance are:

$$R_{12} = R_{34} = \frac{1}{D} \cdot \frac{6LW}{-L^2 + 2W^2}$$

$$R_{13} = R_{24} = \frac{1}{D} \cdot \frac{6LW}{L^2 + W^2}$$

$$R_{14} = R_{23} = \frac{1}{D} \cdot \frac{6LW}{-W^2 + 2L^2}$$

where D is the diffusion constant associated to the diffusing molecule.

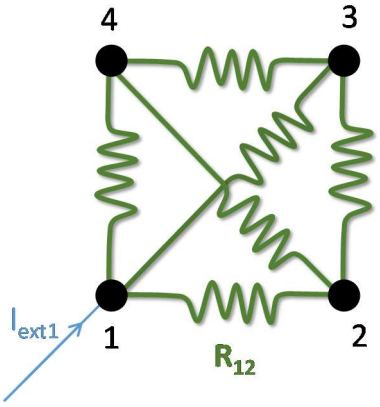


Fig. 10 The electrical mesh used in the diffusion model. Here all nodes are connected to one another with resistors. External currents can be taken into account on each node.

The degradation of molecules is equivalent to the convection in thermal physics. On Fig. 10, it corresponds to the external current $I_{ext,i}$. Equations giving $\varphi_{ext,i}$ in the case of thermal convection are given in Appendix:

$$\begin{pmatrix} \varphi_{ext,1} \\ \varphi_{ext,2} \\ \varphi_{ext,3} \\ \varphi_{ext,4} \end{pmatrix} = -d_X \cdot \frac{LW}{36} \cdot \begin{pmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{pmatrix} \cdot \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{pmatrix}$$

By opposition to the finite difference model, the molecule flow $\varphi_{ext,i}$ outing from a node does not depend only on the concentration C_i on that node. Thus, VCCS have been implemented in the Verilog-A model. Moreover, a unitary capacitor is added to each node to model the fluxes related to the accumulation of molecules.

The 4-node model only allows to use regular matrix, which is very limiting in our context. This is why 5-node, 6-node, 7-nodes and 8-nodes models have to be investigated.

5.2. Model of the elementary mesh in adaptive lattices

5.2.1. Direct computation

The theory described in Appendix I is generic and may be applied on mesh with any number of nodes. Thus, it is possible to compute the rigidity matrix, the external flux matrix and the equivalent electronic circuit for a model with extra nodes at the middle of edges. Let us start with the 5-node model.

K_e matrix

The additional node is hereafter called a segment node. This node can be positioned between either of the following nodes: node 0 and 1, 1 and 2, 2 and 3 and 3 and 0, and is respectively named 01, 12, 13 and 30. The canonic base used is either $[1, x, y, xy, x^2]$ when the additional node is on an horizontal edge or $[1, x, y, xy, y^2]$ when the additional node is on a vertical edge. Computation has been carried out with MAPLE. The analytic expression of the rigidity matrix is too complex to be written explicitly here. Equation 9 correspond to the case of in which the segment node is between 1 and 2 and with $W = L$.

Equation 9

$$\mathbf{K_e} = \frac{D}{6} \cdot \begin{pmatrix} 4 & -1 & -2 & -1 & 0 \\ * & 12 & 7 & -2 & -16 \\ * & * & 12 & -1 & -16 \\ * & * & * & 4 & 0 \\ * & * & * & * & 32 \end{pmatrix}$$

The fifth row and column correspond to segment node. We first notice that we also obtain a symmetrical matrix, as expected. Moreover, also expected is the sum of each row and column, which is equal to 0.

The rigidity matrix give the value of the resistance that needs to be implemented between each couple of nodes of the electric equivalent model. With the appearance of a fifth node, negative resistance appears between node 1 and node 2, the two nodes having a segment node inbetween. The resistance value between two adjacent corner nodes, named $R_{adj,ij}$, is the same as with the 4-node element when the two adjacent nodes do not have a segment node inbetween (e.g. node 0 and 1, node 2 and 3 and node 3 and 0). The diagonal resistance value between two diagonal nodes, R_{diag}, is unchanged from the previous model. Finally, resistors connecting the fifth node to the other nodes are named as follows: $R_{adj,X}$ is the resistance value between a segment node and its corner nodes neighbors (here between nodes 12 and 1 and between nodes 12 and 2), $R_{diag,X}$ is the resistance value between a segment node and its opposite corner nodes (here between nodes 12 and 0 and between nodes 12 and 3).

From a modeling perspective, we can generalize the values found for the resistors. We obtain a generic expression including Boolean parameters X_{ij} valid for 4 or 5 nodes. For instance, with $W = L$,

$$R_{adj,ij} = \frac{6}{1-8*X_{ij}} \qquad R_{adj,X} = \frac{6}{16} X_{ij} \qquad R_{diag} = 3 \qquad R_{diag,X} = 0$$

F_e matrix

We also computed the matrices $\mathbf{F_e}$ for five nodes element. Again, the analytic expression of the rigidity matrix is too complex to be written explicitly here. As above, we only show the matrix for the case with a segment node at node 12 and $W = L$.

$$\mathbf{F_e} = -d_X \cdot \frac{L^2}{180} \cdot \begin{pmatrix} 20 & -5 & -10 & 10 & 30 \\ * & 14 & 4 & -10 & -18 \\ * & * & 14 & -5 & -18 \\ * & * & * & 20 & 30 \\ * & * & * & * & 96 \end{pmatrix} \times \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_{12} \end{pmatrix}$$

Again, negative resistance values appear with the presence of an additional fifth node. We also computed the analogy to the electronic model by implementing VCCS and generalized it. For

simplicity's sake, we will show here the implementation of one such VCCS for a corner node and for a segment node. A simple circular permutation allows the reader to find the other VCCS. As above, `Xab` is a Boolean value equal to 1 when a segment node is present between node a and b and 0 otherwise.

We also create the constant deg as follows:

$$deg = -d_X \frac{L^2}{180}$$

For the corner node 0 we obtain:

```
I(nref, n0) <+ deg *((20 - 4*(X01+X30)) * V(n0, nref) +(10 - 6*X01 -
15*(X12+X30)) * V(n1, nref) +(5 - 15*(X01+X12+X23+X30)) * V(n2, nref)
+(10 - 15*(X01+X23) -20*X12 - 6*X30) * V(n3, nref));
```

For the segment node 01 we obtain:

```
I(nref, n01) <+ deg * X01*(-18*V(n0, nref) - 18*V(n1, nref) + 30*V(n2,
nref)+ 30*V(n3, nref));
```

In order to simulate any kind of lattice, we need a generic model of an 8-node element that can be parametrized by the Boolean parameters `Xij`. Such approach has two main shortcomings. Firstly, the canonical base for such an element has to be defined, which is not very intuitive. For a 6-node mesh with one segment node on each direction, the canonical base $[1, x, y, xy, x^2, y^2]$ seems to be suitable. For other configurations (6-node mesh with segment nodes on the same direction, 7-node and 8-node meshes) we need to introduce 3rd degree terms in the base. It was unclear whether to add x^2y and xy^2 or x^3 and y^3. Secondly, generic analytic expressions are already very complex with 5 nodes. The definition of a single model that covers all the possible configurations, as it has been done for finite differences, seems very tricky.

5.2.2. Composition of triangle models

Because of these shortcomings, we change our approach: the rectangular mesh is seen as a composition of triangular elements. In the following, let us consider only square elements. Thus, triangles are right-angled and isosceles.

Model of an elementary triangle

Again, the model for the elementary isosceles right-angled triangle with a side of length L (see Fig. 11) is computed with a generic formalism described in the Appendix I. The canonical base used to find the $\mathbf{K_e}$ and $\mathbf{F_e}$ matrices is $[1, x, y]$.

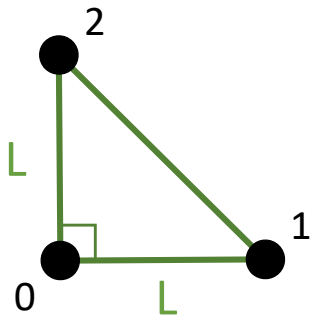


Fig. 11 Model of a triangular element.

We obtain the following $\mathbf{K_e}$ matrix:

$$\mathbf{K_e} = D \cdot \frac{1}{2} \cdot \begin{pmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

We observe that to reach the value 0, the resistance value has to be infinite. This means that nodes 1 and 2 are not directly connected. Nodes 0 and 1 and nodes 0 and 2 are connected by resistance values of 2 (relative values to a D coefficient).

Moreover, we obtain the following $\mathbf{F_e}$ matrix:

$$\mathbf{F_e} = -d_X \cdot \frac{L^2}{48} \cdot \begin{pmatrix} 2 & 1 & 1 \\ * & 2 & 1 \\ * & * & 2 \end{pmatrix} \times \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix}$$

Construction of the rectangular model

The superposition of 4 such elementary triangles in 4 different orientations but with the right-angled corner at the center gives rise to a 4-node square element (see Fig. 12 left element). Upon addition of a segment node, the triangle located at this segment is divided into two smaller triangles (see Fig. 12 middle and right elements). The smaller triangle is computed following the same model, where L is replaced by $L/2$ and the right-angled corner of the triangle corresponds to the additional segment node. On Fig. 12, the resistance values are relative: in the model they are multiplied by $\frac{1}{D}$.

On Fig. 12, the resistance values correspond to the value of the equivalent resistance. Indeed, two resistors connect a corner node to the central node, due to the existence of two triangles. If R_1 and R_2 are two resistors connecting the same two nodes, here is the formula of the equivalent resistor that can replace the two resistors:

$$R_{eq} = \frac{R_1 R_2}{R_1 + R_2}$$

As the two resistors connecting a corner node and the central node have a value of 2 when no segment node is present, the equivalent resistor has a value of 1.

To generalize this approach to rectangular meshes, we would need to use generic isosceles triangles.

Verilog-A model

In order to keep square elements with 8 terminals, the 9th node is kept internal to the model. It does not appear outside of the Verilog-A model, so that we can still use the same mesher to divide space.

In the model, the central node is treated like the other nodes (it has a degradation flux and is connected to its neighbor nodes). However, it has no capacitor, which is irrelevant in an operating point simulation (capacitors only influence the dynamics of a system and not its steady state).

5.2.3. Boundary conditions

As for finite differences, three boundary conditions can be defined: no-flux (blocked diffusion), fixed concentration and further diffusion. When nodes have a no-flux conditions, they are not connected to any external device. Fixed concentration means the application of a fixed voltage to the node. This can be done with an external fixed voltage source. Finally, for further diffusion, the model to implement depends on the position of the border. These conditions are described in the Appendix I.

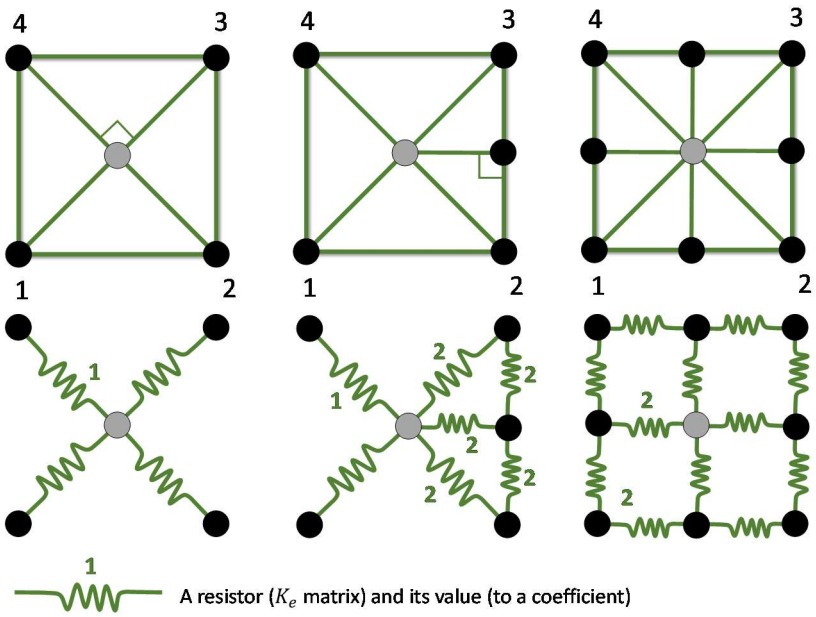


Fig. 12 Square elements modeled with triangles. The upper line shows a representation of the elements and the number of nodes they contain (the grey node is the central node and is an internal node of the model). The lower line show the value of the resistors connecting each node after implementing the adequate triangles. On the left, an element with 4 nodes, on the center with 5 nodes and on the right with 8 nodes.

We see again that we need to model a VCCS dependent on the voltage of the two nodes located on the border of the lattice.

In conclusion, these fluxes are to be represented by VCCS depending on the potential of the node located on the border. The relative contribution has been established. As for the absolute value of these fluxes ($\frac{1}{R}$), it corresponds to the diffusion coefficient D of the molecule (as defined in the model described in section 4).

6. Conclusion

We described in this chapter the tool we developed for the modeling and the simulation of space- and time-dependent biological system. The model is written and simulated with the SPICE language. It is generated by a software environment combining C++ and Python. First, space is discretized in an adaptive lattice. Then the tool assembles the building blocks of the model (model of each mesh of the lattice, boundary conditions, local biological models). Two types of discretization schemes have been implemented, finite difference and finite elements.

In the next chapter, we first validate the model by comparing simulation and theory on well-chosen examples for which the reaction-diffusion equation can be analytically integrated. Then, the tool is applied on some use cases.

7. Reference

Amar, Patrick, Gilles Bernot, and Vic Norris. 2004. “HSIM: A Simulation Programme to Study Large Assemblies of Proteins.” *Journal of Biological Physics and Chemistry* 4: 79–84.

Amar, Patrick, and Loïc Paulevé. 2015. “HSIM: A Hybrid Stochastic Simulation System for Systems Biology.” *Electronic Notes in Theoretical Computer Science* 313 (May). Elsevier: 3–21.

doi:10.1016/J.ENTCS.2015.04.016.

Blundell, Stephen, and Katherine M. Blundell. 2010. *Concepts in Thermal Physics*. Oxford University Press.

Digele, G., S. Lindenkreuz, and E. Kasper. 1997. "Fully Coupled Dynamic Electro-Thermal Simulation." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 5 (3): 250–57. doi:10.1109/92.609867.

Ermentrout, G.Bard, and Leah Edelstein-Keshet. 1993. "Cellular Automata Approaches to Biological Modeling." *Journal of Theoretical Biology* 160 (1): 97–133. doi:10.1006/jtbi.1993.1007.

Evans, Gwynne A., J. M. (Jonathan M.) Blackledge, and Peter D. Yardley. 2000. *Numerical Methods for Partial Differential Equations*. Springer London.

Fick, Adolph. 1855. "On Liquid Diffusion." *Philosophical Magazine Series 4* 10 (63): 30–39.

Garci, Maroua, Jean-Baptiste Kammerer, and Luc Hebrard. 2014. "Compact Modeling and Electro-Thermal Simulation of Hot Carriers Effect in Analog Circuits." In *2014 IEEE 12th International New Circuits and Systems Conference (NEWCAS)*, 125–28. IEEE. doi:10.1109/NEWCAS.2014.6933999.

Gillespie, Daniel T. 1977. "Exact Stochastic Simulation of Coupled Chemical Reactions." *The Journal of Physical Chemistry* 81 (25). American Chemical Society: 2340–61. doi:10.1021/j100540a008.

Haiech, J, Y Gendrault, and MC Kilhoffer. 2014. "A General Framework Improving Teaching Ligand Binding to a Macromolecule." *Biochimica et Biophysica Acta (BBA)-Molecular Cell Research* 1843 (10): 2348–55. http://www.sciencedirect.com/science/article/pii/S0167488914000974.

Harrison, Jonathan U., and Christian A. Yates. 2016. "A Hybrid Algorithm for Coupling Partial Differential Equation and Compartment-Based Dynamics." *Journal of The Royal Society Interface* 13 (122): 20160335. doi:10.1098/rsif.2016.0335.

Hecht, F. 2012. "New Development in Freefem++." *Journal of Numerical Mathematics* 20 (3–4). De Gruyter: 251–66. doi:10.1515/jnum-2012-0013.

Hoops, Stefan, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. 2006. "COPASI—a Complex Pathway Simulator." *Bioinformatics* 22 (24). Oxford Univ Press: 3067–74.

Johnson, Claes. 2009. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Dover Publications.

Konkoli, Zoran. 2011. "Safe Uses of Hill's Model: An Exact Comparison with the Adair-Klotz Model." *Theoretical Biology & Medical Modelling* 8 (1). BioMed Central Ltd: 10. doi:10.1186/1742-4682-8-10.

Krencker, Jean-Christophe, Jean-Baptiste Kammerer, Yannick Hervé, and Luc Hébrard. 2010. "Direct Electro-Thermal Simulation of Integrated Circuits Using Standard CAD Tools." In *Thermal Investigations of ICs and Systems (THERMINIC), 2010 16th International Workshop on*, 1.

Lecca, Paola, Adaoha E C Ihekwaba, Lorenzo Dematté, and Corrado Priami. 2010. "Stochastic Simulation of the Spatio-Temporal Dynamics of Reaction-Diffusion Systems: The Case for the Bicoid Gradient." *Journal of Integrative Bioinformatics* 7 (1): 150. doi:10.2390/biecoll-jib-2010-150.

Loew, L M, and J C Schaff. 2001. "The Virtual Cell: A Software Environment for Computational Cell Biology." *Trends in Biotechnology* 19 (10): 401–6. doi:10.1016/S0167-7799(01)01740-1.

Madec, Morgan, Yves Gendrault, Christophe Lallement, and Jacques Haiech. 2012. "A Game-of-Life like Simulator for Design-Oriented Modeling of BioBricks in Synthetic Biology." In *Conference Proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2012:5462–65. doi:10.1109/EMBC.2012.6347230.

Madec, Morgan, Christophe Lallement, and Jacques Haiech. 2017. "Modeling and Simulation of Biological

Systems Using SPICE Language." *PloS One* 12 (8). doi:10.1371/journal.pone.0182385.

Mazumder, Sandip, and Sandip Mazumder. 2016. "Chapter 2 – The Finite Difference Method." In *Numerical Methods for Partial Differential Equations*, 51–101. doi:10.1016/B978-0-12-849894-1.00002-0.

Michaelis, Von L, and Miss L Maud Menten. 1913. "Die Kinetik Der Invertinwirkung." *Biochemistry* 29: 332–69.

Nenzi, P, and H Vogt. 2011. "Ngspice Users Manual Version 23."

Schaff, J, C C Fink, B Slepchenko, J H Carson, and L M Loew. 1997. "A General Computational Framework for Modeling Cellular Structure and Function." *Biophysical Journal* 73 (3): 1135–46. doi:10.1016/S0006-3495(97)78146-3.

Spill, Fabian, Pilar Guerrero, Tomas Alarcon, Philip K Maini, and Helen Byrne. 2015. "Hybrid Approaches for Multiple-Species Stochastic Reaction-Diffusion Models." *Journal of Computational Physics* 299 (October). Elsevier: 429–45. doi:10.1016/j.jcp.2015.07.002.

Takahashi, Kouichi, Satya Nanda Vel Arjunan, and Masaru Tomita. 2005. "Space in Systems Biology of Signaling Pathways--towards Intracellular Molecular Crowding in Silico." *FEBS Letters* 579 (8): 1783–88. doi:10.1016/j.febslet.2005.01.072.

Chapter 7
Validation and Results

In this chapter, we first focus on the validation of the models. For this purpose, two cases are considered. The first one is the transvers diffusion from one border (where molecule are synthesized) to the other. This is a 1-D problem for which an analytic solution can be found at steady state. The second case is the radial diffusion from a punctual source at the center of the lattice. Again, an analytical solution, which will serve as a reference for the validation process, exists for this. Then, the validated model is applied on some actual biological problems.

1. Validation of the finite difference model

1.1. Transverse diffusion

First, we consider a 2D square of length L. A reaction generates a constant and uniform flux of molecules ϕ_0 on each point of the left border and the concentration of the molecule is set to 0 on the right border. At the top and bottom border, a no-flux boundary condition is applied (Fig. 1). Thus, the concentration only depends on the position along the X axis.

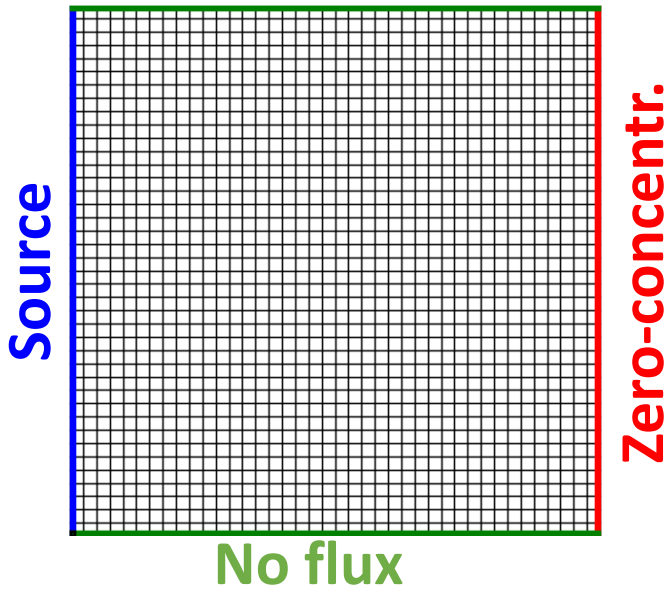


Fig. 1 Description of the transverse diffusion problem and reference lattice composed on 40x40 squares.

1.1.1. Analytical solution

The PDE equation governing the space and time variation of the concentration is reminded here:

Equation 1

$$\frac{\partial C}{\partial t} = D \cdot \Delta C - d \cdot C$$

where D is the diffusion constant and d the degradation constant. Taking into account that $C(x, y, z, t) = C(x, t)$, the PDE at the steady state becomes a simple ODE:

$$D \cdot \frac{d^2C}{dx^2} - d \cdot C(x) = 0$$

with the following boundaries conditions:

$$C(L) = 0 \ ; \ \frac{dC}{dx}(0) = -\phi_0$$

An analytic solution can be computed for this equation. If $d \neq 0$, the solution of this equation is a combination of two exponential:

$$C(x) = C_1 \cdot e^{\frac{x}{L_0}} + C_2 \cdot e^{-\frac{x}{L_0}}$$

with C_1 and C_2 two integration constants and $L_0 = \sqrt{\frac{D}{d}}$. Two relationships can be established between C_1 and C_2 from the boundary conditions:

Equation 2

$$C_1 \cdot e^{\frac{L}{L_0}} + C_2 \cdot e^{-\frac{L}{L_0}} = 0$$

And

$$\frac{C_1}{L_0} - \frac{C_2}{L_0} = -\phi_0$$

Reinjecting $C_1 = C_2 - \phi_0 \cdot L_0$ in Equation 2 leads to:

$$C_2 \cdot e^{\frac{L}{L_0}} + C_2 \cdot e^{-\frac{L}{L_0}} - \phi_0 \cdot L_0 \cdot e^{\frac{L}{L_0}} = 2 \cdot C_2 \cdot sh\left(\frac{L}{L_0}\right) - \phi_0 \cdot L_0 \cdot e^{\frac{L}{L_0}} = 0$$

Hence,

$$C_2 = \frac{\phi_0 \cdot L_0 \cdot e^{\frac{L}{L_0}}}{2 \cdot sh\left(\frac{L}{L_0}\right)}$$

And

$$C_1 = \phi_0 \cdot L_0 \left(\frac{e^{\frac{L}{L_0}}}{2 \cdot sh\left(\frac{L}{L_0}\right)} - 1 \right) = \frac{\phi_0 \cdot L_0}{2 \cdot sh\left(\frac{L}{L_0}\right)} \cdot e^{-\frac{L}{L_0}}$$

Finally,

$$C(x) = \frac{\phi_0 \cdot L_0}{2 \cdot sh\left(\frac{L}{L_0}\right)} \cdot \left(e^{\frac{x-L}{L_0}} + e^{\frac{-x+L}{L_0}} \right) = \phi_0 \cdot L_0 \cdot \frac{sh\left(\frac{L-x}{L_0}\right)}{sh\left(\frac{L}{L_0}\right)}$$

If $d = 0$, the solution can be obtained by a double integration:

$$C(x) = A \cdot x + B$$

Boundary conditions leads to $A = -\phi_0$ and $B = L \cdot \phi_0$. Hence,

$$C(x) = \phi_0 \cdot (L - x)$$

1.1.2. Simulation results

The system is simulated with four different lattices, and with or without degradation. Lattice A and C are regular lattices composed respectively of 40x40 and 160x160 squares. Lattice B corresponds to Lattice A with two vertical refinement zones (Fig. 2A). Finally, lattice D corresponds to lattice A with a circular refinement zone at the center of the lattice (Fig. 2B). Characteristics of the lattices are given in Table 1.

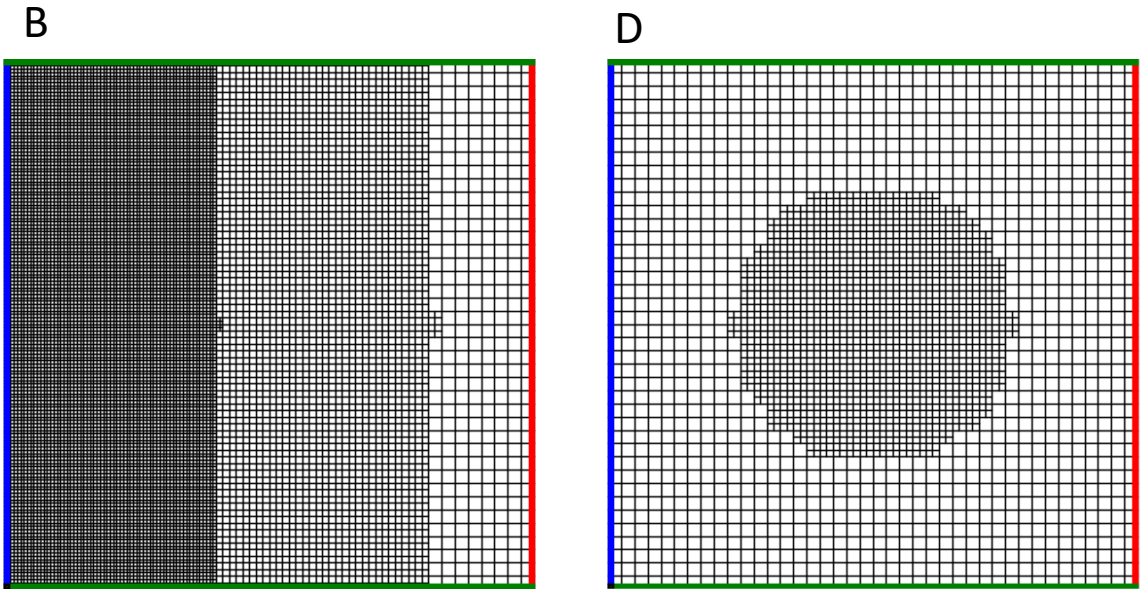


Fig. 2 Lattice B and D, obtained from the Lattice A (Fig. 1) by vertical or circular refinement zone.

Table 1 Characteristics of the lattices used for the validation.

Lattice number	Initial Lattice	Refinement	Number of nodes	Number of meshes
A	40 x 40		1 681	1 600
B	40 x 40	n=2 for x < 40 n=1 for x < 80	13 399	13 132
C	160 x 160		25 921	25 600
D	40 x 40	n=1 for r < 25	2 779	2 656

Fig. 3 shows the accordance between the simulation results and the analytical solution. Error introduced by the refinement is analyzed on Fig. 4. As expected, the most refined lattice (lattice C) is the one giving the best results. The adaptive lattice is a good tradeoff: the maximal error is about 2.5 % but it reduces the number of nodes by a factor of about 2. Fig. 5 also shows the simulation results in semilog scale for three different cross-sections at different values of y. These last results confirm the validity of the model, even for small concentrations far from the sources.

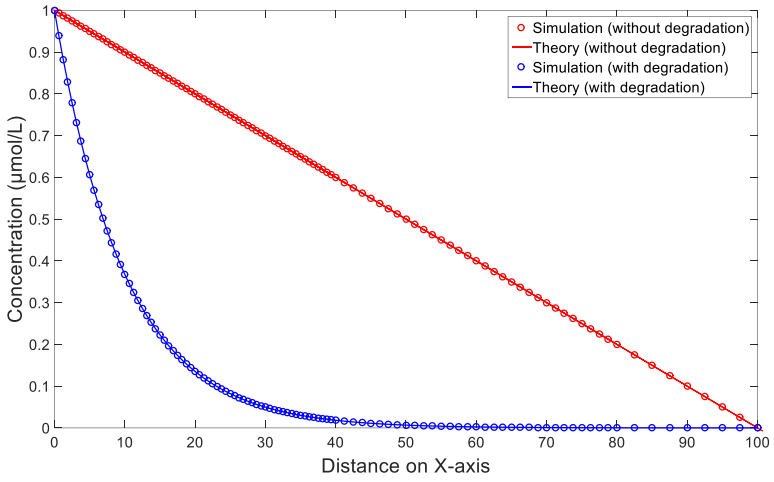


Fig. 3 Simulation results and comparison with the theoretical models for the lattice A. Parameters used for the simulation are the following: L=100, D=10 , d=0.1, ϕ_0=0.01 without degradation, and ϕ_0=0.1 with degradation.

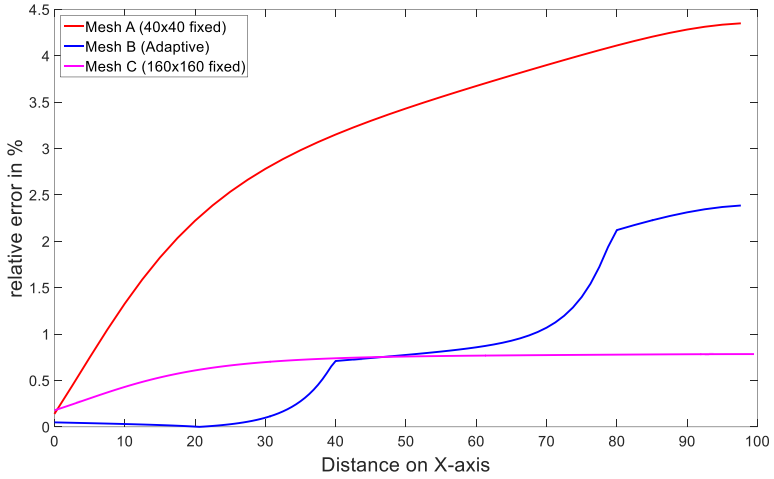


Fig. 4 Comparison of the relative error between simulation results and analytic solution for the lattice A, B and C. Parameters used for the simulation are the following: L=100, D=10 , d=0.1 and ϕ_0=0.1 with degradation.

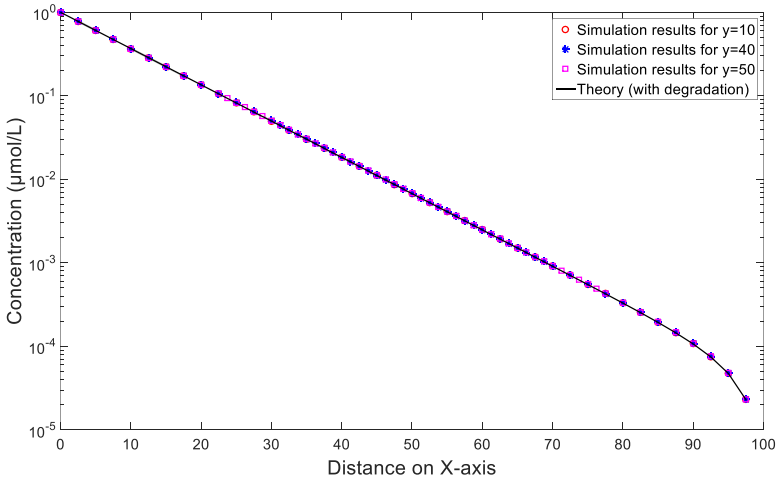


Fig. 5 Comparison of the analytical solution and the simulation results for lattice D for different values of y. Parameters used for the simulation are the following: L=100, D=10 , d=0.1 and ϕ_0=0.1.

1.2. Radial diffusion from a central source

In this case, we still consider the same 2D square of length L. The no-flux boundaries condition is applied on each border and an incoming flux is applied to the center of the square O.

1.2.1. Analytical solution

For analytic computation, polar coordinates (r, θ) centered on O are preferred. As our problem is isotropic, the concentration is independent of θ and, at the steady state, Equation 1 can be rewritten as follows:

$$D \cdot \left(\frac{d^2C}{dr^2} + \frac{1}{r} \cdot \frac{dC}{dr} \right) - d \cdot C(r) = 0$$

The analytical solution of this equation is a linear combination of the modified Bessel functions. More precisely, $C(r)$ can be written as follows:

$$C(r) = C_1 \cdot I_0\left(\sqrt{\frac{d}{D}} \cdot r \right) + C_2 \cdot K_0\left(\sqrt{\frac{d}{D}} \cdot r \right)$$

With I_0 and K_0 the modified Bessel functions of order 0 and of the first and second kind respectively and C_1 and C_2 the integration constants.

The first boundary condition is $C(\infty) = 0$. As $\lim_{x \to \infty} I_0(x) = \infty$ and $\lim_{x \to \infty} K_0(x) = 0$, it is only met if C_1 is equal to zero. Computation of C_2 is a bit more tricky. Because the source is punctual, the outgoing flux is $-\phi_0 \cdot \delta(r)$ where $\delta(r)$ is the Dirac function. Neuman boundary condition cannot be used for $r = 0$ because $\delta(0)$ is infinite. We have to integrate the flux along the border Σ of a small circle on radius δr around O.

$$\int_{\Sigma} \mathbf{grad}(C) \cdot \mathbf{n} \cdot dl = \int_{\Sigma} \frac{dC}{dr}(\delta r) \cdot dl = \frac{dC}{dr}(\delta r) \int_{\Sigma} dl = 2 \cdot \pi \cdot \delta r \cdot \frac{dC}{dr}(\delta r) = -\phi_0$$

By definition:

$$\frac{dC}{dr}(r) = C_2 \cdot \frac{dK_0}{dr}(r) = -C_2 \cdot K_1(r)$$

where K_1 is the modified Bessel function of second kind and at order 1. The asymptotic form of Bessel function $K_1(r)$ for $r \ll 1$ is:

$$K_1(r) \sim \frac{1}{r}$$

Thus,

$$-C_2 \cdot 2 \cdot \pi \cdot \delta r \cdot K_1(\delta r) \sim -C_2 \cdot 2 \cdot \pi = -\phi_0$$

Finally,

$$C(r) = \frac{\phi_0}{2 \cdot \pi} \cdot K_0\left(\sqrt{\frac{d}{D}} \cdot r \right)$$

1.2.2. Results

Eleven different lattice layouts have been compared. They are summarized in Table 2.

Table 2 Description of the lattices for the benchmark. For all of them, the initial (unrefined) lattice, the refinement zones, the number of nodes and elementary meshes, as well as the simulation time for steady state (SSA) analysis and time-course (TC) simulations are given, both for NGSpice and Spectre. Values in the refinement column are the diameter of the refinement zone (centered on O). Computation time with Spectre and NGSPICE have been obtained on a 12-core 2.6-GHz CPU with 40 Go of RAM. TC simulations are performed with a built-in adaptive time step algorithm configured to simulate over 100 seconds with a minimal time step of 5 s.

Lattice	Description				Model		CPU time (s)			
	Initial Lattice	Refinement			Number of nodes	Number of meshes	Spectre		NG Spice	
		n = 1	*n = 2*	*n = 3*			SSA	TC	SSA	TC
1	40 x 40				1 681	1 600	1.5	1.7	1.4	9.4
2	40 x 40	25			2 779	2 656	2.1	4.6	3.1	20.1
3	20 x 20				441	400	0.6	0.5	0.4	2.4
4	80 x 80				6 561	6 400	5.5	8.7	11.3	59.1
5	40 x 40	100			6 561	6 400	5.3	9.3	11.0	57.4
6	40 x 40	100	50	25	36 991	36 514	111.0	202.0	323.0	996.0
7	100 x 100				10 201	10 000	14.5	18.7	20.5	80.8
8	25 x 25	50	25		3 997	3 844	3.1	6.6	5.6	29.9
9	25 x 25	30	15		1 969	1 855	1.29	3.4	2.2	13.2
10	25 x 25	20	5		1 109	1 024	0.7	1.1	1.3	6.3
11	25 x 25		50		8 647	8 446	7.7	15.4	19.0	74.4

When comparing the steady state values of the nodes for the different layouts, it appears that they are very similar whatever far they are from the source. For this reason, we choose to show only the close vicinity of the source in the results. Mesh #4 and Mesh #5 are equivalent but have been obtained by two different ways: one directly, the other using a refinement on the whole area. Thus, even if their netlists are different, the ODE sets generated by model compilation and thus the simulation results are identical. This indicates that the refinement has been well considered in the model of the elemental mesh. Moreover, we can see that most results obtained with different lattices overlap and are in accordance with the theoretical model (Fig. 6).

However, we note a significant difference at the source. This difference is an effect of discretization. It is illustrated on Fig. 7. When PDEs are solved with a finite difference model, the value computed for the concentration at a given node is the integral of the concentration on a surface around this node. Obviously, the size of this surface depends on the mesh refinement around the given node. The blue curve on Fig. 7 corresponds to the theoretical model and thus tends to infinity for $r = 0$. Consider now the center of the lattice. For large meshes (typ. Lattice #3), the integration surface is larger and the value at the central point is lower. For thinner meshes (typ. Lattice #5), the integration surface decreases and the value at the central point increases. This phenomenon is amplified around the center where the gradient of concentration is higher than near the borders. The difference between observed curves shows that some lattices used for simulation are not refined enough to produce accurate results near the center (Fig. 6). Conversely, if the area of interest is far from the center, all the lattices give equivalent results.

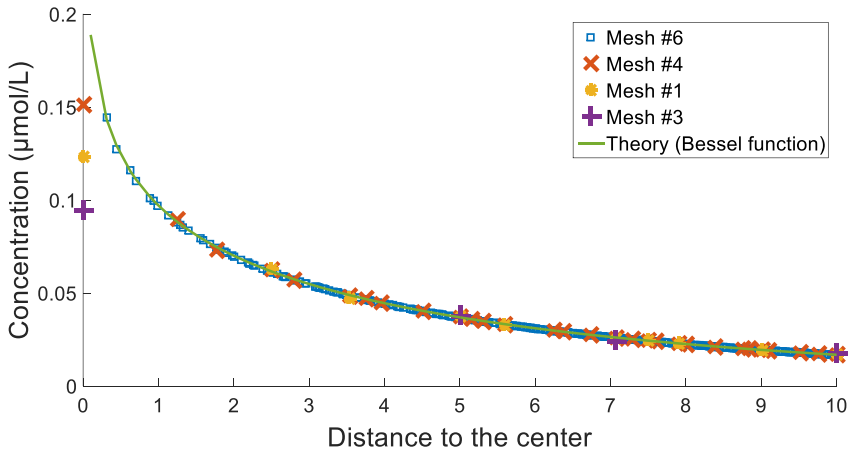


Fig. 6 Simulation results of the steady state concentration on nodes according to their distance toward the source for different lattice layouts, and comparison with the analytic solution. L=100, D=10 , d=0.1 and ϕ_0=0.628.

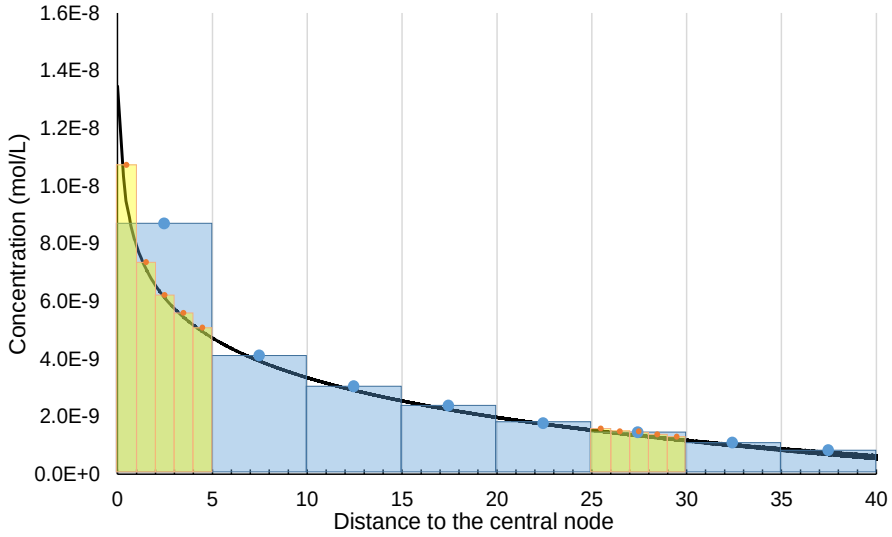


Fig. 7 Illustration of the discretization effect on the central node. Blue and yellow rectangles illustrate the integration surface of a lattice with respectively a low and high refinement level.

The relative error between simulation results and the analytic solution as a function of the lattice is represented on Fig. 8. As expected, the finer lattice, the more accurate the results.

Obviously, the drawback of a highly-refined grid is the increased number of nodes and meshes and thus an increase in the computation time, as illustrated on Fig. 9. In addition, the two simulators compatible with our tool (namely Spectre and NGSPICE) have been compared on this benchmark of models. On Fig. 9A, we observe that for a steady state analysis, the computation time with NGSPICE and Spectre are comparable. A linear increase in the computation time with respect to the complexity of the model is observed. Conversely, Fig. 9B shows that for transient analysis, the solver of Spectre is about 10 times more efficient. Finally, the tradeoff between accuracy and computation time is highlighted on Fig. 9C.

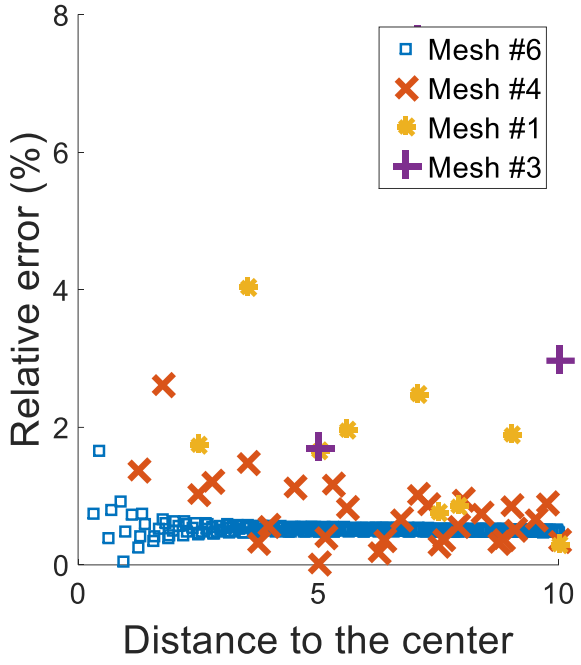


Fig. 8 Relative error between simulation and analytical solution as a function of the distance to center for different lattices.

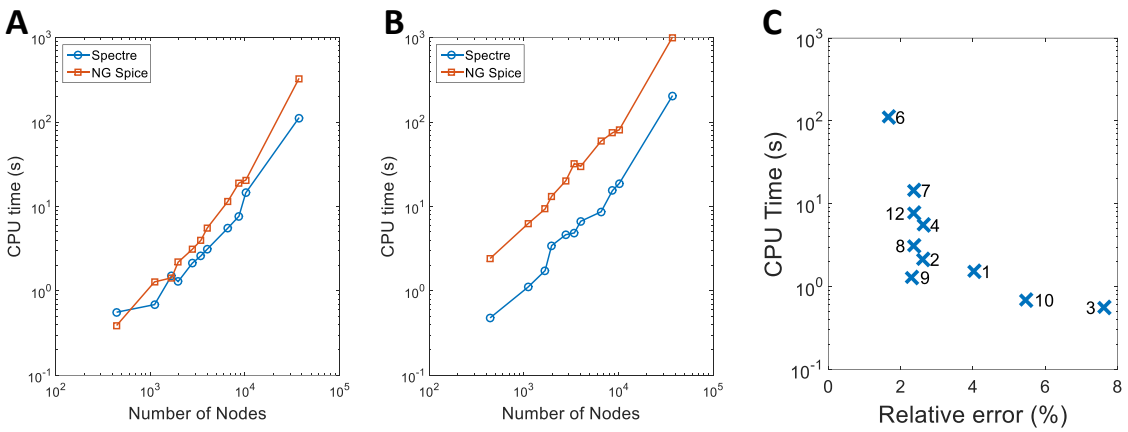


Fig. 9 Computation time for the different lattices. (A) gives a comparison of the evolution of the computation time as a function of the number of nodes for a steady state simulation with Spectre and NG Spice. (B) is the same comparison for a time course analysis. Finally, (C) gives the position of the different lattices on maximal relative error versus computation time map for a steady state simulation with Spectre. Simulation have been carried out on a 12-core 2.6-GHz CPU with 40 Go of RAM.

Both examples demonstrate the validity of the finite difference model. On the one hand, the results of simulations on these use cases are in accordance with the theoretical expectations. Moreover, we obtain strictly equivalent models with a fine lattice and with a coarse lattice that has been refined, which proves the validity of our refinement equations. Finally, we also demonstrate that results obtained with an adaptive lattice get very close to those obtained with a fine lattice while preserving a good compromise between precision and complexity of the model. Transitions between areas of

different degrees of refinement seem to introduce some artifacts but they remain acceptable in terms of model accuracy.

2. Validation of the finite element model

2.1. Comparison with finite difference model

First, we compare the results obtained with the finite element model to those obtained with the finite difference model. The radial diffusion with the lattices described in Table 2 are used. First, let us consider Lattice #7 which is a regular lattice (Fig. 10). Results are in accordance with those obtained with finite differences and thus, with theory. As expected, the 4- and 5-nope models give equivalent results on a regular lattice. Near the source, the triangular model is closer to the theory than the finite difference model and the 4-nodes model (2% of error). Nevertheless, a drift seems to appear as one moves away from the center by opposition to other models for which the error vanished (Fig. 11). This drift remains to be explained.

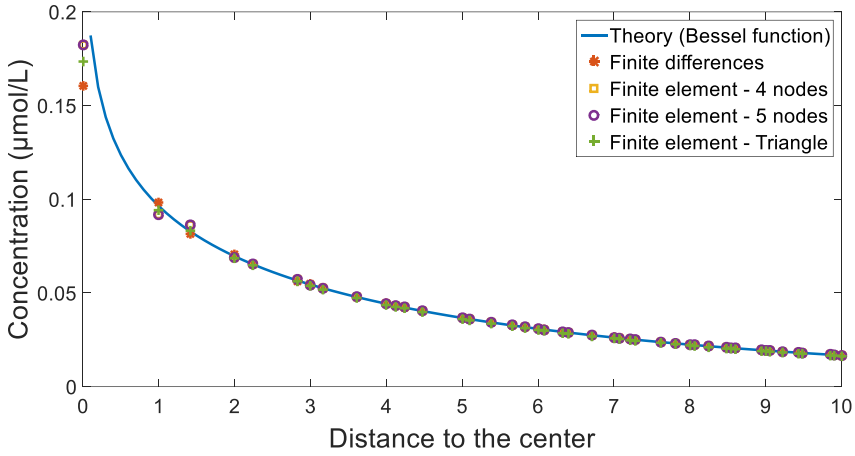


Fig. 10 Simulation results obtained with different versions of the model for the steady state concentration on nodes according to their distance toward the source for different lattices for the Lattice 7 (regular). L=100, D=10 , d=0.1 and ϕ_0=0.628.

Lattice #11 is a refined lattice which can be used to validate the refinement model. The 4-node model is no longer usable for this lattice. Again, results are in accordance with those obtained with finite differences and thus, with theory. The same observation as above can be done regarding the triangular model (Fig. 12 and Fig. 13).

Finally, we compare the computation time for several lattices (Table 3). Results show that even though the finite element model is more complex than the finite difference one, the computation time is similar for small lattices (<7000 nodes). For larger lattices however, the gap between these two approaches increases.

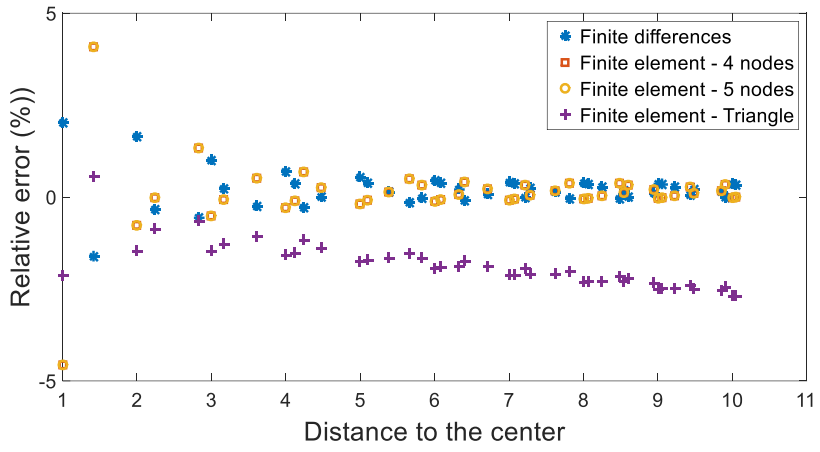


Fig. 11 Relative results obtained between the different versions of the model and the analytic solution for the steady state concentration on nodes according to their distance toward the source for the Lattice 7 (regular). L=100, D=10 , d=0.1 and ϕ_0=0.628.

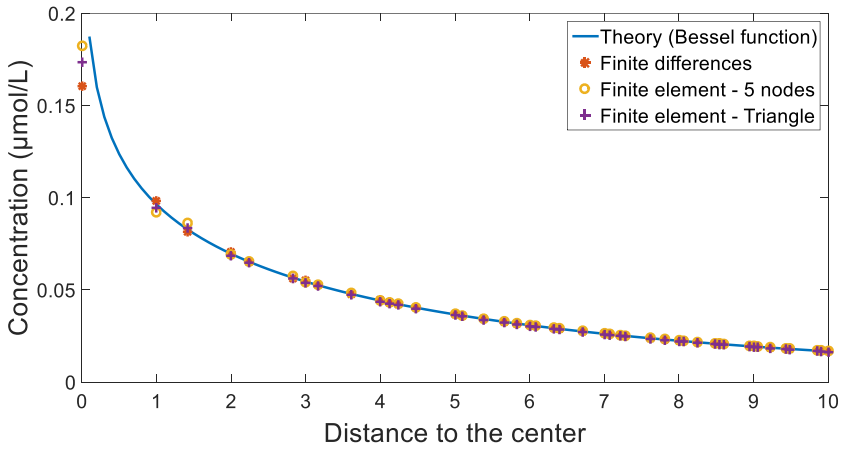


Fig. 12 Simulation results obtained with different versions of the model for the steady state concentration on nodes according to their distance toward the source for different lattice layouts, and comparison with the analytic solution for the Lattice 11 (refined near the source). L=100, D=10 , d=0.1 and ϕ_0=0.628.

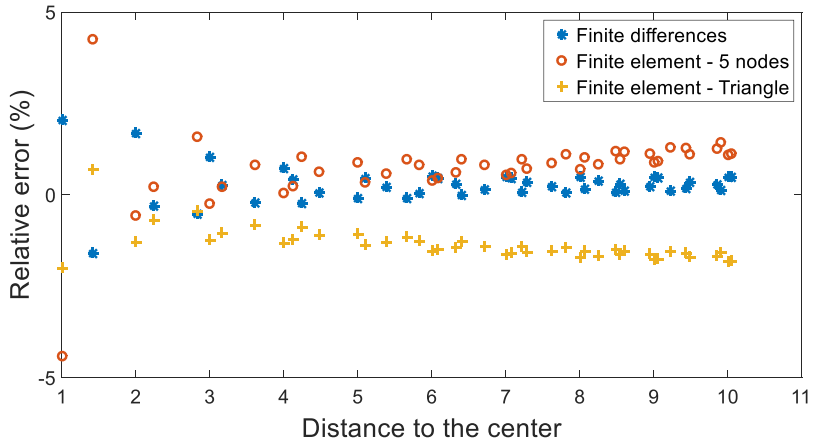


Fig. 13 Relative error obtained between the different versions of the model and the analytic solution for the steady state concentration on nodes according to their distance toward the source for the Lattice 11 (refined near the source). L=100, D=10 , d=0.1 and ϕ_0=0.628.

Table 3 Comparison of the computation time for different models and different lattices. It corresponds to a steady state analysis performed with Spectre on a 12-core 2.6-GHz CPU with 40 Go of RAM. Figure given in parenthesis corresponds to the computation time increase (in %) between the model and the finite difference model

Lattice number	8	4	7	6
Number of nodes	3 997	6 561	10 201	36 991
Computation time (finite differences)	3.15	4.27	6.07	55.5
Computation time (finite elements / 4-nodes model)		4.53 (+5.1%)	7.16 (+18%)	
Computation time (finite elements / 5-nodes model)	3.31 (+5.1%)	5.13 (+5.1%)	7.84 (+29%)	109 (+96%)
Computation time (finite elements / triangular model)	3.09 (-2.0%)	4.93 (+5.1%)	7.88 (+30%)	92 (+66%)

2.2. Different boundary conditions

We use the lattice 2 with a source in the same position (50, 50) to test the different border conditions. The 3 conditions to be tested are blocked diffusion where no additional treatment is performed in border nodes (no additional resistors are implemented), fixed condition where all border nodes are fixed to a given concentration (we chose to fix all border nodes to C=0) and further diffusion where additional border resistors are computed for all border nodes. The additional border resistors values are scaled on the diffusion constant D. The results can be seen on Fig. 14.

We first notice that all the curves follow the same trend. The only difference between the different border conditions is an offset. As expected, the lower curve corresponds to the simulation where some of the nodes were fixed to 0. This has the effect to drain all the others. Moreover, the higher curve corresponds to contained diffusion. Indeed, with no additional border resistors it is expected that the overall steady-state concentration level of the lattice would be higher than in the simulation with additional border resistors (further diffusion).

2.3. Comparison between the triangular model and the 4- and 5-node models

Simulation results on lattice #1 also show that the triangle model is actually more “smooth” than the 4- and 5-node models. This is particularly observable on the high-gradient part of the curve (the nodes closest to the center). On close-up (see Fig. 15), we see that the triangle model is less rugged than the two other models and closest to the expected curve (not plotted here).

On a regular lattice, the triangle model is therefore more accurate than the two previously mentioned models.

2.4. Interface between two zones of different refinement level n

Lattices #6 and #7 gave results that showed the same glitch: on the regular parts of the lattice the model performs well, as expected, but at each frontier between meshes of different refinement levels a “bump” appears. This phenomenon is best observed on lattices with fewer nodes (shown on the right panel on Fig. 16), this is why we will show here results obtained on elementary lattices of 100x100 with 4 initial divisions.

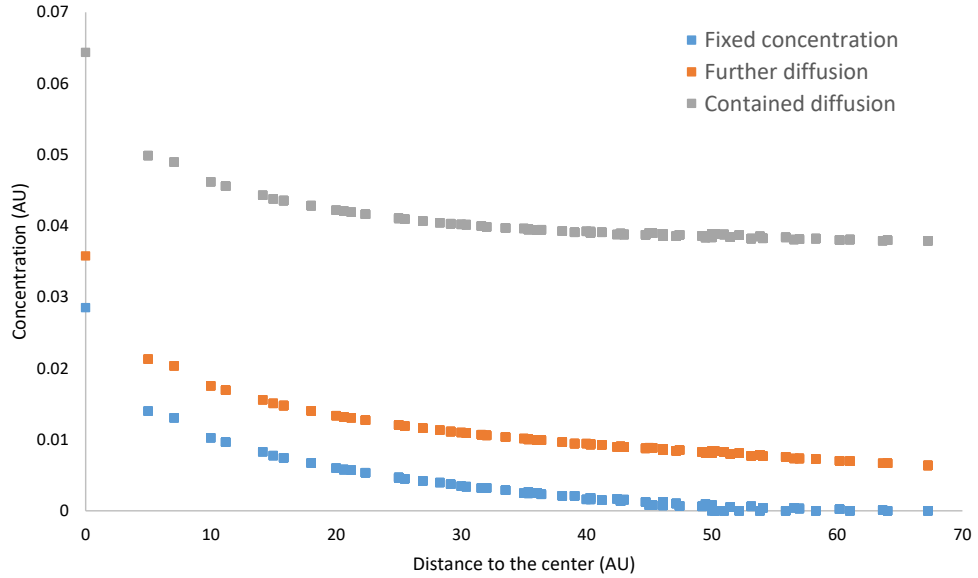


Fig. 14 Comparison of the 3 different border conditions on lattice 2. In fixed concentration, all border nodes are fixed to C=0. In further diffusion, additional border resistors are implemented following the matrix found previously. In contained diffusion, no additional border resistance is implemented.

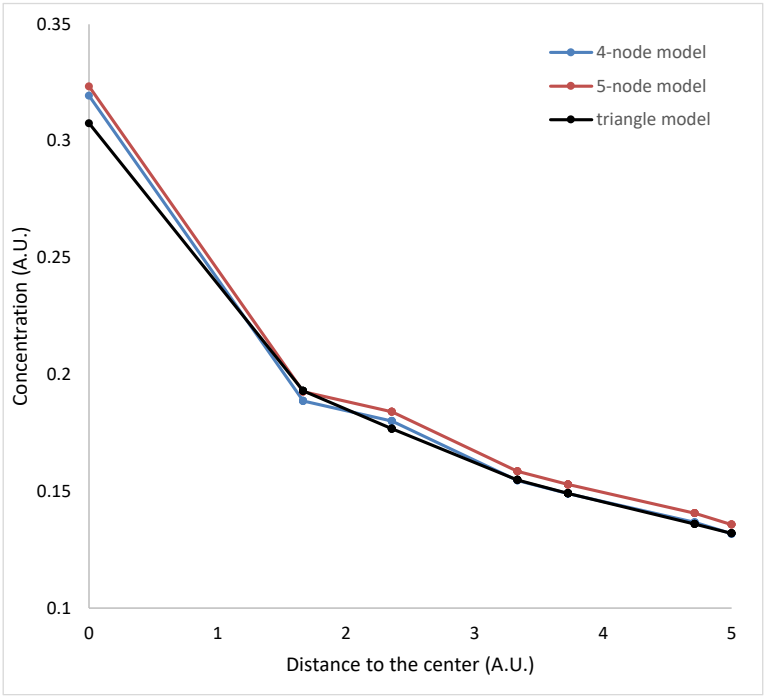


Fig. 15 Comparison of 3 models: 4-node, 5-node and triangle models. Close-up of the concentration of the nodes closest to the center, where the curve is less rugged for the triangle model than for the 4-node and 5-node models.

Lattice alpha has a circular central refinement zone of n=1 and of radius 25. Lattice beta is lattice alpha with an additional circular central refinement zone of n=2 and of radius 12.5.

The results show an unexpected increase in the concentration as a function of the distance to the source. These increases are located on the segment nodes (see the red and purple circles on Fig. 16).

This problem was not solved during the thesis hence the triangle model was only fit for regular lattices.

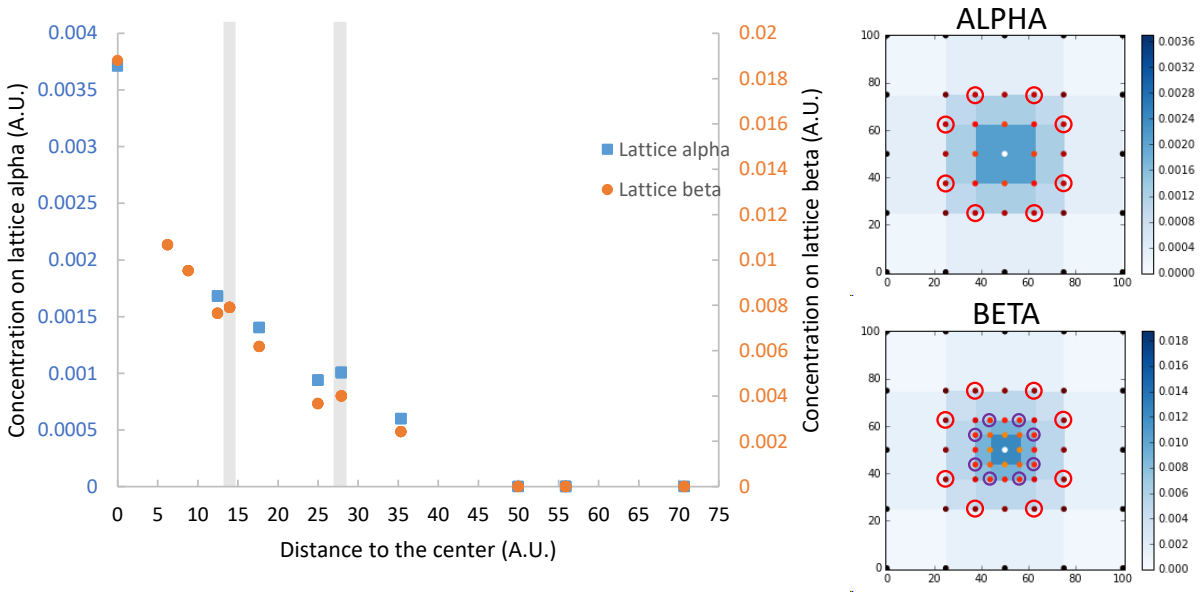


Fig. 16 Simulation results of the triangle model for the lattices alpha and beta. Unexpected increases (grey rectangles on the graph) are observed at segment nodes (circled on the 2D representation of the lattices) located at the border between meshes of different refinement levels n. In red, the border between n=0 and n=1 and in purple the border between n=1 and n=2.

2.5. Conclusion and outlook

We tried to establish a finite element version of our finite differences model. When restraining the lattices to only 4- or 5-node elements, we found a model that functions appropriately, the 5-node model. However, with more than one refinement zone, it often becomes necessary to model elements with more nodes. As the initial model allowed the simulation of 8-node elements, we tried to find an 8-node element with the finite element method. It appeared clear that an additional internal node was needed, which lead to the division of the square into triangles. This new model performs better on regular lattices but is not mature for irregular lattices involving elements with more than 5 nodes. In the following sections of the simulation tool, the results are therefore presented using the finite difference model.

The next step is obviously to fix the segment node problem observed on the triangle model. Moreover, the current triangle model is only applicable to square elements. An important generalization to rectangular elements should be envisioned.

3. Summary on both models

The finite difference model is correct for each configuration of the lattice. The finite element model is more accurate than its counterpart as the computing of the flux in a node are dependent on all its neighbors. However, this version of the model is only stable for a restrained pool of configurations. Therefore, we present the following results with the initial version of the model which corresponds to a finite difference model. All the results have been generated with the Spectre simulator.

4. Results on biological use cases

Now that we have validated the tool on from a theoretical point of view, we will show its pertinence over several use cases. We will use our tool to demonstrate what type of results it can provide in synthetic biology experiments and problems. Beforehand, two issues must be addressed: the addition of external components (cell, localized reaction, membrane walls ...) and the description of multi-layer lattices in the case if a system composed of multiple diffusions species.

For the first point, descriptions of biological models in SPICE or in Verilog-A are required. A SPICE model can be generated from a SBML description or from a proprietary input file (which is composed of a list of parameters, a list of reactions and a list of reaction with predefined rate equation) with BB-SPICE (Madec, Lallement, and Haiech 2017). Moreover, writing from scratch a Verilog-A model associated with a biological mechanism is not so challenging. This is done with the help of the analogy between electronic and biology described in Chapter 2. The writing of a biological model in VHDL language has also been demonstrated in (Gendrault et al. 2014). The transposition of the methodology to Verilog-A is pretty straightforward.

Second, the description of multi-layered lattices has already been taken into account in the netlist generator (see the description of the tool, in Chapter 6). The current version of the generator duplicates the basic lattice as many times as necessary by creating new nodes whose numbers are incremented by the total number of nodes in one lattice. Thus, the nodes of each layer are unique and the lattices are independent from each other. By this way, they are superimposable, which facilitates their interconnection via external component models.

4.1. Band-pass system

The first system we study is the band-pass system described in Chapter 5 (Basu et al. 2005). This system involves two populations of cells, placed at different positions, and allow the creation of a variety of patterns. In this system, the spatiotemporal behavior of AHL is decisive in the obtained pattern (the receiver cells that express GFP). We therefore want to simulate the diffusion of AHL and the response of the receiver cells. We first describe the system and how we model it and then present our simulation results.

4.1.1. Description of the system

The Basu system allows the formation of GFP (Green Fluorescent Protein) patterns in a closed space. Two populations of cells compose this system: the senders and the receivers. The senders communicate with the receivers by synthesizing and isotropically emitting a small molecule able to diffuse and enter cells: acyl-homoserine lactone (AHL). The senders emit AHL when aTc (anhydrotetracycline) is present. In the results however, we will consider that AHL is always emitted by the sender cells. The system is described in more details in Chapter 5. In brief, GFP is only produced by the cells receiving an intermediate concentration of AHL. This translates in 2D by induction of the receiver cells located at medium range from the sender cells.

4.1.2. Modeling

Our model is composed of a lattice with two layers: one layer for the AHL (red layer on Fig. 17) and the other for the GFP (blue layer on Fig. 17). On the AHL layer, we dispose punctual AHL sources on various nodes. AHL diffuses on this layer. Each node of the GFP layer is connected to a node of the AHL

layer by the means of a component modeling the band-pass system. On this layer, diffusion is set to 0 because GFP does not diffuse: it stays inside the receiver cell.

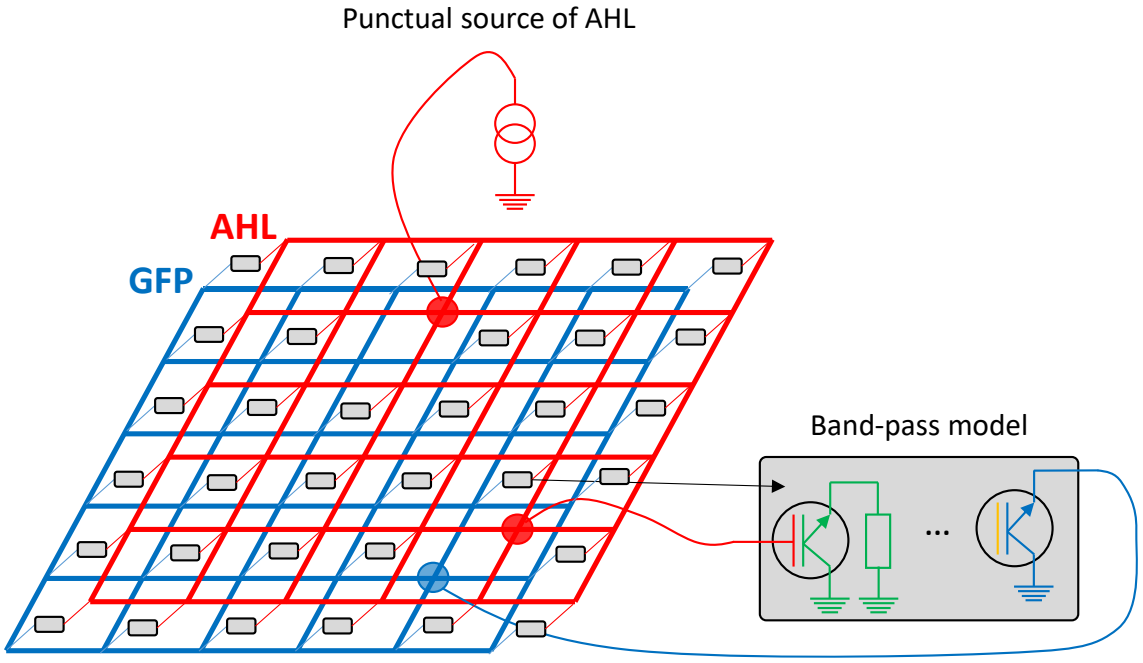


Fig. 17 Schematic representation of the 2D model used to simulate the band-pass behavior in space and time. The band-pass model grey square represents the Verilog-A model of the band-pass.

4.1.3. Simulation results

Two configurations are tested: one with only one group of sender cells in the middle and another one with 3 groups of sender cells (Fig. 18). Corresponding experimental results can be seen in (Basu et al. 2005). In both configurations, we use as a base lattice a square of 100x100 units with 40 divisions per axis. In the first configuration, we added one circular refinement zone with a radius of 25 centered at the sender cell ($x = 50$ and $y = 50$), with a refinement coefficient of 1 (each square in this zone is divided once into 4 sub-squares). The concentration of AHL and GFP according to space at the steady-state can be observed in Fig. 18A and B, with the sender cell represented as a red dot. Transient evolution of AHL and GFP concentrations are monitored at three nodes (Fig. 18E), located on the diagonal from the center towards the lower left corner (red dots). They are represented in Fig. 18A and B. As AHL diffuses from its centered source, the peak of GFP propagates from the center toward the borders of the lattice, as it first appears at node (40,40) in black and then at node (37.5, 37.5), where it stops as expected. In the second configuration, we added three circular refinement zones with a radius of 35 centered at the sender cells (namely coordinates (50,35), (35,60) and (65,60)), with a refinement coefficient of 2 (each square is divided twice). Steady-state spatial map of AHL and GFP for this configuration are also given in Fig. 18D and E. The simulation results concur with the results provided by Basu in (Basu et al. 2005). Indeed, our simulation successfully obtains a ring of fluorescent cells around the emitting node (Fig. 18B), or a triangular ring around the 3 emitting cells in the second case (Fig. 18E).

4.1.4. Conclusion

In conclusion, the tool can be used to accurately reproduce experimental results in a low computation time. This means that one could use our tool to predict the pattern given by any other disposition of the sender cells, thus sparing experimentation time.

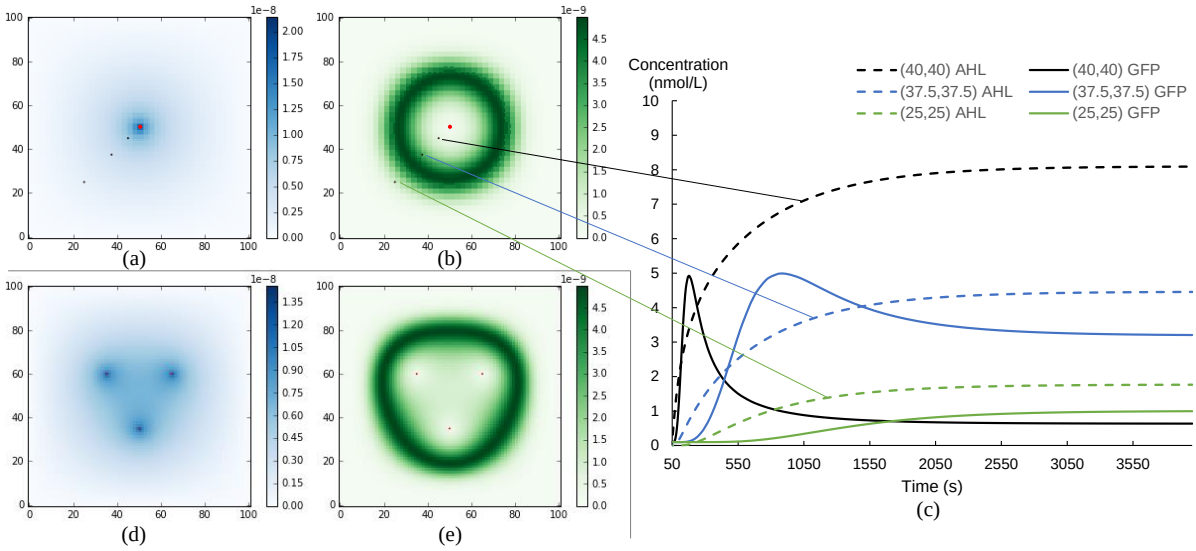


Fig. 18. Simulation of the band-pass system with one (A and B) or three (D and E) sender cells groups. A and D resp. B and E) represent the concentration of AHL (resp. GFP) per mesh. The sources of AHL are represented as red dots. Blue dots (in A and B) show the nodes of which the transient AHL (dotted line) and GFP (full line) concentrations are shown on C.

4.2. XOR

A XOR gate is a 2-input logic gate which is on when only one of its inputs is on (see the table on Fig. 19 B). Many biological XOR gates can be found in the literature (Ausländer et al. 2012; Terzer et al. 2007; Tamsir, Tabor, and Voigt 2011). We will focus on the system by Tamsir et al., a system splitted on 4 cells. We use our tool to study the influence of the position of these different cells on the correct functioning of the XOR gate.

4.2.1 Presentation of the XOR system

In microelectronics a XOR gate can be obtained with a large variety of gates combinations. Tamsir et al. decided to use 3 NOR gates and an implicit OR gate modeled by a buffer. Contrarily to microelectronics where the signals are confined and directed through wires, inside a cell the molecules diffuse and can reach any other component of a GRN. To avoid such a crosstalk, a solution is to split the system into different cells (strains). Tamsir et al. opted for this solution and cloistered each NOR gate into a different strain. To “wire” the NORs, they use the common AHL cell-cell communication system encountered previously in this thesis (e.g. the band-pass system described hereabove). These AHL molecules are the only ones able to enter and exit the cells. “Wiring” between cell is made possible by the variety of AHL molecule and the high selectivity of each promoter towards its specific AHL.

The NOR gate used by this American team is composed of a repressor that inhibits the expression of the output (Fig. 19 A). Two promoters allow the activation of the repressor’s expression by two activators, the inputs. When at least one of them is present, the repressor is expressed. Hence it is only when no activator is present that the output can be produced.

The inputs of this XOR gate are arabinose (Ara) and anhydrotetracycline (aTc). The first NOR gate takes as inputs Ara and aTc and produces 3OC12-HSL (N-3-oxo-dodecanoyl-homoserine lactone), an AHL synthesized by LasI. The two other NOR gates are sensitive to 3OC12-HSL22 and respectively to Ara for Cell 2 and aTc for Cell 3 (see on Fig. 19). These two NOR gates produce another AHL named C4-HSL (N-butyryl-homoserine lactone). For an easier monitoring of the output of the XOR gate, Tamsir et al. use a fourth strain that acts as a buffer gate. This buffer gate produces YFP, a fluorescent protein, when

receiving C4-HSL. *In silico*, it is possible to monitor directly the concentration of every molecule (e.g. C4-HSL), which is not the case in vivo. Thus, for simplicity sake, this 4th gate is not modeled. The output is monitored by adding the signals emitted by Cell 2 and Cell 3 (hence the implicit OR gate).

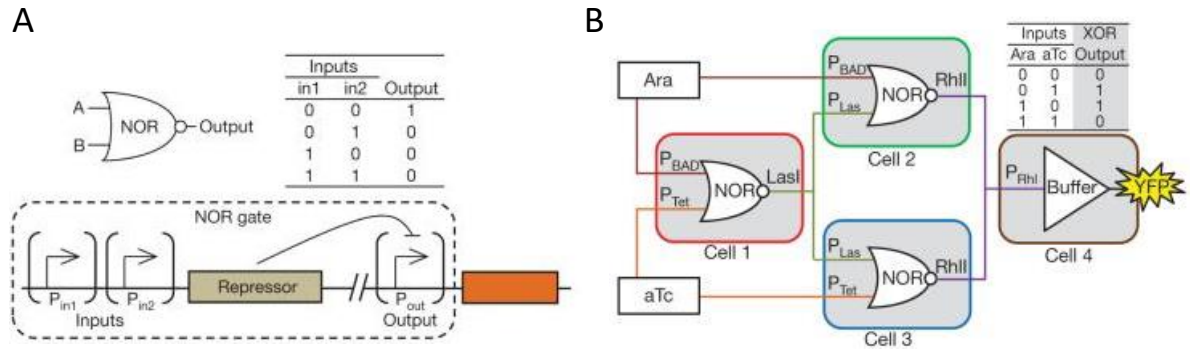


Fig. 19 A biological XOR gate composed of 3 NOR gates and a buffer. A: generic NOR gate. B: "wired" NOR gates using the AHL cell-cell communication system with a buffer that outputs YFP, a reporter protein (Tamsir, Tabor, and Voigt 2011).

The behavior of such a gate is that when no input (Ara and aTc) is present, the first NOR is high (its output is present). Because only one of their input is high, the two other NORare low (their output is absent). Hence the output of the XOR gate (YFP) is low. When the two XOR inputs are present, the first NOR is low. Similarly, the two other NORare low and YFP is low. When Ara only is present, the first NOR is low. As a consequence, the NOR of Cell 3 has both of its inputs low, meaning that its output is high. Hence YFP is high. By symmetry, when aTc only is high, YFP is also high.

We have seen in the previous section that AHL degrades over time so that the concentration of an AHL at one point in space depends on the distance between this AHL source(s) and this point. We want to see whether the positioning of these AHL sources, the XOR gates, has an influence on the correct functioning of the XOR gate.

4.2.2. Modeling

In the following, the length and surface units are arbitrary and designed respectively as u and u². By default, the input sources are set to 1 µM/u²/s.

A Verilog-A model is written for a cell that performs a generic NOR gate. The module is composed of two input nodes and one output node. In addition, internal nodes are created inside the Verilog-A model and simulate the same 3 molecular species (inputs and output) inside the cell. A parameter called membrane permeability η is designed to model the diffusion across the plasma membrane of the cell. This ratio is by default set to 0.5. The molecules have a degradation rate inside the cell that can be different than outside the cell. In our case, the outside degradation rate is 0.01 s^{-1} and the inside degradation rate d_{int} is 0.1 s^{-1} by default. The NOR gates are described by the following equations:

$$\frac{d[Rep]}{dt} = \beta \cdot \frac{\left(\frac{[In_0]}{KA_0}\right)^{nA_0} + \left(\frac{[In_1]}{KA_1}\right)^{nA_1}}{1 + \left(\frac{[In_0]}{KA_0}\right)^{nA_0} + \left(\frac{[In_1]}{KA_1}\right)^{nA_1}} - \alpha \cdot [Rep]$$

$$\frac{d[mRNA]}{dt} = K_{TR} \cdot \frac{1}{1 + \left(\frac{[Rep_0]}{KR}\right)^{nR}} - d_{mRNA} \cdot [mRNA]$$

$$\frac{d[Out]}{dt} = K_{TL} \cdot [mRNA] - d_{int} \cdot [Out]$$

$[In_0]$, $[In_1]$ and $[Out]$ are respectively the internal concentration of both inputs and the output molecule inside the cell. This concentration inside and outside are balanced by a flux computed depending on the value of η and which is connected between the internal node at one end and the lattice node at the other end.

$$[X]_{int} = [X]_{ext} \cdot \frac{1-\eta}{\eta} \cdot \frac{1}{d_{int}}$$

Table 4 shows the meaning of the parameters and their default value.

In this problem, 4 different molecules diffuse: Ara, aTc, 3OC12-HSL and C4-HSL. Thus, a four-layer regular lattice of size 100x100 u^2 with 100 division per axis is generated. The nodes of each NOR gate are connected to the layer corresponding to the molecule they input/output. All the nodes of a NOR are located on the same position. Moreover, for the two inputs of the system Ara and aTc, a constant source positioned at specific spots in space is instantiated.

Table 4 NOR parameters and default values

Parameter	Description	Default value
$\boldsymbol{\beta}$	Repressor synthesis rate	100 µM.s^{-1}
$\boldsymbol{\alpha}$	Repressor degradation rate	10 s^{-1}
$\boldsymbol{KA_{0,1}}$	Dissociation rate of input i	1e-3 µM
$\boldsymbol{nA_{0,1}}$	Hill number of input i	2
$\boldsymbol{d_{mRNA}}$	Output mRNA degradation rate	10 s^{-1}
$\boldsymbol{K_{TR}}$	Output transcription rate	10 µM.s^{-1}
$\boldsymbol{K_{TL}}$	Output translation rate	100 µM.s^{-1}
$\boldsymbol{n_R}$	Hill number of the repressor	2
$\boldsymbol{K_R}$	Dissociation rate of the repressor	1e-4 µM
$\boldsymbol{d_{int}}$	Internal degradation rate	1.0e-1 s^{-1}
$\boldsymbol{d_{ext}}$	External degradation rate	2.0e-1 s^{-1}
$\boldsymbol{\eta}$	Membrane permeability	0.5
$\boldsymbol{D}$	Diffusion constant	1 m^2.s-1

4.2.3. Results

We hereafter name NORa, NORb and NORc the NOR gates that correspond respectively to Cell 1, Cell 2 and Cell 3. As we do not model the buffer of Cell 4, we name GFP the output of NOR 2 and NOR 3. GFP is the output of our XOR gate. It should be mentioned that in our model, GFP diffuses.

We simulate the system for 10000s divided into 4 periods: during the first period, no input is present, during the second period only Ara is present, during the third period only aTc is present and during the fourth both inputs are present. We first want to simulate the XOR gate with a disposition similar to the published results. The sources are coherently placed (Ara source between NOR 1 and NOR 2 and aTc source between NOR 1 and NOR 3) as shown on Fig. 20A.

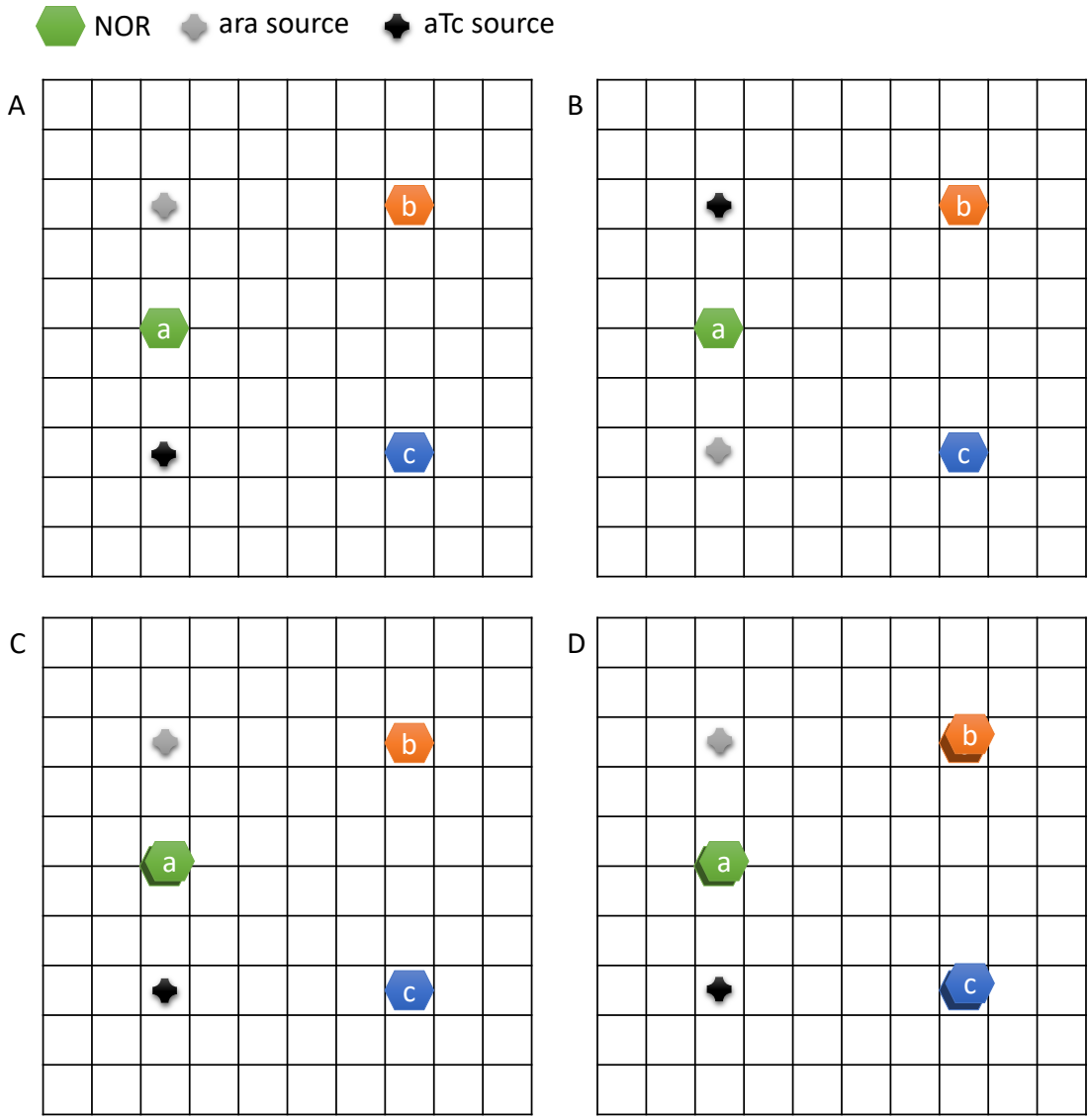


Fig. 20 Disposition of the NOR gates (hexagones) and the input sources. Superimposition of hexagons means that two NORs of the same kind are connected to the same node.

We observe the concentration of each gate's inputs and output (Fig. 21). 3OC12-HSL is designated as AHL. NORa shows a very clear behavior of a NOR gate with sharp transitions. NORb and NORc show also a correct behavior except for a first spike of output observed at the very beginning of the simulation (before 500 s). This peak of GFP is due to the delay caused by AHL diffusion and degradation. The global XOR behavior cannot be observed at NORb nor at NORc but by combining the output of both NORs a XOR behavior is indeed obtained. NORc is high only when Ara alone is present and NORb is high only when aTc alone is present. This is observable at spots located between all three NORs (data not shown).

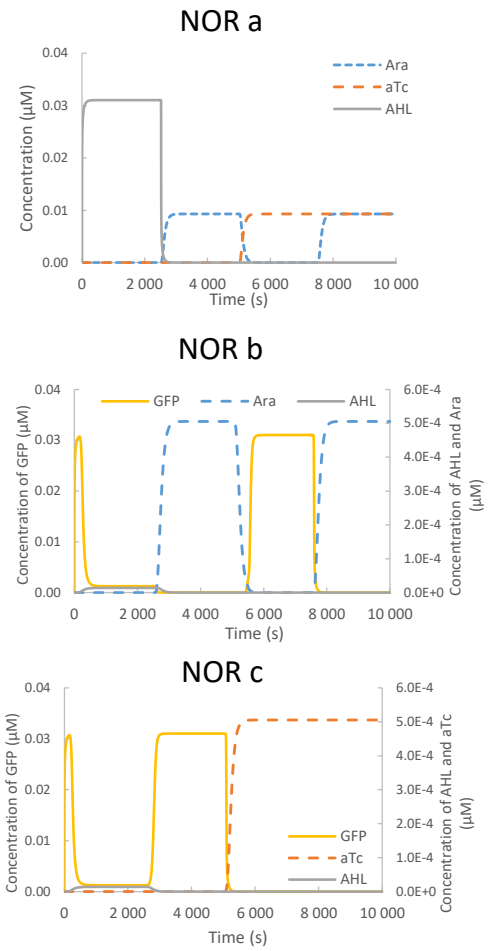


Fig. 21 Dynamic simulation results of a XOR gate. Input and output concentrations are shown at each NOR gate.

We tried to obtain even sharper edges, i.e. faster transitions between two different states, in order to reduce the first undesirable peak of GFP. To that end, we increased NORa Ktr to 100µM.s^{-1}. The results showed a reduced GFP first peak in width as it was less than 200 s large. Decreasing extracellular degradation also reduces this initial peak but breaks the XOR behavior (GFP is almost always low). The opposite also applies as a higher external degradation rate leads to a constant high GFP signal.

We then tried to invert the positions of the aTc and Ara sources (see disposition B on Fig. 20). The XOR behavior was only achieved for increased fluxes of aTc and Ara. Indeed, the molecules have a higher distance to travel before reaching their target cell.

We then tweaked the number of NORs. First, we position 2 NORa on the same node (see disposition C on Fig. 20). As expected, this results in a doubled AHL production. A XOR behavior can be observed. Dividing NORa Ktr by 2 restores the initial AHL production. We then doubled every NOR with each a Ktr divided by two (compared to the default value), meaning that two NORs are connected to the same node. When compared to the situation where only NORa is doubled the results are quasi identical. A minor decrease (less than 5%) in Ara, aTc and GFP concentration is observed at NORb and NORc. Ara and aTc are both pumped by an additional NOR each, explaining the decrease. Moreover, with more NORs connected to the same node, the GFP is also more degraded, explaining the slight decrease observed.

Finally, we tried to position each NOR randomly. Each source is still at the same spot as previously. None of the 4 trials we made gave satisfying results. We then tried to place 10 of each NOR randomly, with a Ktr divided by 10, and 100 of each with a Ktr divided by 100 (compared to the default value). In both cases, none of the 4 trials produced a XOR behavior. To limit the scattering of the NOR of each type, we clustered them. The 100 NORs of a type are randomly positioned inside a square of 10x10. The position of the square is random and different for each type of NOR. Again, no XOR behavior could be observed, even when placing only 10 NORs of each type.

4.2.4. Conclusion

The results showed that the position of the NORs have to be chosen carefully. The external degradation rate, influenced by the nature of the external medium, is also critical. Simulation results revealed a first undesirable peak of output of NORb and NORc gates due to delays introduced by the diffusion. The results showed that the size and the amplitude of this peak can notably be controlled by the external degradation rate. These peaks are typically the short-lived phenomena that are responsible for malfunctions in some sequential circuits (see Chapter 4). The fact that the spatio-temporal simulator highlights these glitches is important in the perspective of using this tool for the study of sequential systems split in multiple cells.

As systems grow in size and complexity, and become also more standardized, the same component have to be reused in the same system. Consequently, many systems also have to be splitted on different cells to avoid crosstalk. With different strains of cells comes the question of the positioning of these strains and we showed here that our tool can help for this topic. Our tool also allowed an easy tinkering of the different parameters, making the search for a solution of a design problem (e.g. an undesirable peak of output) easy, quick and cheaper than to redo an experiment.

Ideally, an even greater gain of time could be achieved by using an evolutionary algorithm (see Chapter 6) to optimize the position of each cell.

4.3. A prey-predator-like system

Up until now, we have studied interacting systems of cells with only unidirectional interaction. Cell-cell communication systems described above (like the AHL system) are also used for bidirectional interactions. Synthetic biologists have been studying many bidirectional systems (Brenner et al. 2007; Shou, Ram, and Vilar 2007; Balagadde et al. 2008). In this paragraph, we want to focus on the simplified oscillatory prey-predator system.

Prey-predator ecosystems are composed of two populations whose activity depends on each other's (*e.g.* rabbits and foxes). Recently, artificial prey-predator ecosystems have been demonstrated with artificially reprogrammed cell (Fujii and Rondelez 2013; Balagaddé et al. 2008). In existing models, spatiality, which may play an important role in the process, is not taken into account. In this work, focus is put on the diffusion process. Thus, the model of the prey and the predator will be adapted to retain the core oscillatory module only.

4.3.1. Presentation of the system

The modeled system is composed of two cells. A gene in the cell A, which mimics the prey, synthesizes a regulator X for a gene inside the cell B, which mimics the predator. This gene synthesizes in turn an inhibitor Y for the gene in the cell A. By this way, when the concentration of X increase, it promotes

the proliferation of Y which in turn reduces the concentration of X. Under appropriate condition, such system should oscillate.

4.3.2. Model

Again, we use as a base lattice a square of 100x100 units with 4 divisions per axis. The positions of cells A and B are respectively (25; 75) and (75; 25). Two circular refinement zones with a radius of 10 and a degree of refinement of 4 are built around each cell. The obtained mesh is represented in Fig. 22A. Two independent diffusion layers are defined, one for the activator molecule and one for the inhibitor. Both layer are independent. They are connected on nodes (25; 75) and (75; 25) to two external model, one for the cell A and one for the cell B. Models used in cell A and B correspond to simple regulation using Hill's equations (Konkoli 2011). Thus, in cell A, the transcription of DNA and the translation of mRNA into the activator are given by:

$$\frac{d[mRNA_A]}{dt} = K_{TR} \cdot \frac{[Inh]^{n_R}}{{K_R}^{n_R} + [Inh]^{n_R}} - d \cdot [mRNA_A]$$

$$\frac{d[Act]}{dt} = K_{TL} \cdot [mRNA_A] - d \cdot [Act]$$

By the same way, in cell B, the transcription of DNA and the translation of mRNA into the repressor are given by:

$$\frac{d[mRNA_B]}{dt} = K_{TR} \cdot \frac{{K_A}^{n_A}}{{K_A}^{n_A} + [Inh]^{n_A}} - d \cdot [mRNA_B]$$

$$\frac{d[Inh]}{dt} = K_{TL} \cdot [mRNA_B] - d \cdot [Inh]$$

Again, to simplify the model, we consider that cells membrane are permeable to both the activator and the inhibitor, so that the concentrations inside and outside are equal.

4.3.3. Results

As expected from (Balagaddé et al. 2008), three different dynamics can be found depending on the parameters of the system as well as the distance between cells: coexistence, extinction or oscillations. Oscillatory behavior is represented in Fig. 22B and Fig. 22C.

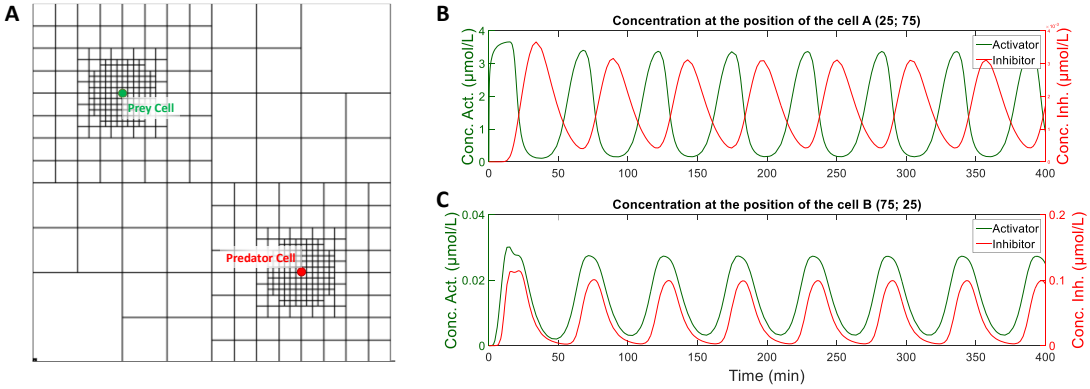


Fig. 22 On the left: mesh generated for the simulation of the simplified prey-predator system. On the right: simulation results. Transient evolution of the concentration of activator and inhibitor at the position of cells A and B. Parameters used for the simulation are the following: D=2 m2.s-1, d=10^{-6} min^{-1} for the diffusion, K_{TR}=10^{-3} µmol.L^{-1}.min^{-1}, K_{TL}=10^{-2} min^{-1} for transcription rate and translation rate inside the cells, K_R=10^{-6} mol.L^{-1} and n_R=3 for Hill's equation used to model the transcription in the cell A and K_A=10^{-4} mol.L^{-1} and n_R=3 for Hill's equation used to model the transcription in the cell B.

4.3.4. Conclusion

The model can now be used to study the impact on system parameters on the characteristics of the oscillations (amplitude, frequency). In particular, the model can be used to tune the relative position of the cell and/or the properties of the diffusion medium in order to match oscillation characteristics defined a priori. This tuning is not straightforward and is greatly enhanced by simulation tools. Again, this task can be done with the evolutionary algorithms described in Chapter 5.

4.4. Synchronized Oscillators

The aim of this application is to address the second main question related to the distribution of a synchronous clock to sequential system divided in multiple cells. Literature shows examples of synchronized oscillators (Zhou et al. 2008; Russo and Bernardo 2009; Wang and Chen 2005; Wagemakers et al. 2006; Garcia-Ojalvo, Elowitz, and Strogatz 2004). In this part, we use our simulator to address this problematic.

4.4.1. Description of the repressilator

A clock is an oscillatory signal. With GRNs, a simple negative feedback can give rise to oscillatory behaviors. The best candidate to generate a clock signal is probably the repressilator, designed by Elowitz and Leibler in 2000 (Elowitz and Leibler 2000). The system is composed of 3 genes and 3 interdependent repressors: the gene expressing CI is repressed by TetR, the gene expressing LacI is repressed by CI and the gene expressing TetR is repressed by LacI (cartoon on the left on Fig. 23). This system is oscillatory as the presence of a repressor induces its own inhibition. For example, if CI is present (high), LacI is absent (low). Because LacI is low, TetR is not repressed and is therefore high. However, because TetR is high, CI is repressed and becomes low. As CI falls, LacI is not repressed anymore and rises, inducing the repression of TetR. With this repression, CI is again high and the cycle goes on. Noteworthy is that these transitions from low to high and high to low (due to the degradation of the repressor) are not instantaneous: they produce a delay in the system. Because of this delay we observe the oscillations presented on Fig. 23, where each repressor has a peak while the two others are falling for one and rising for the other. In this section, we want to synchronize a population of repressilator units to obtain a space-synchronized biological clock.

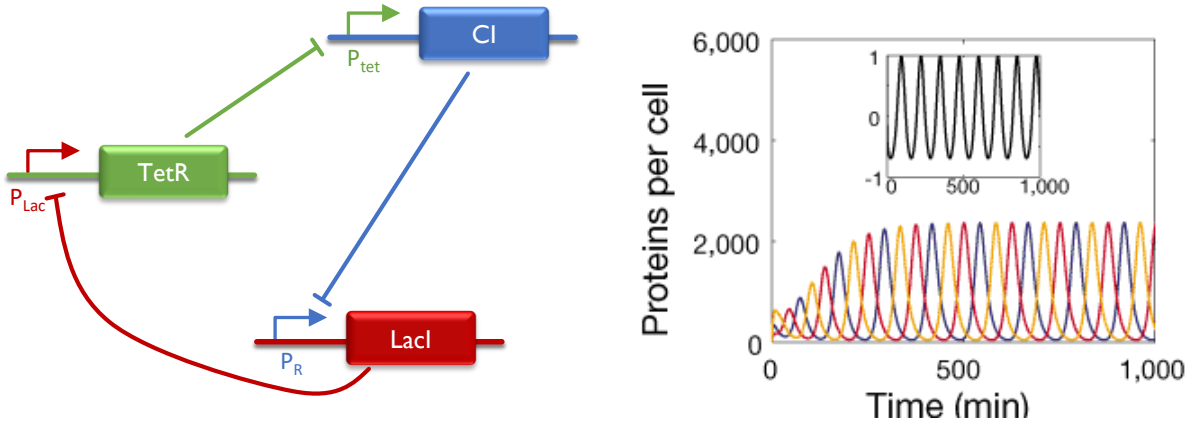


Fig. 23 Repressilator system with its repressors and associated promoters and simulation results (ELOWITZ AND LEIBLER 2000)**: in red LacI, in yellow tetR and in blue CI.**

4.4.2. Model

In order for different repressilators to synchronize, they need to be able to communicate. To that end, we introduce a gene coding for AHL, a molecule able to cross the cellular membrane and diffuse across

space. We choose to introduce AHL in the system with the following behavior: AHL production is inhibited by LacI and AHL activates TetR production. By this way, the repressilator is sensitive to an incoming AHL signal and also emits its own AHL signal, understandable by other repressilators. A single-layer lattice is generated and each repressilator is connected to this lattice via the AHL terminal.

For LacI, TetR and CI, mRNA is modeled whereas for AHL, a single equation is used to model AHL production (the intermediates, namely mRNA and LuxI, are skipped for a simpler system). The equations are as follows:

$$\frac{d[mRNA_{CI}]}{dt} = K_{TR} \cdot \frac{{K_{RTet}}^{n_{RT}}}{{K_{RTet}}^{n_{RT}} + [TetR]^{n_{RT}}} - d_{mRNA} \cdot [mRNA_{CI}]$$

$$\frac{d[CI]}{dt} = K_{TL} \cdot [mRNA_{CI}] - d_{Prot} \cdot [CI]$$

$$\frac{d[mRNA_{LacI}]}{dt} = K_{TR} \cdot \frac{{K_{RCI}}^{n_{RC}}}{{K_{RCI}}^{n_{RC}} + [CI]^{n_{RC}}} - d_{mRNA} \cdot [mRNA_{LacI}]$$

$$\frac{d[LacI]}{dt} = K_{TL} \cdot [mRNA_{LacI}] - d_{Prot} \cdot [LacI]$$

$$\frac{d[mRNA_{TetR}]}{dt} = K_{TR} \cdot \frac{{K_{RLacI}}^{n_{RL}}}{{K_{RLacI}}^{n_{RL}} + [LacI]^{n_{RL}}} \cdot \frac{[AHL]^{n_A}}{{K_A}^{n_A} + [AHL]^{n_A}} - d_{mRNA} \cdot [mRNA_{TetR}]$$

$$\frac{d[TetR]}{dt} = K_{TL} \cdot [mRNA_{TetR}] - d_{Prot} \cdot [TetR]$$

$$\frac{d[AHL]}{dt} = \beta \cdot \frac{{K_{RLacI}}^{n_{RL}}}{{K_{RLacI}}^{n_{RL}} + [LacI]^{n_{RL}}}$$

with K_{TR} the transcription rate, K_{TL} the translation rate, d_{mRNA} and d_{Prot} respectively the mRNA and the protein degradation rate, n_A, n_{RL}, n_{RC} and n_{RT} the Hill coefficient of respectively AHL, LacI, CI and TetR, K_{RX} the dissociation coefficient of repressor X, K_A the dissociation coefficient of AHL and β the synthesis rate of AHL. In this model, AHL degradation comes only from the lattice. Default parameters are given in Table 5.

Table 5 Default parameters of our repressilator model and of the diffusion model

	Parameter	Default value
Biological model	β	1e-2 µM.s^{-1}
	K_A	1e-2 µM
	n_A	2
	d_{mRNA}	10 s^{-1}
	d_{Prot}	10 s^{-1}
	K_{TR}	10 µM.s^{-1}
	K_{TL}	50 µM.s^{-1}
	n_R	2
	K_R	1e-2 µM
Diffusion model	D_0	1
	R_0	1e3

4.4.3. Results

The lattice used is a regular 100x100 lattice with 100 divisions per axis.

Single repressilator

First, we simply simulate our model without diffusion, to see if our modification impairs the oscillatory behavior of the repressilator. Simulation results show that the introduction of AHL in the system does not affect the possibility to obtain sustained oscillations (Fig. 24). We note that the AHL concentration is identical to the CI concentration.

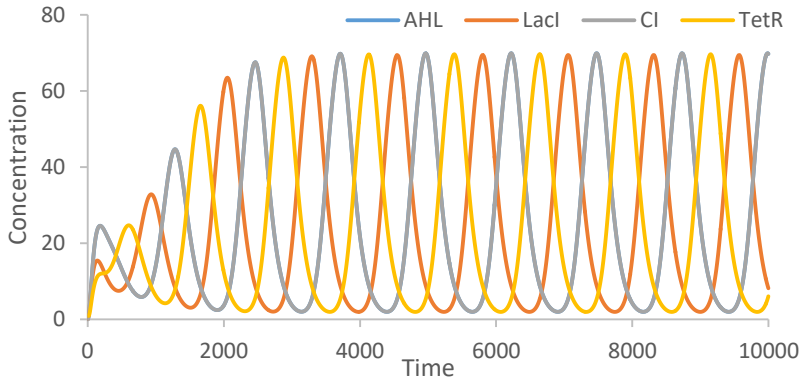


Fig. 24 Simulation of the repressilator.

We wanted to assess the effect of modifications of the lattice degradation rate (named *deg* on Fig. 25). A single repressilator is positioned at the center of the mesh (coordinates (50, 50) with an initial concentration of 0.01 µM for the node).

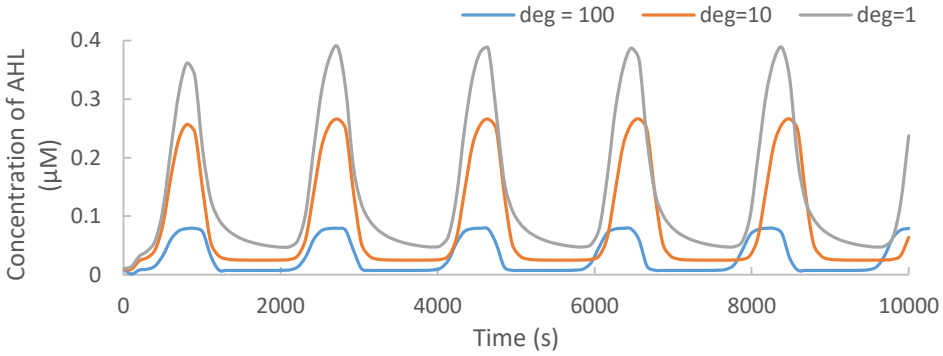


Fig. 25 Influence of the degradation rate of AHL (in s^{-1}) over the oscillations of the repressilator

As expected, the oscillations amplitude and the length of the transitions from high to low state increase when degradation decreases. We also notice a modification in the period length of the oscillations but not with a monotone trend. Indeed, the period is the highest with a degradation rate of 10 s^{-1} (period = 1872 s) whereas it is lower for a degradation rate of 1 s^{-1} and 100 s^{-1} (respectively period = 1872 s and 1838 s).

Multiple repressilators

Two repressilators

Two repressilator units are disposed at the following coordinates: repressilator 1 at (25,25) and repressilator 2 at (75,75). We monitor the concentration of AHL on the medium at these two

coordinates and also at two additional points at coordinates (40,60) and (50,50). The disposition is shown on Fig. 26.

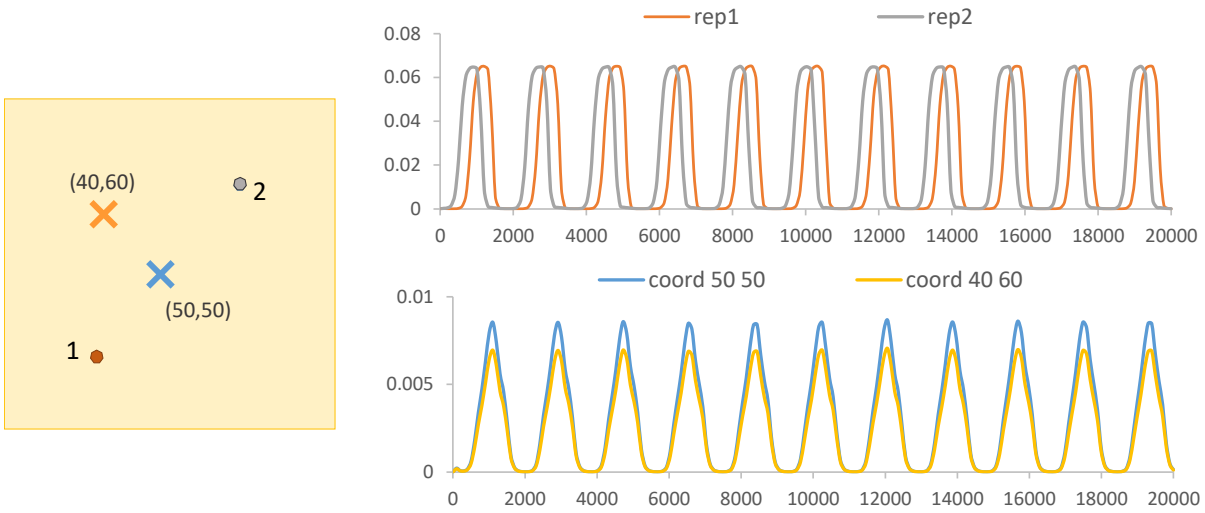


Fig. 26 Simulation of two repressilators on a lattice. The disposition is summarize on the cartoon on the left. AHL concentration is monitored at the two repressilators (upper diagram) and at two points at coordinates (40,60) and (50,50) (lower diagram).

We see that combining two repressilators on a lattice results in oscillations both at the location of the repressilators (upper diagram on Fig. 26) and other points on the lattice (lower diagram Fig. 26). The oscillations are sustained, regular and synchronized even if the two sources exhibit a phase shift between each other. This is a mandatory requirement for a synchronized clock. However, we notice that at these different points, the concentration is not the same. If the system relying on the clock is sensitive to a minimal concentration that is higher than the lowest amplitude of these different oscillations, this system would only work at particular zones of the lattice. As this might be a problem, we try to see if we can obtain a more uniform level of AHL over the whole lattice by increasing the number of repressilator units.

Three repressilators

Three repressilator units are disposed at the following coordinates: repressilator 1 at (50,50), repressilator 2 at (75,75) and repressilator 3 at (25,25). We monitor the concentration of AHL on the medium at these 3 coordinates and also at an additional point at coordinates (40,60). The disposition is shown on Fig. 27.

Each repressilator unit is set to a different initial condition. One of the three proteins has an initial concentration set to 0.1 whereas the two others start at 0. This protein is different for each repressilator: it is LacI, TetR and CI for respectively repressilator 1, 2 and 3.

We first observe the results for the default parameters (diagram D_0=1 on Fig. 28). The three repressilators are able to oscillate. Repressilators 2 and 3 oscillate at similar frequencies and this frequency is only slightly reduced over time. Concentration oscillations at the repressilator in the middle however gradually shift with regards to the two others, meaning that the frequency of these oscillations increases more than the frequency at repressilators 2 and 3. The repressilator most affected by diffusion seems to be the one in the middle. Since that with a higher diffusion coefficient, the molecules will travel more easily from repressilator 2 to 3, we try to increase it to see whether the diffusion can alter the oscillations of each repressilator so that they synchronize (Fig. 28).

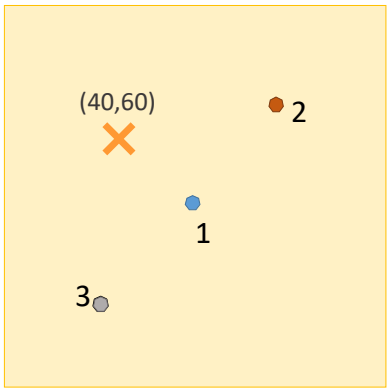


Fig. 27 Disposition of the 3 repressilators. The repressilators are represented by hexagons with their number. The cross corresponds to our additional observation point at coordinates (40,60).

Simulation results show that the higher the diffusion constant, the larger is the overlap between repressilator 2's oscillations and repressilator 3s'oscillations. However, with a high diffusion constant (superior to 5) oscillations at repressilator 1 are not regular anymore. When observing AHL concentration at a stop deprived of repressilators (namely the point at coordinates (40,60)), we see that regular oscillations are never formed. Moreover, the amplitude of the oscillations is not the same at every point of the lattice.

Decrease in the distance between the repressilators

We tried to position the repressilators closer to each other. Repressilator 1 is still at coordinates (50,50) whereas repressilator 2 and repressilator 3 are respectively at coordinates (60,60) and (40,40). Initial conditions are set as above. Simulation results show that the 3 repressilators have a similar frequency, which they keep during the 10000s of the simulation. They are however not in phase. As there seem to be no frequency shift, it is very unlikely that they would synchronize after a longer period of time. Noteworthy, the amplitudes of the oscillations are in the same range.

Random initial conditions

The disposition on Fig. 27 is used. To see the influence of initial conditions on the synchronization of oscillations, we tried to initialize each protein of the repressilators with random numbers between 0 and 1. Simulation results show that the oscillation of each repressilator gradually shifts with regards to one another, indicating that they are not synchronized.

Adding an external source of AHL

We use the disposition shown on Fig. 27 with default parameters except for D_0 which is set to 10. Moreover, we instantiate a source of AHL at coordinates (50,50) and with a value of 0.1. We set the initial concentrations of all proteins to 0. Repressilator 2 and 3 synchronize, whereas repressilator 1's oscillations are not sustained (data not shown). Their amplitudes are however very close (less than 5% of difference).

4.4.4. Conclusion

We see that it is difficult to obtain both sustained regular oscillations and oscillations with a uniform amplitude. Other configurations and other sets of parameters were tested, with similar results. Nevertheless, as it is possible to obtain sustained oscillations of similar amplitudes in particular zones, it could be envisioned to combine this imperfect synchronized spatial clock with the state machine studied in Chapter 4 (the counter).

A genetic programming could be used to find the optimal number, disposition and parameters of repressilators to successfully implement a spatial synchronized clock. This requires the spatial simulator to be interfaced with a genetic algorithm. During the time of this thesis, a first step consisting in interfacing an evolutionary algorithm with a SPICE simulator was realized. The next steps were still ongoing at the end of this thesis and are therefore not discussed in this manuscript.

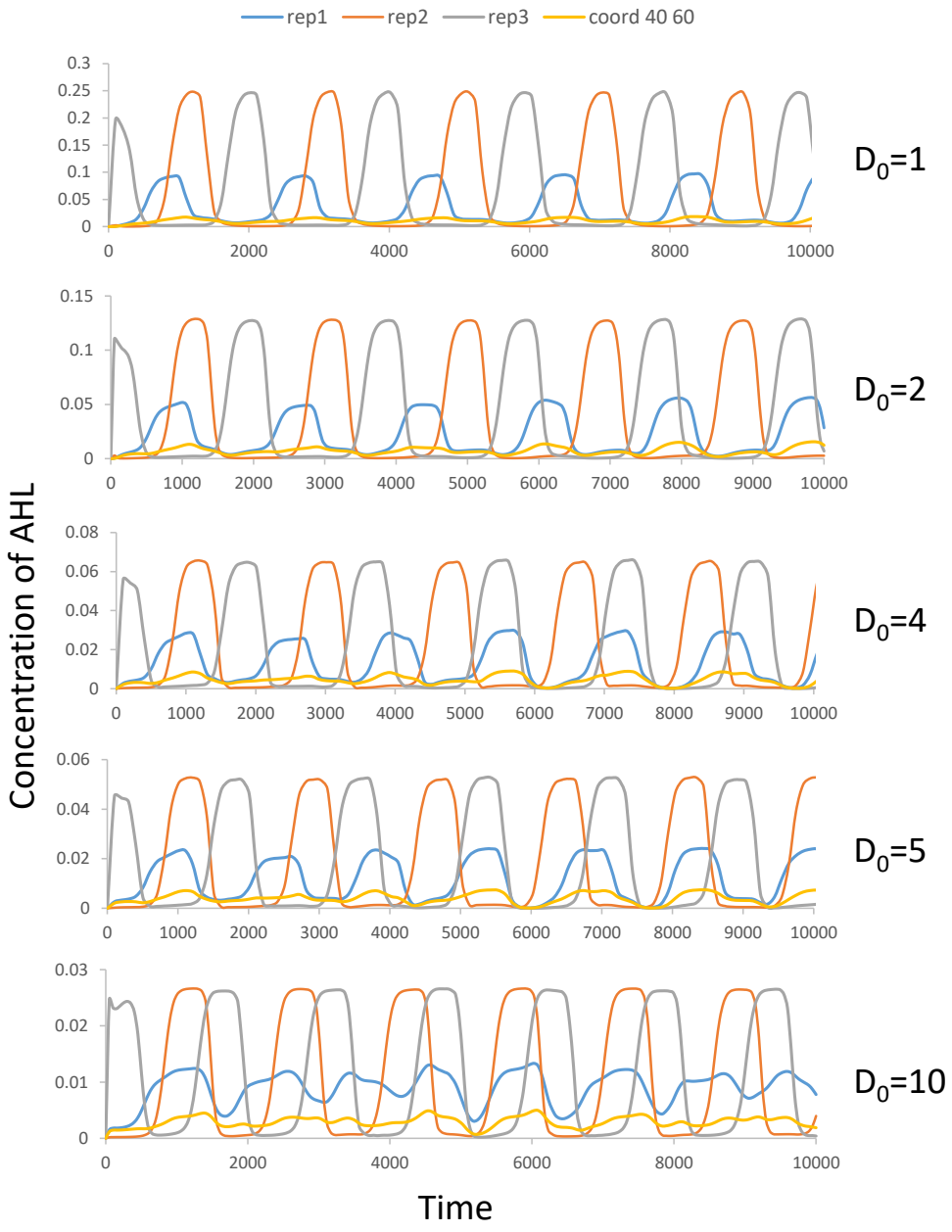


Fig. 28 Simulation results of the diffusion of AHL for a lattice with 3 repressilators. In blue, orange and grey the AHL concentration at the position of respectively repressilator 1, 2 and 3. For the position of these repressilators, see Fig. 27. The yellow curve corresponds to the concentration of AHL at coordinates (40,60).

5. Conclusion

Different systems were simulated with our tool and interesting results were generated. These results prove the validity of this approach (Sections 1 and 2). Moreover, we showed that our tool can be used for the study and the optimization of simple biological systems (Section 4). The use of the tool in actual

cases also highlighted some limitations of the tool as well as some ways of improvement. These are discussed in the conclusion of this part 3 hereafter.

6. Reference

Ausländer, Simon, David Ausländer, Marius Müller, Markus Wieland, and Martin Fussenegger. 2012. "Programmable Single-Cell Mammalian Biocomputers." *Nature* 487 (7405). Nature Publishing Group: 123–27. doi:10.1038/nature11149.

Balagadde, Frederick K, Hao Song, Jun Ozaki, Cynthia H Collins, Matthew Barnet, Arno, Frances H Ld, and Stephen R Quake. 2008. "REPORT A Synthetic Escherichia Coli Predator – Prey Ecosystem," no. 187: 1–8. doi:10.1038/msb.2008.24.

Balagaddé, Frederick K, Hao Song, Jun Ozaki, Cynthia H Collins, Matthew Barnet, Frances H Arnold, Stephen R Quake, and Lingchong You. 2008. "A Synthetic Escherichia Coli Predator-Prey Ecosystem." *Molecular Systems Biology* 4 (1): 187. doi:10.1038/msb.2008.24.

Basu, Subhayu, Yoram Gerchman, CH Collins, FH Arnold, and R Weiss. 2005. "A Synthetic Multicellular System for Programmed Pattern Formation." *Nature* 434 (April).

Brenner, Katie, David K Karig, Ron Weiss, and Frances H Arnold. 2007. "Engineered Bidirectional Communication Mediates a Consensus in a Microbial Biofilm Consortium." *Proceedings of the National Academy of Sciences of the United States of America* 104 (44): 17300–304. doi:10.1073/pnas.0704256104.

Elowitz, Michael B, and S Leibler. 2000. "A Synthetic Oscillatory Network of Transcriptional Regulators." *Nature* 403 (6767): 335–38. doi:10.1038/35002125.

Fujii, Teruo, and Yannick Rondelez. 2013. "Predator–Prey Molecular Ecosystems." *ACS Nano* 7 (1). American Chemical Society: 27–34. doi:10.1021/nn3043572.

Garcia-Ojalvo, J., M. B. Elowitz, and S. H. Strogatz. 2004. "Modeling a Synthetic Multicellular Clock: Repressilators Coupled by Quorum Sensing." *Proceedings of the National Academy of Sciences* 101 (30): 10955–60. doi:10.1073/pnas.0307095101.

Gendrault, Yves, Morgan Madec, Christophe Lallement, and Jacques Haiech. 2014. "Modeling Biology with HDL Languages: A First Step toward a Genetic Design Automation Tool Inspired from Microelectronics." *IEEE Transactions on Biomedical Engineering* 61 (4). IEEE Computer Society: 1231–40.

Konkoli, Zoran. 2011. "Safe Uses of Hill's Model: An Exact Comparison with the Adair-Klotz Model." *Theoretical Biology & Medical Modelling* 8 (1). BioMed Central Ltd: 10. doi:10.1186/1742-4682-8-10.

Madec, Morgan, Christophe Lallement, and Jacques Haiech. 2017. "Modeling and Simulation of Biological Systems Using SPICE Language." *PloS One* 12 (8). Public Library of Science: e0182385. doi:10.1371/journal.pone.0182385.

Russo, G., and M. Di Bernardo. 2009. "How to Synchronize Biological Clocks." *Journal of Computational Biology* 16 (2). Mary Ann Liebert, Inc. 2 Madison Avenue Larchmont, NY 10538 USA : 379–93. doi:10.1089/cmb.2008.21TT.

Shou, W., S. Ram, and J. M. G. Vilar. 2007. "Synthetic Cooperation in Engineered Yeast Populations." *Proceedings of the National Academy of Sciences* 104 (6): 1877–82. doi:10.1073/pnas.0610575104.

Tamsir, Alvin, Jeffrey J Tabor, and Christopher A Voigt. 2011. "Robust Multicellular Computing Using Genetically Encoded NOR Gates and Chemical 'Wires'." *Nature* 469 (7329): 212–15. doi:10.1038/nature09565.

Terzer, M, M Jovanovic, A Choutko, O Nikolayeva, A Korn, D Brockhoff, F Zu, et al. 2007. "Design of a Biological Half Adder." *IET Synthetic Biology*, 53–58. doi:10.1049/iet-stb.

Wagemakers, Alexandre, Javier M. Buldú, Jordi García-Ojalvo, and Miguel A. F. Sanjuán. 2006. “Synchronization of Electronic Genetic Networks.” *Chaos: An Interdisciplinary Journal of Nonlinear Science* 16 (1). American Institute of Physics: 13127. doi:10.1063/1.2173048.

Wang, Ruiqi, and Luonan Chen. 2005. “Synchronizing Genetic Oscillators by Signaling Molecules.” *Journal of Biological Rhythms* 20 (3). Sage PublicationsSage CA: Thousand Oaks, CA: 257–69. doi:10.1177/0748730405275653.

Zhou, Tianshou, Jiajun Zhang, Zhanjiang Yuan, and Luonan Chen. 2008. “Synchronization of Genetic Oscillators.” *Chaos: An Interdisciplinary Journal of Nonlinear Science* 18 (3). American Institute of Physics: 37126. doi:10.1063/1.2978183.

Summary of Part Three

As mentioned in the introduction of Part Three, it has become necessary to account for the spatio-temporal behavior of biological systems. Indeed, with the systems growing in size and complexity, many biologists are now splitting their systems into sub-systems implemented in different cell populations. As these cell populations are often scattered in space, they communicate by sending and receiving a molecular signal whose diffusion in space and time is of key importance for the proper functioning of the system.

Current tools for the simulation of biological systems over space and time include VirtualCell and COMSOL. The key findings of our state of the art about these tools are summarized thereafter.

On the one hand, Virtual Cell has an intuitive Graphical User Interface (GUI) that can be used to define the geometry of the problem and the biological reactions involved in the system. Several solvers are available, including deterministic and stochastic simulations for 0D to 3D problems. It was tested and led to results comparable to those obtained with our tool. Nevertheless, Virtual Cell performs only transient simulations. Static simulation results have been obtained by performing a transient simulation up to the steady state, which is time consuming. Computations are not done locally on the computer but are dispatched and executed on a remote server. Thus, comparison of computation time is not very relevant. Results given in Table 1 corresponds to clock-wall computation time and are independent of the computer on which they are performed. Virtual Cell also uses a lattice to solve PDE but the mesh size is fixed. Thus, to obtain a 1mm spatial resolution at the center, it is necessary to have a 1mm length mesh on the whole surface, which leads to 10201 nodes. It is between 3 to 4 times more than the number of nodes required to obtain the same spatial resolution near the source and the same accuracy with an adaptive lattice. Finally, another limitation of Virtual Cell is the difficulty to couple biological models with models from other domains of physics. It may be possible in several cases, but it requires a translation of physical problems into an equivalent biochemical problem.

On the other hand, COMSOL is a software dedicated to the simulation of multi-physics systems. It provides modules dedicated to the diffusion of chemical species and to the engineering of reactions that can be coupled together and with other modules from other domains of physics. It has already been used for the study of biochemical systems (Dreij et al, 2011; Vollmer et al, 2013). The PDE solver as well as the mesher implemented in COMSOL are more sophisticated than those proposed in our approach. The adaptive lattice can be computed automatically by COMSOL according to the geometry and the problem itself (detection of hot spots to determine automatically the refinement zones) whereas it has to be defined by the user in our case. The definition of the geometry and the diffusion equations are simplified by a GUI. Contrariwise, the definition of the reactions is less intuitive. Importing existing biological models or integrating complex reaction models is painful. The PDE resolution algorithms are various and optimized for a multi-core implementation, which leads to low computation times as shown in Table 1. Most common simulation types (i.e. static, dynamic, parametric) are available and COMSOL can be coupled with MATLAB to build an enhanced test bench. Unfortunately, COMSOL is an expensive commercial tool which limits its application range, especially in the academic field.

Compared to them, our approach offers a solution that positions itself as a good trade-off between accuracy and computation speed, relies on tools that have proved their efficiency for years (to simulate systems composed of 1-billion transistors microprocessors) and offers a direct coupling with other domains of

physics through the generalized Kirchhoff's laws. Comparison of several features of our approach, COMSOL and Virtual Cell are given in Table 1. Comparison between simulation results and computation time is not straightforward because of the different ways the simulators are implemented and the different computers on which the results were generated. The benchmark used for comparison is the one described in Chapter 7 Section 1 Table 2, *i.e.* a 100x100 mm² square with a source at (50,50) and no-flux boundary conditions. The spatial resolution near the source should be at least 1 mm. As Virtual Cell does not feature an adaptive mesher, a 100x100 lattice is used to discretize the whole surface. With COMSOL, an adaptive 1858-node mesh is generated so that the smallest elements are less than 1mm at the center. Finally, with our tools, different lattices have been tested. Reduction in the number of nodes obviously degrades the quality of the results but, in counterpart, reduces drastically the computation time. Results retained for the comparison are those from Mesh #7 (*i.e.* a 100x100 regular lattice) and Mesh #9 (*i.e.* a 25x25 adaptive lattice refined once in a 30mm radius centered circle and twice in a 5mm centered circle) which can be considered as a good trade-off between computation time and accuracy. Moreover, we compared the performances of Spectre and NGSPICE for our approach. The commercial simulator has better performance that the open-source one when the number of nodes exceeds 2000. This is even more marked on transient simulation. This is probably due to the multi-thread feature proposed by Spectre that is not yet supported by NGSpice.

Table 1 Comparison of the features and the performances of COMSOL, Virtual Cell, and our approach (both with Spectre and NGSpice).

	COMSOL	Virtual Cell	Spectre	NGSPICE
Mesher	Triangle Adaptive mesh Driven by physics	Rectangular Fixed	Rectangular Adaptive Defined by user	
Discretization scheme	Finite element (various scheme)	Finite Differences	Finite Differences Finite Element (*still under dev*.)	
Multiphysic interface	Yes Coupling with other COMSOL module	Not straightforward	Yes, using Kirchhoff-based equivalent circuits and HDL	
Simulation type	Deterministic Steady State, Transient, Parametric	Deterministic or Stochastic Transient	Deterministic Steady State, Transient, Parametric, Frequency analysis, Noise analysis	
Scripting for the development of complex testbenchs	Yes, with MATLAB	No	Yes, with SPICE simulation control directives	
Software type	Commercial	Freeware	Open-source model generator Commercial simulator	Fully open-source
Computation time at the steady state	< 1 sec for a 1858-nodes model		14.5 sec for the Mesh #7 (regular, 10201 nodes) 1.29 sec for Mesh #9 (adaptive, 1969 nodes).	20.5 sec for the Mesh #7 (regular, 10201 nodes) . 2.19 sec Mesh #9 (adaptive, 1969 nodes).
Computation time in transient	19 sec for a 1858-nodes model and 20 time steps	140sec for a 10201-nodes model and 20 time steps	18.7 sec for the Mesh #7 and 20 time steps 3.44 sec for The Mesh #9and 20 time steps	80.8 sec for the Mesh #7 and 20 time steps 13.18 sec for the Mesh #9 and 20 time steps

To summarize, our simulator exhibits four main advantages over existing tools:

- it is based on a very simple algorithm for the discretization of space, which facilitates the description of the diffusion phenomena with simple compact models
- it provides a direct coupling between the diffusion model of molecules, models of biological systems that play a role inside the diffusion medium and models from other domains of physics
- it uses a SPICE simulation core, which has proven its efficiency for years, especially for systems with a high number of differential equations and which will be improved in the near future in order to face the new challenges of microelectronics
- it is open-source.

With those features, it is a very powerful tool for the simulation and the virtual prototyping of biological systems inside cells or involving diverse types of cells that communicate between them through chemical messengers. For instance, it can be coupled with evolutionary algorithms to explore novel solutions that exploit cells consortia in synthetic biology or to gain a level of complexity in the modeling of biological systems.

Although the model is simplified due to the implemented discretization algorithm (in comparison with COMSOL models for instance), the simulation of a complete model can be very time-consuming, especially when the number of species increases. Considering the properties of the equations to solve, the deployment of Graphical Processor Unit (GPU) could provide a solution to speed-up the computation. Recent versions of GPU-optimized open-source SPICE simulator have been released (Keiter et al, 2014; Lannutti, 2014) and their coupling with our tool is currently under investigation. Another outlook of this work is to improve the way several mesh layers can be interconnected for systems with multiple diffusing species. When multiple mesh layers are implemented (*e.g.* in the simplified prey-predator example), the same refinement is applied on each layer to facilitate the interconnection between them (lattices of each layer overlap and the node coordinates are the same on every layers). Applying specific refinement on each layer would make the generation of the netlist more complex but, on the other hand, would reduce the number of nodes in the model and speed-up the computation.

Up to now, our simulator was only used inside our team (see below for the details). The expertise on the tool was therefore nearby and no GUI was required to properly run a simulation. However, a user-friendly interface would have facilitated the set-up and running of our tool.

Moreover, to widen the field of application, a validation on a use case of a different scale could be the next step. Indeed, in our examples, the cells (and therefore their compartments) are punctual. In some biological systems, cell-level simulation of the fluxes of molecules is required. A notable example is the study of calcium oscillations in a cell. These oscillations are involved in numerous processes in the cells and are therefore widely studied by the biology community. Calcium is stored in different compartments in the cell (e.g. the cytoplasm, the endoplasmic reticulum, the mitochondria…) and the fluxes of calcium between these compartments (the pattern of calcium oscillations) determine the behavior of a cell in reaction to an external signal. Being able to simulate these calcium exchanges at the level of a cell would surely bring a new piece of knowledge in the wide world of calcium oscillation patterns. Additionally, this would open the gates to predicting unobserved calcium patterns. With this possibility at hand, the designers would have a dedicated tool to modulate a cell's behavior. As dysregulations in calcium oscillations are of key importance in several diseases, an obvious application is therapeutics. A useful extension to a lower-scale usage of our tool would be the implementation of a 3D model.

In addition, as mentioned above, the tool has already been used in another project which is not directly connected to synthetic biology. The goal was to simulate the diffusion transport between droplets in a microfluidic chip. In such device, each droplet can be seen as a single independent bioreactor. However,

the leakage of chemical products from a droplet to the other changes the chemical composition inside the reactor and introduces a crosstalk between each assay. Our simulator has been used to simulate the diffusion phenomenon. More precisely, the droplets are described as two 2D disks with a high diffusion coefficient separated by oil which is modeled by a low diffusion coefficient. Simulation results are promising and can be used to optimize the device in the future.

To conclude, we developed an open-source tool that can be used for the modeling and the simulation of biological systems that depend on space and time. The applications are numerous in systems biology, in synthetic biology coupled with our virtual prototyping and/or automation design environment and for applications at the interface with other domain of physics whose phenomena can be modeled by diffusion equation and/or Kirchhoff networks, *e.g.* electronic devices, thermal phenomena, microfluidics in lab-on-chips and biosensors.

Conclusion & Outlook

Our contribution to the wondrous world of synthetic biology

In Part One, we presented a design flow for biology. Our work palliates the two identified missing elements by providing a solution for the automatic design of biological systems and an (almost) ready-to-use open source space and time simulator. As our tools are not encapsulated by a limiting graphical user interface or by a cloaking commercial license (sometimes you do not really know what these software really do…), our environment is open to extension. By our choice of description language, simulating mixed systems comprising elements from other domains than biology is easier with our tool than with existing biology-dedicated ones. Moreover, our SBML-handling translator, BB-SPICE, allows an easy specification of novel biological systems into the Verilog format. The EASEA platform, as it name suggests, also facilitates the adaptation of evolutionary algorithms to any biological problems even for any young padawan of programming.

Ease the developments of innovative systems

Our design flow is now almost complete and permits the handling of complex biological systems, from the design step to the simulation. A recent example (Müller et al. 2017) possesses all the features to illustrate the potential contribution of our tool. Müller et al. designed a biological analog-to-digital converter. Their system involves an analog part, the biosensor and a digital part composed of a Boolean circuit. They also split their system over different cell populations separated by a liquid interface where diffusion happens. With our tool, they could have performed the virtual prototyping of the GRN to implement in these different cell populations. Different solutions would probably have been found by the algorithm, giving them alternatives to choose from. Having a choice allows to order the suggested solutions by criteria that might otherwise not have been considered. Besides enlarging the horizon of the designers, our environment would probably have allowed them to save some time at the bench, by proceeding to spatiotemporal simulations of the systems to be tried.

Missing connections

As the bridge from an *in silico* simulation to the wet bench experiments requires an actual implementation of the designed system, we need to mention here the limitations of the current environment. Indeed, several connections are missing, notably to a database of parts. As mentioned in Part One, there is no mature standard database of parts for synthetic biology. The most advanced library is currently the BioBrick repository, even though a standard description of a BrioBrick is not implemented yet. As this ideal database does not exist, we did not feature the possibility for the design tool to connect to such a repository. A first outlook would be to manually curate such a library and connect our tool to it, for the time being. Moreover, an actual implementation of a GRN requires more than just being able to provide the user with the DNA sequences corresponding to the parts of the system. Experimental constraints related to wet bench implementation would probably need to be considered. As the experimental side of this thesis was rather scarce, I gradually shifted to the dark side of synthetic biology, which corresponds to the upstream work

(namely tool development) that has to be completed so that experimenters and designers have tools at hand to perform their task.

The utopian tool

Beyond these missing links, other improvements were envisioned for the design flow. The connections between the different tools could be boosted. Indeed, the design tool relies on the capacity to simulate the biological system to achieve. As seen in Chapter 5, simulation is required to assess the fitness score used to evaluate the different solutions to a biological problem. Up to now, ODEs only were used to simulate GRNs in the design tool. When designing a system displaying a spatiotemporal behavior however, the simulation tool we developed would need to be connected to the design tool. A first outlook would be to create a weak interaction between these tools: the design tool would only be able to run SPICE simulations on a fixed lattice. A second outlook would require to bring intelligence in the choice of the lattice. A first step would be to have the algorithm choose more and more refined lattices to simulate the system over the course of the convergence (the refinement of the lattice increases with the iterations of the evolutionary algorithm). In a second step, we could imagine that the algorithm would be able to optimize the lattice as well with regards to the ratio of the precision over the number of nodes (as mentioned in Chapter 7, the finer the lattice, the more precise the results). These trails are left open for the synthetic biology PhD newcomers…

List of publications

Papers in international peer reviewed journal

[1] M. Madec, F. Pêcheux , Y. Gendrault, E. Rosati, C. Lallement, J. Haiech, GeNeDA: an open-source workflow for design automation of gene regulatory networks inspired from microelectronics, Journal of Computational Biology, Volume 13, n° 6, juin 2016, doi:10.1089/cmb.2015.0229

[2] E. Rosati, M. Madec, J.-B. Kammerer, L. Hébrard, C. Lallement, J. Haiech, Efficient modeling and simulation of space-dependent biological systems, Journal of Computational Biology, mai 2018, doi: 10.1089/cmb.2018.0012

Papers in national peer reviewed journal

[3] M. Madec, J. Haiech, E. Rosati, A. Rezgui, Y. Gendrault, C. Lallement. Application à la biologie synthétique des méthodes et outils de CAO de la microélectronique, Médecine/Sciences, EDP Sciences, Volume 32, n° 2, 2017

International conferences with proceedings and peer review process

[4] E. Rosati, M. Madec, F. Pêcheux , Y. Gendrault, C. Lallement, J. Haiech, Design and simulation of a compact genetic flip-flop, Advances in Systems and Synthetic Biology, Strasbourg, France, pages 115-124, mars 2015

[5] L. Talide , Z. Blanck , M. Renou , T. Wallois , E. Rosati, M. Madec, A. Rezgui, C. Lallement, J. Haiech, Modeling of intercellular transport for emerging applications in synthetic biology, Advances in Systems and Synthetic Biology, Strasbourg, France, pages 149-156, mars 2015

[6] E. Rosati, M. Madec, J.-B. Kammerer, A. Rezgui, C. Lallement, J. Haiech, Verilog-A Compact Space-dependent Mode for Biology, 22nd IEEE International Conference on Mixed Design of Integrated Circuits & Systems (MIXDES 2015), Torun, Poland, pages 171-176, juin 2015, doi:10.1109/MIXDES.2015.7208505.

[7] E. Rosati, M. Madec, A. Rezgui, Q. Colman, N. Toussaint, C. Lallement, P. Collet, Application of Evolutionary Algorithms in Synthetic Biology, EuroGP - EvoStar 2016, Porto, Portugal, mars 2016.

[8] E. Rosati, M. Madec, J-B. Kammerer, A. Rezgui, C. Lallement, J. Haiech, Verilog-A Compact Space-dependent Model for Biology, Advances in Systems and Synthetic Biology, Evry, France, pages 103-117, Genopole (Eds.), EDP science, ISBN : 978-2-7598-1971-3, Volume 15, mars 2016

[9] A. Biquet, R. Goerlich, E. Rosati, M. Madec, C. Lallement, Modeling of biological asynchronous sequential systems, Advances in Systems and Synthetic Biology, Evry, France, Genopole (Eds.), EDP science, ISBN : en cours, Volume 17, mars 2018.

[10] M. Madec, A. Bonament, E. Rosati, L. Hebrard, C. Lallement, Virtual Prototyping of Biosensors Involving Reaction-diffusion phenomena, 16th IEEE International Conference on New Circuits and Systems (NEWCAS 2018), Montreal, Canada, juin 2018.

International conference without proceeding

[11] M. Madec, F. Pêcheux , Y. Gendrault, E. Rosati, C. Lallement, J. Haiech, Reuse of microelectronics software for gene regulatory networks design automation, BioSynSys 2015, Paris, France, septembre 2015

[12] E. Rosati, M. Madec, Q. Colman, N. Toussaint, A. Rezgui, C. Lallement, J. Haiech, P. Collet, A Nature-Inspired Evolutionnary Algorithm for the Design and Optimization of Gene Regulatory Networks, BioSynSys 2015, Paris, France, septembre 2015

[13] E. Rosati, M. Madec, J-B. Kammerer, A. Rezgui, C. Lallement, J. Haiech, A Verilog-A Mesher Based on Eletronics to Model Space-Dependent Biological Systems, BioSynSys 2015, Paris, France, septembre 2015.

[14] M. Madec, A. Rezgui, Y. Gendrault, E. Rosati, F. Pêcheux , C. Lallement, J. Haiech, Application of Electrical Design Automation in Biological Context, CDN Live 2016, Munich, Germany, mai 2016. Academic Track Best Paper Award

[15] O. Bolaji, M. Madec, E. Rosati, C. Lallement, Evaluation of Open Source Electrical Simulator in a Biological Context, 2nd international conference of the GDR BioSynSys, Bordeaux, France, juin 2016

[16] N. Dumas, E. Rosati, M. Madec, V. Pasteur, D. Funfschilling, Simulating diffusion transport between droplets in a microfluidic chip, Proceedings of the 5th European Conference on Microfluidics, Debruary 28 – March 2, 2018, Strasbourg, France

National conference without proceeding

[16] E. Rosati, M. Madec, F. Pêcheux, Y. Gendrault, C. Lallement, J. Haiech, Development of design tools for biosystems engineering, Journées du Campus d'Illkirch, mai 2014.

[17] E. Rosati, M. Madec, J.-B. Kammerer, L. Hébrard, C. Lallement, J. Haiech, Modélisation compacte Verilog-A des problems biologiques dependent de l'espace, Journées Nationales du Réseau des Doctorants en Microelectronique, Strasbourg, France, novembre 2017.

Appendices

Appendix I
Demonstration Used in the Finite Element Discretization Scheme

1. Basics

1.1. Notation

Let Ω be a rectangular plane. Let Σ be the border of Ω. We define $T(x, y)$ the temperature at a point of coordinates(x, y). We will divide the space in rectangular elements Ω of length L and width W (see Fig. 1).

1.2. Equation of heat conduction

1.1.1. Heat equation

In thermal physics, Fourier law gives us:

$$\boldsymbol{q} = -\lambda \cdot \mathbf{grad}(T)$$

where $\mathbf{q}$ is the rate of flow of heat energy per area unit, k the thermal conductivity (supposed to be isotropic) and $\vec{\nabla}T$ the gradient of temperature.

On the other hand, the first principle of thermodynamics (conservation law) states that locally

$$\mathrm{div}(\mathbf{q}) + P = 0$$

where P is the heat power received by the system. Combining previous equations leads to the heat equation without external sources:

$$C\frac{\partial T}{\partial t} = \lambda \cdot \Delta T$$

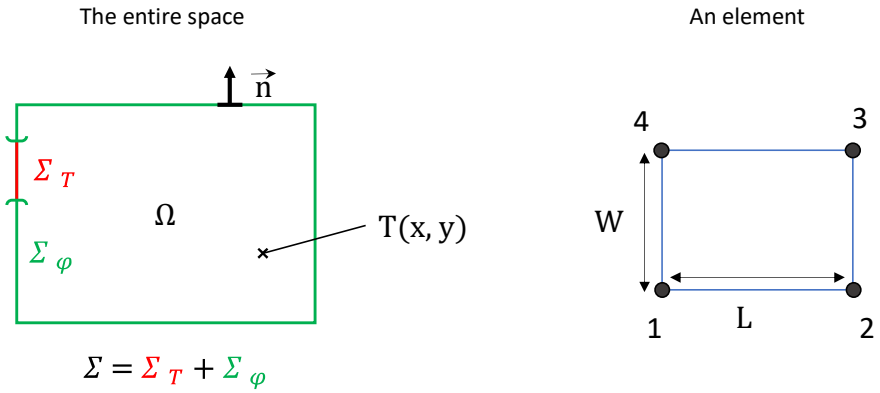


Fig. 1 Notations

1.1.2. Boundary conditions

Boundaries are divided in two subsets: Σ_T is the subset for which a temperature boundary condition is defined:

$$T(x,y) = T_0 \quad \forall (x,y) \in \Sigma_T$$

Σ_φ is the subset for which a flux boundary condition is defined.

$$\lambda \cdot \mathbf{grad}(T(x,y)) \cdot \mathbf{n} = q_n \qquad \forall (x,y) \in \Sigma_\varphi$$

In our case, we have:

$$q_n = \varphi_S + h(T_f - T)$$

With φ_S the constraint on incoming flux and $h(T_f - T)$ the expression of convection with T_f the ambient temperature far from the modeled surface.

1.3. Variational formulation

Let $\hat{T}$ be a function which associates to a point (x,y) an arbitrary temperature T. $\hat{T}$ is also constrained on the border Σ_T by the following condition:

$$\hat{T}(x,y) = 0 \quad \forall (x,y) \in \Sigma_T$$

There is an infinity of such functions. Let us multiply the conservation equation by $\hat{T}$ and integrate on the whole plane Ω:

$$\int_\Omega \hat{T}(\mathrm{div}(\mathbf{q}) - P)dS = 0$$

From Liebnitz formula, we know that:

$$\mathrm{div}(\hat{T}\mathbf{q}) = \hat{T}\mathrm{div}(\mathbf{q}) + \mathbf{grad}(\hat{T}) \cdot \mathbf{q}$$

So that:

$$\int_\Omega \mathrm{div}(\hat{T}\mathbf{q})dS - \int_\Omega \left(\mathbf{grad}(\hat{\ }) \cdot \mathbf{q}\right)dS - \int_\Omega \hat{T} \cdot P \, dS = 0$$

The Green-Ostrogradski theorem applied on the first term leads to:

$$\int_\Sigma \hat{T}(\mathbf{q} \cdot \mathbf{n})dL - \int_\Omega \left(\nabla\hat{T} \cdot \boldsymbol{q}\right)dS - \int_\Omega \hat{T} \cdot P \, dS = 0$$

Finally, by separating Σ in Σ_T and Σ_φ, we can write that:

$$\int_\Omega \left(\mathbf{grad}(\hat{\ }) \cdot \mathbf{q}\right)dS = \int_{\Sigma_T} \hat{T}(\mathbf{q} \cdot \mathbf{n})dL + \int_{\Sigma_\varphi} \hat{T}(\mathbf{q} \cdot \mathbf{n})dL - \int_\Omega \hat{T} \cdot P \, dS$$

The integral on Σ_T is equal to zero due to the boundary condition. Replacing $\mathbf{q}$ by its expression and introducing q_n (boundary condition on and replacing Σ_φ) leads to the following expression.

$$\int_\Omega \mathbf{grad}\widehat{\ })^T(-\lambda\, \mathbf{grad}(T\ )dS = \int_{\Sigma_\varphi} \hat{T}(-q_n)dL - \int_\Omega \hat{T} \cdot P \, dS$$

Now, we have to find the expression of $T(x, y)$ that verifies previous equation for any function $\hat{T}(x, y)$ described previously.

2. Space discretization – General case

Let us divide the plane Ω in n elemental surfaces S_e and the border Σ_φ in elemental lengths $\Sigma_{\varphi i}$ (see Fig. 2).

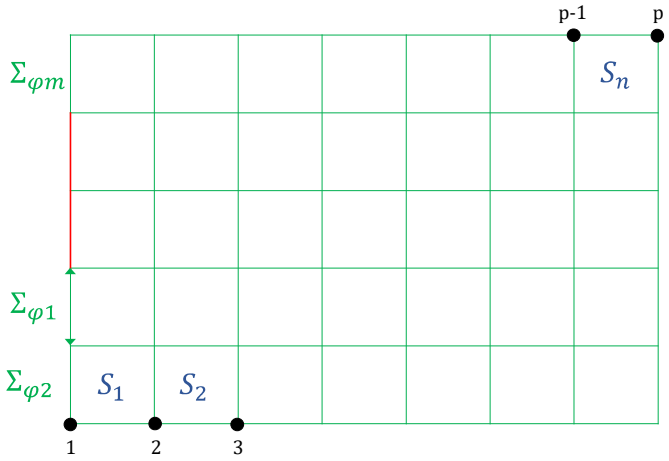


Fig. 2 Discretization of the plane Ω.

For each element S_e, we define an interpolation function N^e as:

$$T(x, y) = \sum_{i=1}^{nnode} N_i^e(x, y)T_i^e \qquad \forall (x, y) \in S_e$$

With $nnode$ the number of nodes of element S_e, N_i^e the interpolation functions of S_e associated with the node i (its value is 0 outside of S_e) and T_i^e the temperature at the node i of S_e (i.e. $T_i^e = T(x_i^e, y_i^e)$).

In case of a 4-nodes element, the previous equation can be written as follows:

$$T(x,y) = [N^e(x,y)] \begin{pmatrix} T_1^e \\ T_2^e \\ T_3^e \\ T_4^e \end{pmatrix} = [N_1^e \quad N_2^e \quad N_3^e \quad N_4^e] \begin{pmatrix} T_1^e \\ T_2^e \\ T_3^e \\ T_4^e \end{pmatrix} \quad \forall (x,y) \in S_e$$

Based on Galerkin's method, we chose for $\hat{T}$ the same approximation:

$$\hat{T}(x,y) = \sum_{i=1}^{nnode} N_i^e(x,y)\hat{T}_i^e \quad \forall (x,y) \in S_e$$

We can therefore explicit the gradient of T as follows:

$$\mathbf{grad}\big(T\ (x,y)\big) = \mathbf{grad}\left(\sum_{i=1}^{nnode} N_i^e(x,y)T_i^e\right) = \sum_{i=1}^{nnode} \mathbf{grad}(N_i^e(x,y)) \cdot T_i^e = \mathbf{grad}(\mathbf{N_e}(x,y))^T \times \mathbf{T_e}$$

Where $\mathbf{N_e}$ is a vector matrix of dimension $nnode$ regrouping the $N_i^e(x,y)$ and $\mathbf{T_e}$ a vector matrix of dimension $nnode$ regrouping the T_i^e.

$$\mathbf{grad}\big(T\ (x,y)\big) = \mathbf{grad}\left([N^e(x,y)] \begin{pmatrix} T_1^e \\ T_2^e \\ T_3^e \\ T_4^e \end{pmatrix}\right) = \boldsymbol{grad}([N^e(x,y)]) \begin{pmatrix} T_1^e \\ T_2^e \\ T_3^e \\ T_4^e \end{pmatrix}$$

$$\nabla T\ (x,y) = \begin{pmatrix} \frac{\partial N_1^e}{\partial x} & \frac{\partial N_2^e}{\partial x} & \frac{\partial N_3^e}{\partial x} & \frac{\partial N_4^e}{\partial x} \\ \frac{\partial N_1^e}{\partial y} & \frac{\partial N_2^e}{\partial y} & \frac{\partial N_3^e}{\partial y} & \frac{\partial N_4^e}{\partial y} \end{pmatrix} \begin{pmatrix} T_1^e \\ T_2^e \\ T_3^e \\ T_4^e \end{pmatrix} = \sum_{i=1}^{nnode} T_i^e \nabla N_i^e(x,y) \quad \forall (x,y) \in S_e$$

In the same way we obtain:

$$\mathbf{grad}\left(\hat{T}\ (x,y)\right) = \sum_{i=1}^{nnode} \mathbf{grad}(N_i^e(x,y)) \cdot \hat{T}_i^e = \mathbf{grad}(\mathbf{N_e}(x,y)) \times \hat{\mathbf{T}}_{\boldsymbol{e}}$$

$\hat{\mathbf{T}}_{\boldsymbol{e}}$ a vector matrix of dimension $nnode$ regrouping the $\hat{T}_i^e$.

When introducing the new values for $\mathbf{grad}\big(T\ (x,y)\big)$ and $\mathbf{grad}\left(\hat{T}\ (x,y)\right)$ in the variational formulation expression and integrating on the whole surface Ω, we obtain:

$$\sum_{e=1}^{n} \hat{\mathbf{T}}_{\boldsymbol{e}}{}^{\mathrm{T}} \times \int_{S_e} \mathbf{grad}(\mathbf{N^e}(x,y))^{\mathrm{T}} \cdot \lambda \cdot \mathbf{grad}(\mathbf{N^e}(x,y)) \cdot dS \times \mathbf{T^e} - \left[\int_{\Sigma_\varphi \cap S_e} \mathbf{N^e} \cdot q_n \cdot dL + \int_{S_e} \mathbf{N^e} \cdot P \cdot dS\right] \times \hat{\mathbf{T}}_{\boldsymbol{e}} = 0$$

Let

$$\mathbf{K_e} = \int_{S_e} \mathbf{grad}(\mathbf{N^e}(x,y))^{\mathrm{T}} \cdot \lambda \cdot \mathbf{grad}(\mathbf{N^e}(x,y)) \cdot dS$$

and

$$\mathbf{F_e} = \left[\int_{\Sigma_\varphi \cap S_e} \mathbf{N^e} \cdot q_n \cdot dL + \int_{S_e} \mathbf{N^e} \cdot P \cdot dS\, dS\right]^{\mathrm{T}}$$

Equation X can be rewritten:

$$\sum_{e=1}^{n} \widehat{\mathbf{T}}_e{}^{\mathrm{T}} \times (\mathbf{K_e} \times \mathbf{T_e} - \mathbf{F_e}) = 0$$

It should be noticed that $\boldsymbol{F_e}^{\mathrm{T}} \times \widehat{\mathbf{T}}_e$ is a scalar, thus it is equal to $\widehat{\mathbf{T}}_e{}^{\mathrm{T}} \times \boldsymbol{F_e}$

Now, let $\widehat{\mathbf{T}}$, $\mathbf{T}$, $\mathbf{K}$ and $\mathbf{F}$ be vector matrices (for $\widehat{\mathbf{T}}$ and $\mathbf{T}$) and square matrices ($\mathbf{K}$ and $\mathbf{F}$) of dimension d (with d being the number of nodes of the whole surface p minus the number of nodes on Σ_T), regrouping the element matrices $\widehat{\boldsymbol{T}}_e$, $\boldsymbol{T_e}$, $\boldsymbol{K_e}$ and $\boldsymbol{F_e}$ respectively. For this purpose, first element matrices are extended to the dimension d and completed by zeros for the terms that concern nodes that are not involved in the mesh. Then, extended matrices are added to form $\widehat{\mathbf{T}}$, $\mathbf{T}$, $\mathbf{K}$ and $\mathbf{F}$.

Finally, since the previous expression is true for any $\widehat{\mathbf{T}}$, we can write:

$$\boldsymbol{K} \times \boldsymbol{T} - \boldsymbol{F} = 0$$

$\boldsymbol{K}$ is the matrix of rigidity.

3. Space discretization – 4-nodes models

In this case, we consider that the plane Ω is divided in n rectangular surfaces S_e.

3.1. Matrix of rigidity $\mathbf{K_e}$

We use the notation for an element as seen on the right-hand panel of Fig. 1. Let us first find N_e.

3.1.1. $\boldsymbol{N^e}$ calculation

Let T_1, T_2, T_3 and T_4 be the temperature at the four corners of the rectangular surfaces S_e (Fig. 1).

$$\mathbf{T_e} = \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix}$$

We define $\mathbf{N_e}(x, y)$ as a linear combination of four functions of x and y, named canonical base. In our case, we chose the standard bilinear canonical base $(1, x, y, xy)$. To find the coefficient of the linear combination, we use the values of T_1, T_2, T_3 and T_4. Let $\mathbf{P}$ be a 4x4 matrix which contain these coefficients.

$$\mathbf{N_e}(x, y) = (1 \quad x \quad y \quad xy) \times \mathbf{P}$$

Thus,

$$T_e(x, y) = (1 \quad x \quad y \quad xy) \times \mathbf{P} \times \mathbf{T_e}$$

Let

$$\mathbf{P} \times \mathbf{T}_e = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix}$$

With the temperature at the four corners, we can write:

$$T_e(0,0) = T_1 = P_1$$

$$T_e(L,0) = T_2 = P_1 + LP_2$$

$$T_e(L,W) = T_3 = P_1 + LP_2 + WP_3 + WLP_4$$

$$T_e(0,W) = T_4 = P_1 + WP_3$$

Writing these for equations as a vector-matrix product leads to:

$$\mathbf{T}_e = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & L & 0 & L \\ 0 & 0 & 0 & L \cdot W \\ 0 & 0 & W & W \end{pmatrix}$$

Inversion of the 4x4 matrix lead to the expression of $\mathbf{P}$.

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{L} & \frac{1}{L} & 0 & 0 \\ -\frac{1}{W} & 0 & 0 & \frac{1}{W} \\ \frac{1}{L \cdot W} & -\frac{1}{L \cdot W} & \frac{1}{L \cdot W} & -\frac{1}{L \cdot W} \end{pmatrix}$$

Finally $\mathbf{N}_e$ equals

$$\mathbf{N}_e(x,y) = \left(1 - \frac{x}{L} - \frac{y}{W} - \frac{xy}{L \cdot W} \quad \frac{x}{L} - \frac{xy}{L \cdot W} \quad \frac{xy}{L \cdot W} \quad \frac{y}{W} - \frac{xy}{L \cdot W}\right)$$

3.1.2. $\boldsymbol{K_e}$ calculation

The next step is to compute $\mathbf{grad}\ \mathbf{N}_e$.

$$\mathbf{grad}\ \mathbf{N}_e = \begin{pmatrix} \frac{\partial N_{e1}}{\partial x} & \frac{\partial N_{e2}}{\partial x} & \frac{\partial N_{e3}}{\partial x} & \frac{\partial N_{e4}}{\partial x} \\ \frac{\partial N_{e1}}{\partial y} & \frac{\partial N_{e2}}{\partial y} & \frac{\partial N_{e3}}{\partial y} & \frac{\partial N_{e4}}{\partial y} \end{pmatrix} = \begin{pmatrix} -\frac{1}{L} + \frac{y}{L \cdot W} & \frac{1}{L} - \frac{y}{L \cdot W} & \frac{y}{L \cdot W} & -\frac{y}{L \cdot W} \\ -\frac{1}{W} + \frac{x}{L \cdot W} & -\frac{x}{L \cdot W} & \frac{x}{L \cdot W} & \frac{1}{W} - \frac{x}{L \cdot W} \end{pmatrix}$$

Finally

$$\boldsymbol{K_e} = \int_{S_e} {}^t[\mathbf{grad}\ \mathbf{N}_e] \times \lambda \times\ \mathbf{grad}\ \mathbf{N}_e dS\, \lambda = \begin{pmatrix} \frac{W}{3L} + \frac{L}{3W} & -\frac{W}{3L} + \frac{L}{6W} & -\frac{W}{6L} - \frac{L}{6W} & \frac{W}{6L} - \frac{L}{3W} \\ * & \frac{W}{3L} + \frac{L}{3W} & \frac{W}{6L} - \frac{L}{3W} & -\frac{W}{6L} - \frac{L}{6W} \\ * & * & \frac{W}{3L} + \frac{L}{3W} & -\frac{W}{3L} + \frac{L}{6W} \\ * & * & * & \frac{W}{3L} + \frac{L}{3W} \end{pmatrix}$$

Since $\boldsymbol{K_e}$ is a symmetrical matrix, only half of the matrix is shown.

3.1.3. Representation as a resistor network

In this paragraph, we will see how to draw the analogy between the finite element discretization scheme for an elementary mesh and a representation of this mesh as an electrical resistor network. The model of the elementary is composed of 4 nodes and 6 resistors (Fig. 3). Let R_{ij} (or R_{ji}) be the resistor between the nodes i and j. Let $I_{ext,i}$ be the external current applied on node i.

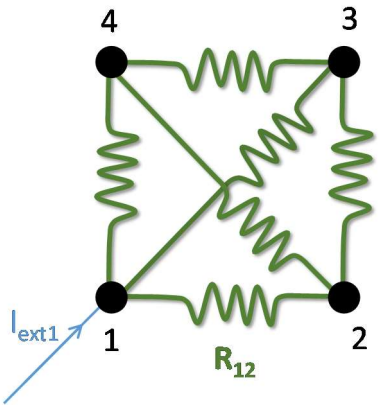


Fig. 3 The electrical mesh used in the diffusion model. Here all nodes are connected to one another with resistors. External currents can be taken into account on each node.

With Kirchhoff's law we can write the following equation:

$$\sum_{i=1,i\neq j}^{4} \frac{V_j - V_i}{R_{ij}} - I_{ext,j} = 0 \qquad \text{for } j = \{1,2,3,4\}$$

Matrix formulation of the previous equation is:

$$\mathbf{G_e} \times \mathbf{V_e} = \mathbf{G_e} \times \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix} = \begin{pmatrix} I_{ext,1} \\ I_{ext,2} \\ I_{ext,3} \\ I_{ext,4} \end{pmatrix} = \mathbf{I_{ext,e}}$$

With the coefficient of $\mathbf{G_e}$ defined as follows:

$$g_{ij} = \begin{cases} \sum_k \frac{\lambda}{R_{ik}} & if\ i = j\ \ \forall k \neq i \\ -\frac{\lambda}{R_{ik}} & if\ i \neq j \end{cases}$$

In a similar fashion as in the previous section (extension to the whole lattice and addition), we can create the square matrix $\mathbf{G}$ of dimension d and a d-element vector $\mathbf{I_{ext}}$ such as :

$$\mathbf{G} \times \mathbf{V} - \mathbf{I_{ext}} = \mathbf{G} \times \begin{pmatrix} V_1 \\ \vdots \\ V_d \end{pmatrix} - \mathbf{I_{ext}} = 0$$

Using the electrical/thermal analogy (voltage are equivalent to temperature and current to heat flow), we can identify this equation to the $\mathbf{K} \times \mathbf{T} - \mathbf{F} = 0$ define hereabove. The conductance matrix $\mathbf{G}$

corresponds to the rigidity matrix $\mathbf{K}$ and $\mathbf{I_{ext}}$ corresponds to the matrix $\mathbf{F}$ of incoming external fluxes. The same analogy can go down to the level of the elementary mesh between $\mathbf{G_e}$ and $\mathbf{K_e}$. Thus

$$R_{12} = R_{34} = \frac{6L}{-L^2+2W^2} \qquad R_{13} = R_{24} = \frac{6LW}{L^2+W^2} \qquad R_{14} = R_{23} = \frac{6LW}{-W^2+2L^2}$$

When meshes are squares, $W = L$, hence:

$$R_{adj} = 6 \quad \text{and} \quad R_{diag} = 3$$

With R_{adj} the resistance value between two adjacent nodes and R_{diag} the relative resistance value between two diagonal nodes (to have the absolute value one still needs to multiply by λ which corresponds to the diffusion constant of the molecule).

3.2. Computation of the external fluxes (matrix $\mathbf{F_e}$)

3.2.1. Computation

We want to compute the matrix $\mathbf{F_e}$ defined as following:

$$\mathbf{F_e} = \left[\int_{\Sigma_\varphi \cap S_e} \mathbf{N^e} \cdot q_n \cdot dL + \int_{S_e} \mathbf{N^e} \cdot P \cdot dS\right]^{\mathrm{T}}$$

We first focus on an element which is not at the border so that $\Sigma_\varphi \cap S_e = \emptyset$. Thus:

$$\mathbf{F_e} = \left[\int_{S_e} \mathbf{N^e} \cdot P \cdot dS\right]^{\mathrm{T}}$$

Here P is the flux incoming the element. We first assume there is no external heat source. Thus, the only flux to consider is the convection modeled by a term $P = -h \cdot T$ with h a constant representing coefficient of convection. Using the approximation of T, we get the following expression for P:

$$P = -h \cdot T(x,y) = -h \cdot \mathbf{N^e} \cdot \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix}$$

We therefore obtain for F_e:

$$\boldsymbol{F_e} = -h \cdot \frac{LW}{36} \cdot \begin{pmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{pmatrix} \times \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix}$$

3.2.2. Representation as a resistor network

In the electrical model, a first model represented degradation by connecting a grounded resistor to each node (see Fig. 4).

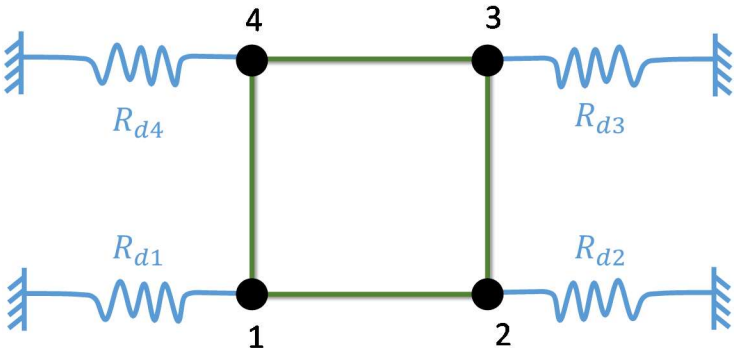


Fig. 4 Initial model of degradation. Each node is connected to a grounded resistor.

The current passing through the resistor R_{di} connected to node i corresponds to the previously defined $I_{ext\,i}$. With the Ohm's law we can write:

$$I_{ext\,i} = -\frac{V_i}{R_{di}}$$

We assume the degradation rate of the molecule is constant over space, so that all the resistors have the same value R_d:

$$R_{di} = R_d \qquad \forall i$$

We obtain for the matrix $I_{ext,e}$:

$$\boldsymbol{I_{ext,e}} = -\frac{1}{R_d}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$$

As seen in the previous section, an analogy can be drawn between $\boldsymbol{I_{ext,e}}$ and $\boldsymbol{F_e}$. We therefore need to find a value of R_d that satisfies the following equation:

$$-d_X \frac{LW}{36}\begin{pmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{pmatrix} = -\frac{1}{R_d}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

It is obviously impossible with this model. Noteworthy, even with different R_{di} values for each node, this equation cannot be solved.

This means that according to finite element method, degradation of a molecule at a certain point not only depends on its own concentration but also on the concentration of its neighboring nodes (corresponding to the 0 in the $\boldsymbol{I_{ext,e}}$ matrix). Noteworthy is also the fact that the major contribution to compute the degradation rate comes from the node itself (4 is the highest value on a line and is always position on the diagonal) and that the closest nodes contribute more that the farthest ones (the value 1 always corresponds to the diagonal node, e.g. V_3 contributes 4 times less that V_1 to V_1's degradation).

Instead of just one resistor, one would actually need to implement as many resistors as coefficients in the matrix. Another solution is envisioned: the new electrical model of an element has a VCCS (Voltage

Controlled Current Source) for each node. This VCCS is dependent on 4 voltages, the potential of the 4 nodes of an element. Moreover, this VCCS is dependent on the size of the element.

We know have to compute the F_e matrix for border nodes.

3.3. Computation of the border conditions

We distinguish 3 types of borders we might want to model. First, the limit of the lattice represents a "wall" the diffusing molecules cannot cross. Second, a fixed concentration is imposed at the border nodes. Third, diffusion proceeds further down after the border.

3.3.1. Different boundary conditions

Blocked diffusion

With this condition, we consider that there is no heat flux at the border Σ_φ. Thus,

$$\int_{\Sigma_\varphi \cap S_e} \mathbf{N^e} \cdot q_n \cdot dL = 0$$

and $\mathbf{F_e}$ is limited to the term:

$$\mathbf{F_e} = \left[\int_{S_e} \mathbf{N^e} \cdot P \cdot dS\right]^{\mathrm{T}}$$

In case of a blocked diffusion,

Fixed temperature

Fixed temperature boundary condition intervene only on Σ_T boundary which in not involved in the computation of $\mathbf{F_e}$**.** Thus,

$$\mathbf{F_e} = \left[\int_{S_e} \mathbf{N^e} \cdot P \cdot dS\right]^{\mathrm{T}}$$

Again, border nodes with fixed temperature condition are treated like every other nodes.

Modeling of further diffusion

We imagine here that heat diffusion proceeds further beyond the border of the lattice up to a distance to which temperature is equal to the ambient temperature. We propose to model this continued diffusion by adding a resistor to each border node (see the red resistor on Fig. 5).

This additional resistor represent the flux outing the lattice because of diffusion since a resistor creates a flux of heat. Moreover, this flux is proportional to the potential of the nodes to which it is connected. That is why the other terminal of the resistors are connected to the thermal ground.

By analogy, we would have in the heat diffusion model:

$$q_n = \varphi_X \cdot T(x, y)$$

We can therefore calculate the first integral of $\mathbf{F_e}$ expression (named $\mathbf{Q_n^e}$). Assuming that φ_X is constant,

$$\mathbf{Q_n^e} = \left[\int_{\Sigma_\varphi \cap S_e} \mathbf{N^e} \cdot q_n \cdot dL\right]^{\mathrm{T}} = \left[\int_{\Sigma_\varphi \cap S_e} \mathbf{N^e} \cdot \varphi_X \cdot dL\right]^{\mathrm{T}} = \varphi_X \cdot \left[\int_{\Sigma_\varphi \cap S_e} \mathbf{N^e} \cdot dL\right]^{\mathrm{T}}$$

The expression of $\mathbf{Q_n^e}$ is a bit different depending on the position of mesh edge that belongs to the boundary.

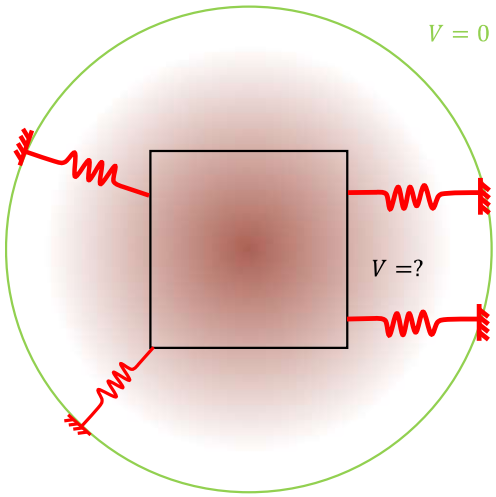


Fig. 5 Representation of the model of heat diffusion treatment beyond the lattice. The black square represent the lattice. The green border shows a limit after which temperature can be considered as fixed (ambient temperature). In red, the resistor used to model the continued diffusion after the lattice.

Vertical border

We imagine an element having an edge at $x = 0$ or $x = L$ (see Fig. 6). In this case,

$$\mathbf{Q_n^e} = \varphi_X \cdot \left[\int_O^W \mathbf{N^e} \cdot dy\right]^{\mathrm{T}}$$

Thus

$$\mathbf{Q_n^e}(x) = \varphi_X \cdot \frac{L}{6 \cdot W^2} \cdot \begin{pmatrix} (L-x)\cdot\left(W\cdot(2\cdot T_1 + T_4) + y\cdot(-2\cdot T_1 + 2\cdot T_2 + T_3 - T_4)\right) \\ x\cdot\left(W\cdot(2\cdot T_1 + T_4) + y\cdot(-2\cdot T_1 + 2T_2 + T_3 - T_4)\right) \\ x\cdot\left(W\cdot(T_1 + 2\cdot T_4) + y\cdot(-T_1 + T_2 + 2\cdot T_3 - 2T_4)\right) \\ (L-x)\cdot\left(W\cdot(T_1 + 2\cdot T_4) + y\cdot(-T_1 + T_2 + 2\cdot T_3 - 2T_4)\right) \end{pmatrix}$$

$\mathbf{Q_n^e}(0)$ correspond to the expression of $\mathbf{Q_n^e}$ at the bottom border and $\mathbf{Q_n^e}(W)$ at the top border. For $x = 0$:

$$\boldsymbol{Q_n^e} = -\frac{1}{R}\frac{W}{6}\begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}\begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$$

And for $x = L$:

$$\boldsymbol{Q_n^e} = -\frac{1}{R}\frac{W}{6}\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$$

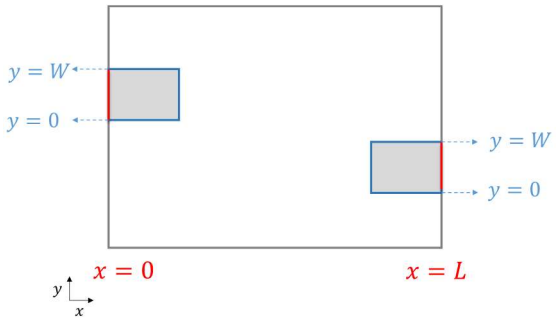


Fig. 6 Element with an edge on a vertical border. X and y coordinates are given in the element frame of reference.

Horizontal border

We repeat the computation for an element having an edge at $y = 0$ or $y = W$ (see Fig. 7). In this case:

$$\mathbf{Q_n^e} = \varphi_X \cdot \left[\int_O^L \mathbf{N^e} \cdot dx\right]^{\mathrm{T}}$$

Thus

$$\mathbf{Q_n^e}(x) = \varphi_X \cdot \frac{W}{6 \cdot L^2} \cdot \begin{pmatrix} y \cdot \left(L \cdot (2 \cdot T_1 + T_2) + x \cdot (-2 \cdot T_1 - T_2 + T_3 + 2 \cdot T_4)\right) \\ (W - y) \cdot \left(L \cdot (2 \cdot T_1 + T_2) + x \cdot (-2 \cdot T_1 - T_2 + T_3 + 2T_4)\right) \\ (W - y) \cdot \left(L \cdot (T_1 + 2 \cdot T_2) + x \cdot (-T_1 - 2T_2 + 2 \cdot T_3 + T_4)\right) \\ y \cdot \left(L \cdot (T_1 + 2 \cdot T_2) + x \cdot (-T_1 - 2T_2 + 2 \cdot T_3 + T_4)\right) \end{pmatrix}$$

$\mathbf{Q_n^e}(0)$ correspond to the expression of $\mathbf{Q_n^e}$ at the left border and $\mathbf{Q_n^e}(L)$ at the right border.

$$\boldsymbol{Q_n^e} = -\frac{1}{R}\frac{L}{6}\begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$$

If $y = W$, we obtain:

$$\boldsymbol{Q_n^e} = -\frac{1}{R}\frac{L}{6}\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}\begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$$

Corner element

The elements located at a corner of a lattice are treated in a similar way, except that two integrals are calculated and added: one for the horizontal border at a given y and one for the vertical border at a given x.

For the upper left corner we obtain:

$$\boldsymbol{Q_n^e} = -\frac{1}{6R}\left(W\begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix} + L\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}\right)\begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$$

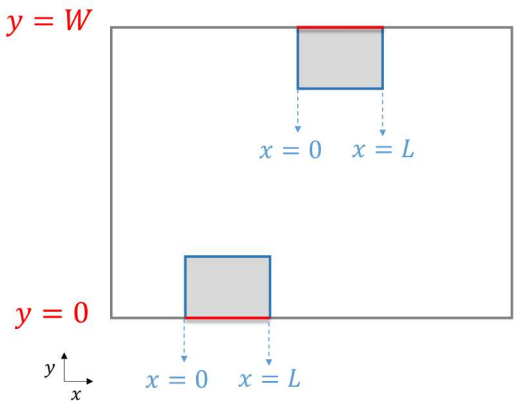


Fig. 7 Element with an edge on a horizontal border. X and y coordinates are given in the element frame of reference.

For the upper right corner we obtain:

$$\boldsymbol{Q_n^e} = -\frac{1}{6R}\left(W\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + L\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}\right)\begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$$

For the lower left corner we obtain:

$$\boldsymbol{Q_n^e} = -\frac{1}{6R}\left(W\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + L\begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\right)\begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$$

And for the lower right corner we obtain:

$$\boldsymbol{Q_n^e} = -\frac{1}{6R}\left(W\begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix} + L\begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\right)\begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$$

3.3.2. Implementation

In conclusion, these fluxes are to be represented by VCCS depending on the potential of the node located on the border. The relative contribution has been establish. As for the absolute value of these fluxes ($\frac{1}{R}$), it corresponds to the diffusion coefficient D of the molecule (as define in the model described in Chapter 6).

Appendix II
Verilog-A Model of the Elementary Mesh

1. Finite differences

```
////////////////////////////////////////
// 2D/3D Multi-level Biological Simulator
//
// Model: 2D mesh (square)
// Version: 8.0
// Date: 29/06/2015
// Author: Elise Rosati
// Last Revision: minor
//   default n = 0


`include "disciplines.vams"

module Mesh2D(n0,n1,n2,n3,n01,n12,n23,n30,nref);
   inout n0,n1,n2,n3;                          // Corners
   inout n01,n12,n23,n30;                      // Middles of edges
   inout nref;                                 // Reference
   electrical n0,n1,n2,n3,n01,n12,n23,n30,nref;

   ///////////////////////
   // MODEL PARAMETERS //
   ///////////////////////

   parameter integer ID = 0;                  // Mesh ID

   parameter real D    = 2;                    // Diffusion constant
   parameter real K    = 1;                    // Mesh capacitor
   parameter real dx   = 1;                    // Degradation constant
   parameter real MeshSize = 1;                // Initial Mesh Size

   parameter integer n = 0;                    // Mesh refinement factor

   parameter integer X01 = 0;                  // Connection of node 0-1
   parameter integer X12 = 0;                  // Connection of node 1-2
   parameter integer X23 = 0;                  // Connection of node 2-3
   parameter integer X30 = 0;                  // Connection of node 3-0




   ////////////////////
```

```
    // MODEL SIGNALS //
    ///////////////////

    real Fx1;                       // x-axis flux from 1 to 0
    real Fx2;                       // x-axis flux from 2 to 3
    real Fy1;                       // y-axis flux from 3 to 0
    real Fy2;                       // y-axis flux from 2 to 1

    real Dn;                        // Meshsize-dependent diffusion coefficient
    real Kn;                        // Meshsize-dependent condensator
    real Rn;                        // Meshsize-dependent resistor

analog begin

    ////////////////////////////////////////////////
    // INTERNAL K, R, D COMPUTATION (INITIAL STEP) //
    ////////////////////////////////////////////////

    Dn = D*pow(2,n) / pow(MeshSize,2);
    Rn = 1/(dx*(0.25* pow(0.5,n)));
    Kn = 0.25*K * pow(0.5,n);


    ////////////////////////
    // FLUX COMPUTATION //
    ////////////////////////

    Fx1 = Dn * (V(n1,nref) - V(n0,nref));
    Fx2 = Dn * (V(n2,nref) - V(n3,nref));
    Fy1 = Dn * (V(n3,nref) - V(n0,nref));
    Fy2 = Dn * (V(n2,nref) - V(n1,nref));


    ////////////////////////
    // FLUX DISTRIBUTION //
    ////////////////////////

    I(nref,n0 ) <+ Fy1/2;
    I(nref,n1 ) <+ Fy2/2;
    if (X01) begin
       I(nref,n01) <+ Fy1/2 +  Fy2/2;
    end

    I(nref,n1 ) <+ - Fx1/2;
    I(nref,n2 ) <+ - Fx2/2;
    if (X12) begin
       I(nref,n12) <+ - Fx1/2 - Fx2/2;
    end

    I(nref,n2 ) <+ - Fy2/2;
    I(nref,n3 ) <+ - Fy1/2;
    if (X23) begin
       I(nref,n23) <+ - Fy1/2 - Fy2/2;
     end

    I(nref,n3 ) <+ + Fx2/2;
    I(nref,n0 ) <+ + Fx1/2;
    if (X30) begin
       I(nref,n30) <+ + Fx1/2 + Fx2/2;
    end
```

```
    ////////////////////////////////////////////////////
    // CURRENT NODE COMPUTATION (CAPACITOR + RESISTOR) //
    ////////////////////////////////////////////////////

    // Corners
    I(n0 ,nref) <+ + (ddt(V(n0 ,nref))*Kn  + V(n0 ,nref)/Rn);
    I(n1 ,nref) <+ + (ddt(V(n1 ,nref))*Kn  + V(n1 ,nref)/Rn);
    I(n2 ,nref) <+ + (ddt(V(n2 ,nref))*Kn  + V(n2 ,nref)/Rn);
    I(n3 ,nref) <+ + (ddt(V(n3 ,nref))*Kn  + V(n3 ,nref)/Rn);

    // Middle of edges
    I(n01,nref) <+ + (ddt(V(n01,nref))*Kn + V(n01,nref)/Rn);
    I(n12,nref) <+ + (ddt(V(n12,nref))*Kn + V(n12,nref)/Rn);
    I(n23,nref) <+ + (ddt(V(n23,nref))*Kn + V(n23,nref)/Rn);
    I(n30,nref) <+ + (ddt(V(n30,nref))*Kn + V(n30,nref)/Rn);

  end
endmodule
```

2. Finite element with triangular composition

The model is pin-to-pin compatible with Finite Difference model and has the same parameters. The only differences between both models are:

- The definition of an internal node:

```
electrical nc;        // central node
```

- The computation of flux (`Fx1`, `Fx1`, `Fx1`, `Fx1`) are no longer defined and fluxes at nodes are computed directly from concentration (*i.e.* voltages) and `X01`, `X12`, `X23`, `X30` parameters:

```
I(n0,n01) <+ 0.5 * X01 * D * V(n0,n01);
I(n0,n30) <+ 0.5 * X30 * D * V(n0,n30);
I(n0,nc) <+ 0.5 * D * (2 - X01 - X30) * V(n0,nc);
I(n1,n12) <+ 0.5 * X12 * D * V(n1,n12);
I(n1,n01) <+ 0.5 * X01 * D * V(n1,n01);
I(n1,nc) <+ 0.5 * D * (2 - X12 - X01) * V(n1,nc);
I(n2,n23) <+ 0.5 * X23 * D * V(n2,n23);
I(n2,n12) <+ 0.5 * X12 * D * V(n2,n12);
I(n2,nc) <+ 0.5 * D * (2 - X23 - X12) * V(n2,nc);
I(n3,n30) <+ 0.5 * X30 * D * V(n3,n30);
I(n3,n23) <+ 0.5 * X23 * D * V(n3,n23);
I(n3,nc) <+ 0.5 * D * (2 - X30 - X23) * V(n3,nc);
```

- Degradation are now computed by additional Voltage Controlled Current Source that replaces the resistor in current node computation:

```
// Corner nodes
I(nref, n0) <+ cfdeg *((4 - (X01+X30)) * V(n0, nref) + (1 - X01) *
      V(n1, nref) + (1 - X30) * V(n3, nref) + (X01/2) * V(n01, nref)
      + (X30/2) * V(n30, nref) + (2 - (X01 + X30)/2) * V(nc, nref));
I(nref, n1) <+ cfdeg *((4 - (X12+X01)) * V(n1, nref) + (1 - X12) *
      V(n2, nref) + (1 - X01) * V(n0, nref) + (X12/2) * V(n12, nref)
      + (X01/2) * V(n01, nref) + (2 - (X12 + X01)/2) * V(nc, nref));
```

```
I(nref, n2) <+ cfdeg *((4 - (X23+X12)) * V(n2, nref) + (1 - X23) *
      V(n3, nref) + (1 - X12) * V(n1, nref) + (X23/2) * V(n23, nref)
      + (X12/2) * V(n12, nref) + (2 - (X23 + X12)/2) * V(nc, nref));
I(nref, n3) <+ cfdeg *((4 - (X30+X23)) * V(n3, nref) + (1 - X30) *
      V(n0, nref) + (1 - X23) * V(n2, nref) + (X30/2) * V(n30, nref)
      + (X23/2) * V(n23, nref) + (2 - (X30 + X23)/2) * V(nc, nref));

// Segment nodes
I(nref, n01) <+ cfdeg * X01 * (2 * V(n01, nref) + V(n0, nref)/2 +
      V(n1, nref)/2 + V(nc, nref));
I(nref, n12) <+ cfdeg * X12 * (2 * V(n12, nref) + V(n1, nref)/2 +
      V(n2, nref)/2 + V(nc, nref));
I(nref, n23) <+ cfdeg * X23 * (2 * V(n23, nref) + V(n2, nref)/2 +
      V(n3, nref)/2 + V(nc, nref));
I(nref, n30) <+ cfdeg * X30 * (2 * V(n30, nref) + V(n3, nref)/2 +
      V(n0, nref)/2 + V(nc, nref));

// Central node
I(nref, nc) <+ cfdeg * (8 * V(nc, nref) + (2 - (X01 + X30)/2) * V(n0,
      nref) + (2 - (X01 + X23)/2) * V(n0, nref) + X01 * V(n01, nref)
      + (2 - (X12 + X23)/2) * V(n1, nref) + X12 * V(n12, nref) + (2 -
      (X23 + X23)/2) * V(n2, nref) + X23 * V(n23, nref) + (2 - (X30 +
      X23)/2) * V(n3, nref) + X30 * V(n30, nref)) ;
```

Résumé de thèse

Outils d'aide à la conception pour l'ingénierie de systèmes biologiques

Ce présent résumé montre une vue d'ensemble du travail de thèse réalisé. Les résultats n'y sont pas présentés, nous invitons le lecteur à consulter le manuscrit de thèse pour plus de détails.

1. Introduction générale

La biologie synthétique est un nouveau domaine d'investigation qui a vu le jour au début du XXI^{ième} siècle. La recherche en biologie synthétique demande d'avoir des connaissances générales dans plusieurs disciplines et est souvent définie comme l'application des principes d'ingénierie à la biologie (http://www.synbioproject.org/topics/synbio101/definition/). Le terme de « biologie synthétique » est d'abord apparu sous la plume de Stéphane Leduc au XX^{ième} siècle, qui écrit que la biologie doit être « successivement descriptive, analytique et synthétique » (Le Duc 1910). Ce principe fait écho à l'idée que pour vraiment comprendre un mécanisme biologique, il faut être capable de le recréer. Si la description et l'analyse des organismes biologiques sont des disciplines bien établies, la synthèse, et donc la biologie synthétique, n'en est qu'à ses débuts.

Les mots « design » (conception), « ingénierie », « systèmes », « *parts* » (composant) et « fonctions » sont des mots auxquels la biologie synthétique est souvent associée. C'est particulièrement bien illustré par la définition établie par un groupe d'experts européens, la « Commission Européenne pour les biotechnologies, l'Agriculture et la Nourriture » :

> La biologie synthétique est l'ingénierie de la biologie : la synthèse de systèmes complexes basés (ou inspirés) de la biologie et qui exhibent des fonctions non présentes dans la nature. Cette perspective d'ingénierie peut être appliquée à tous les niveaux de la hiérarchie des structures biologiques – des molécules individuelles aux cellules, tissus et organismes tout entier. Dans son essence, la biologie synthétique va permettre le design rationnel et systématique de « systèmes biologiques ».

Les caractéristiques principales de la biologie synthétique sont résumées par le SynBERC (un programme de recherche américain), qui énonce les éléments marquants de la biologie synthétique ainsi ("What Is Synthetic Biology? - Synthetic Biology Project" 2017):

> Des dispositifs et *parts* prédictibles, libres, avec des connections standard, des châssis biologiques robustes (comme la levure et E. coli) qui acceptent facilement ces *parts* et dispositifs, des standards pour assembler des composants dans des systèmes fonctionnels de plus en plus sophistiqués, et le développement de *parts*, dispositifs et châssis open source.

Le champ d'application de la biologie synthétique est très large et inclue la thérapeutique, la création de biomatériaux innovants, de biocapteurs… Ce domaine de recherche montre plusieurs facettes. La principale concerne les biotechnologies qui permettent la modification de systèmes biologiques existants ou la création de systèmes biologiques artificiels. En parallèle, des efforts ont été faits pour développer les outils qui facilitent le processus de design de ces systèmes. C'est dans ce dernier contexte que s'inscrit cette thèse.

Ces dernières années, les systèmes conçus ont grandi en complexité et en taille. Le développement d'outils de conception assistée par ordinateur (CAO) efficaces est donc crucial. Partout dans le monde, plusieurs équipes de recherches travaillent au développement de tels outils, génériques ou *ad hoc*. Il y a quelques années, notre équipe a proposé une alternative qui consiste à adapter des outils existants, du domaine des sciences de l'ingénieur à la biologie synthétique. Cette approche a deux avantages : i) les outils de CAO des sciences de l'ingénieur sont fiables et ont déjà fait leurs preuves durant ces dernières décennies et ii) l'ajout de la biologie synthétique à ces outils de CAO pourrait aussi faciliter l'intégration de *parts* biologiques dans des systèmes transdisciplinaires de grande ampleur comme les laboratoires sur puce ou les biocapteurs.

Le point de départ de cette thèse est un environnement préexistant composé de différents outils qui forment un workflow complet, intégrant la spécification des systèmes biologiques à haut niveau d'abstraction jusqu'à sa réalisation pratique. Plusieurs briques de ce workflow ont déjà été développées par des étudiants de thèse ou de master auparavant, en particulier ceux liés à la modélisation et à la simulation de systèmes biologiques. Ma contribution à ce travail concerne deux outils manquants : l'amélioration du processus d'automatisation du design de réseaux de régulation génétique (RRG) pendant les premières phases de conception et le développement d'un outil de simulation pour les modèles dans lesquels la localisation spatiotemporelle joue un rôle important.

Pour le premier point, à l'instar des circuits électroniques, l'automatisation du design des RRGs est très dépendante de leur type et complexité. Pour les RRGs qui peuvent être décrits par une équation Booléenne, nous avons implémenté une solution qui consiste à réutiliser les synthétiseurs digitaux de la microélectronique. Néanmoins, cet outil doit être précisément réglé pour correctement incorporer les spécificités des RRGs qui diffèrent de celles des circuits électroniques. Les circuits digitaux sont composés de deux grandes familles : les circuits combinatoires et les circuits séquentiels. Le dseign de circuits séquentiels est souvent plus délicat à cause de leur boucle de retour interne qui peut causer des instabilités et des mauvais fonctionnements. Ce point est aussi abordé dans la thèse ainsi que les solutions que nous proposons.

D'un autre côté, pour les RRGs décrits par une fonction analogue (fonction de transfert, caractéristiques temporelles ou fréquentielles…) on ne peut plus employer le synthétiseur digital. Malgré une recherche active dans ce domaine, le design de ces circuits est la plupart du temps fait à la main. Dans cette thèse, on évalue le potentiel d'algorithmes inspirés de la nature pour l'automatisation du design de ces RRGs. En particulier, deux types d'algorithmes sont implémentés : des algorithmes évolutionnaires qui peuvent être utilisés pour optimiser les paramètres d'un modèle pour correspondre à une réponse cible, et des algorithmes génétiques qui font évoluer à la fois les paramètres et le modèle en lui-même pour parvenir à une réponse définie *a priori*.

L'autre contribution majeure au workflow concerne le développement d'un outil de simulation pour prendre en charge les modèles qui incluent aussi bien des phénomènes locaux (des mécanismes biochimiques confinés à un espace donné, la diffusion à travers des membranes ou des murs) que des phénomènes globaux (la diffusion libre dans un milieu, la dégradation, etc). Ce type de système est de

plus en plus courant en biologie synthétique et en biologie des systèmes. Du point de vue de la modélisation, leur particularité est que ces systèmes ne sont plus modélisés par des équations différentielles ordinaires mais par des équations aux dérivées partielles. La résolution de ces équations requiert des algorithmes assez sophistiqués de discrétisation de l'espace et de résolution d'importants ensembles d'équations. Pour pallier à ce problème, on tire inspiration d'un problème similaire rencontré dans le design des circuits intégrés en microélectronique : la simulation életrothermale. Dans notre équipe, nous disposons d'un outil développé pour un projet précédent. Nous avons cherché à adapter cet outil au contexte biologique. Dans la thèse, nous présentons cette approche ainsi que les résultats obtenus sur de véritables systèmes biologiques.

Dans ce résumé, nous présentons tout d'abord le contexte et les objectifs de la thèse, puis nous abordons les résultats de la partie 2 de la thèse concernant l'automatisation du design de RRGs. Enfin nous présentons les principaux résultats de la partie 3 qui est dédiée à l'outil de simulation spatiotemporelle.

2. Contexte

2.1. Concepts utilisés dans cette thèse

Comme souvent en biologie synthétique, ce travail de thèse est à l'interface de plusieurs disciplines. Des concepts fondateurs en microélectronique et en biologie sont donc présentés dans le chapitre 2 de la thèse. Il s'agit notamment des modèles des principaux composants de la microélectronique, comme la résistance, la capacité et le transistor. Ces modèles sont souvent utilisés avec les lois de Kirchhoff, la loi des mailles et la loi des nœuds. Du côté de la biologie, un rappel est fait sur le dogme classique de l'expression de l'information génétique. Les étapes de transcription de l'ADN en ARN et de traduction de l'ARN messager en protéine sont décrites. Les réseaux de régulations génétiques sont introduits, ainsi que la régulation par micro ARN.

Ces concepts clés permettent de mieux appréhender l'analogie entre l'électronique et la biologie dréssée en fin de chapitre. En effet, une analyse des modèles montre qu'en considérant la concentration de molécule comme étant une tension, on peut simuler l'expression d'un gène avec un transistor, qui joue le rôle de source de molécule, une résistance qui modélise la dégradation et une capacité qui permet de monitorer la variation temporelle de la concentration.

2.2. Le design flow en biologie synthétique

Comme indiqué dans l'introduction de ce résumé, la complexité des systèmes conçus en biologie synthétique est limitée par la technique d'une part et par les lacunes des outils de design d'autre part. Alors que le design est souvent fait à la main, le protoypage virtuel prend une place de plus en plus importante. Pourtant, la plupart des outils qui proposent cela sont souvent *ad hoc* et manque de généricité et de réutilisabilité. La standardisation du design et des outils de simulation ainsi que de tout le processus de création de nouveaux systèmes ou fonctions biologiques est devenu clé dans la progression vers des systèmes plus compliqués. Pour cela, on va cherche l'inspiration en

microélectronique. En effet, ce domaine a derrière lui des années d'expérience dans le design de systèmes et plutôt que de redéveloper les outils depuis le début, il peut être intéressant en termes de temps d'adapter les outils et le savoir-faire de la microélectronique.

A ces fins, une introduction aux différentes approches de design utilisées en biologie synthétique et dans d'autres domaines de la physique est faite, avec un focus particulier sur la microélectronique. Un état de l'art sur les outils existant pour le prototypage virtuel (modélisation et simulation) et sur l'automatisation du design est ensuite présenté.

2.2.1. Deux approches : top-down et bottom-up

2.2.1.1 Bottom-up

L'approche bottom-up consiste à construire des systèmes en assemblant des composants standard bien caractérisés, souvent répertoriés dans des librairies de composants, en des sous-systèmes eux-mêmes assemblés pour former le système final (cf Figure 1).

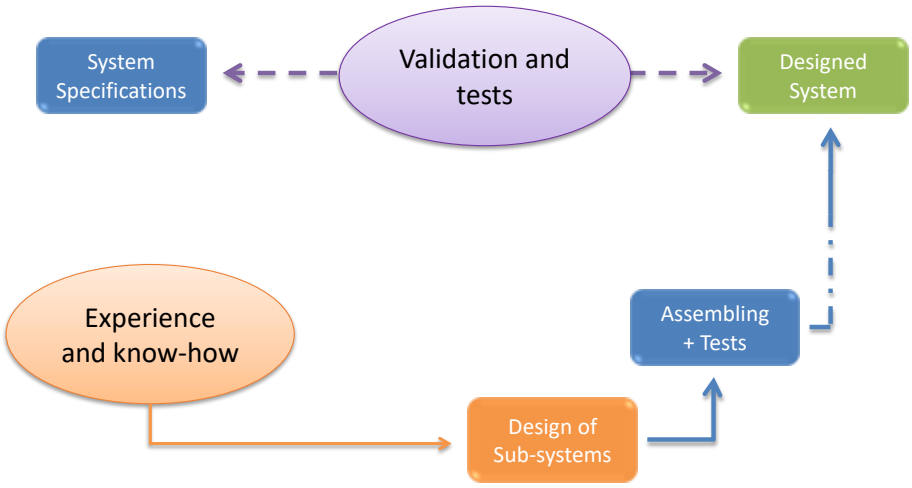


Figure 1 L'approche bottom-up

2.2.1.2 Top-down

L'approche top-down consiste à décomposer hierarchiquement le système attendu en sous-fonctions (cf Figure 2). A chaque étape de la decomposition, on établit la specification de chaque sous-fonction et des intarfaces entre elles. Une vérification de la cohérence entre ces sous-fonctions est faite par la modélisation et la simulation.

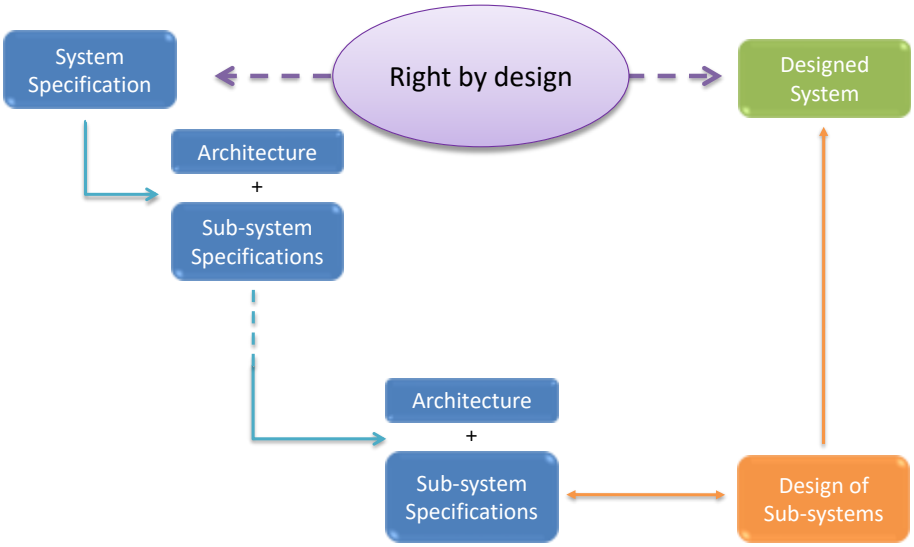


Figure 2 L'approche top-down

2.2.1.3 En biologie synthétique

Les termes "top-down" et "bottom-up" sont souvent rencontrés en biologie synthétique. Ils sont liés à la manière de construire un nouvel organisme artificiel plutôt qu'à la méthodologie de desgin de la nouvelle fonction artificielle en elle-même. Pour construire une cellule minimale via une approche bottom-up, les biologistes auraient besoin de sélectionner des composants non vivant (acides aminés, gènes…), de les assembler en sous-systèmes (des enzymes qui interagissent avec d'autres composants) qui génèrent une cellule fonctionnelle une fois combinés. Avec l'approche top-down, le biologiste part d'une cellule vivante et la modifie en lui retirant différentes fonctions, et ce tant que la cellule reste viable.

2.2.2. Le design flow utilisé pour la biologie synthétique

Le design flow imaginé par Gendrault et al. pour la biologie synthétique (Gendrault, Madec, Lallement, et al. 2014) est repris dans cette thèse et présenté en Figure 3. L'entrée est un ensemble de spécifications et la sortie une liste des gènes/cellules qui composent le RRG. Ce design flow pourrait etre implémenté dans un software unique qui prendrait en charge toutes ces étapes, mais on peut aussi envisager d'adapter plusieurs outils différents séparément. Ces outils séparés échangent ensuite des données dans un format compatible.

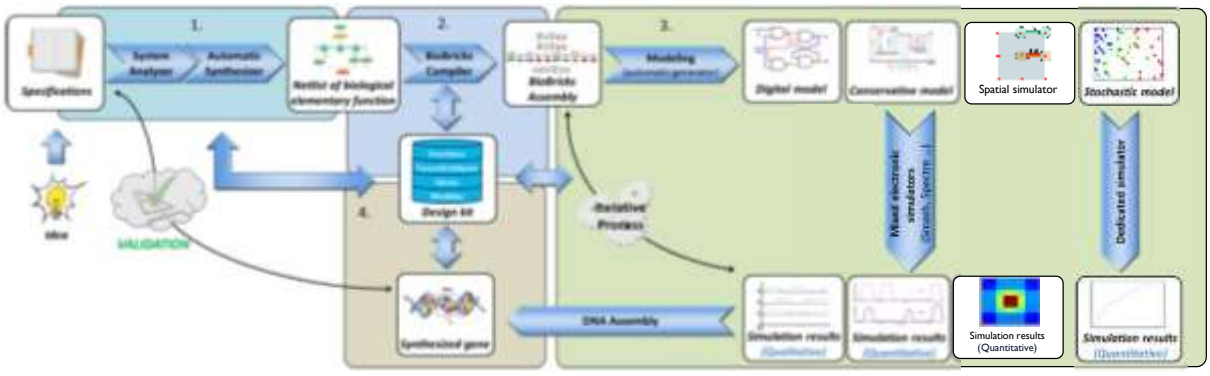


Figure 3 Un desgin flow pour la biologie synthétique inspiré de la microélectronique (Gendrault, Madec, Lallement, et al. 2014)

Le premier bloc est l'analyseur système haut-niveau. Le but de cette étape est de trouver la topology du système à concevoir (par exemple un RRG) en accord avec la spécification donnée en entrée. Cette spécification peut être une équation Booléenne, une fonction de transfer ou une évolution temporelle du système après un stimulus donné.

Le second bloc sélectionne les composants biologiques appropriés dans une database pour correspondre aux fonctions précédemment sélectionnées. Ces composants sont des Biobricks (séquences d'ADN) ou des *parts*. Une database de ces composants existe ("Registry of Standard Biological Parts" 2015) , mais chaque composant n'est pas décrit de manière standard, comme en microélectronique.

Le troisième bloc est constitué du prototypage virtuel (modélisation et simulation) de l'assemblage des Biobricks. Selon le niveau de complexité du système, le degré de précision requis et le temps de simulation alloué, plusieurs niveaux d'abstraction peuvent être envisagés : le haut-niveau qui est l'abstraction digitale (les composants sont vus comme des portes bio-logiques) et le bas niveau, plus précis, qui est l'abstraction digitale et qui modélise les composants par des équations différentielles ordinaires (EDO).

Le quatrième bloc correspond à la fonte des circuits en microélectronique. Avec cette étape, un véritable système est créé. Ce système peut être testé à la paillasse et selon les résultats, une nouvelle boucle de design/optimisation du système peut être lancée.

2.3. Niveaux d'abstraction

Comme mentionné, le troisième bloc est composé d'outils capable de simuler le comportement d'un RRG. Selon les exigences (quel type de fonction doit être réalisé ? Avec quelle précision le RRG doit-il correspondre aux exigences ?), différents types de modèles peuvent être utilisés.

2.3.1. Abstraction digitale

Au plus haut niveau d'abstraction, un gène peut être considéré comme ayant 2 états : un état OFF dans lequel sa protéine associée n'est pas exprimée, et un état ON correspondant au contraire. Dans la réalité, on définit des seuils de concentration de protéine : un sous lequel le gène est OFF (la protéine est absente) et un autre, plus élevé, au-delà duquel le gène est ON (la protéine est présente).

Avec cette approche, il est possible d'obtenir les portes logiques classiques de la microélectronique (cf Figure 4).

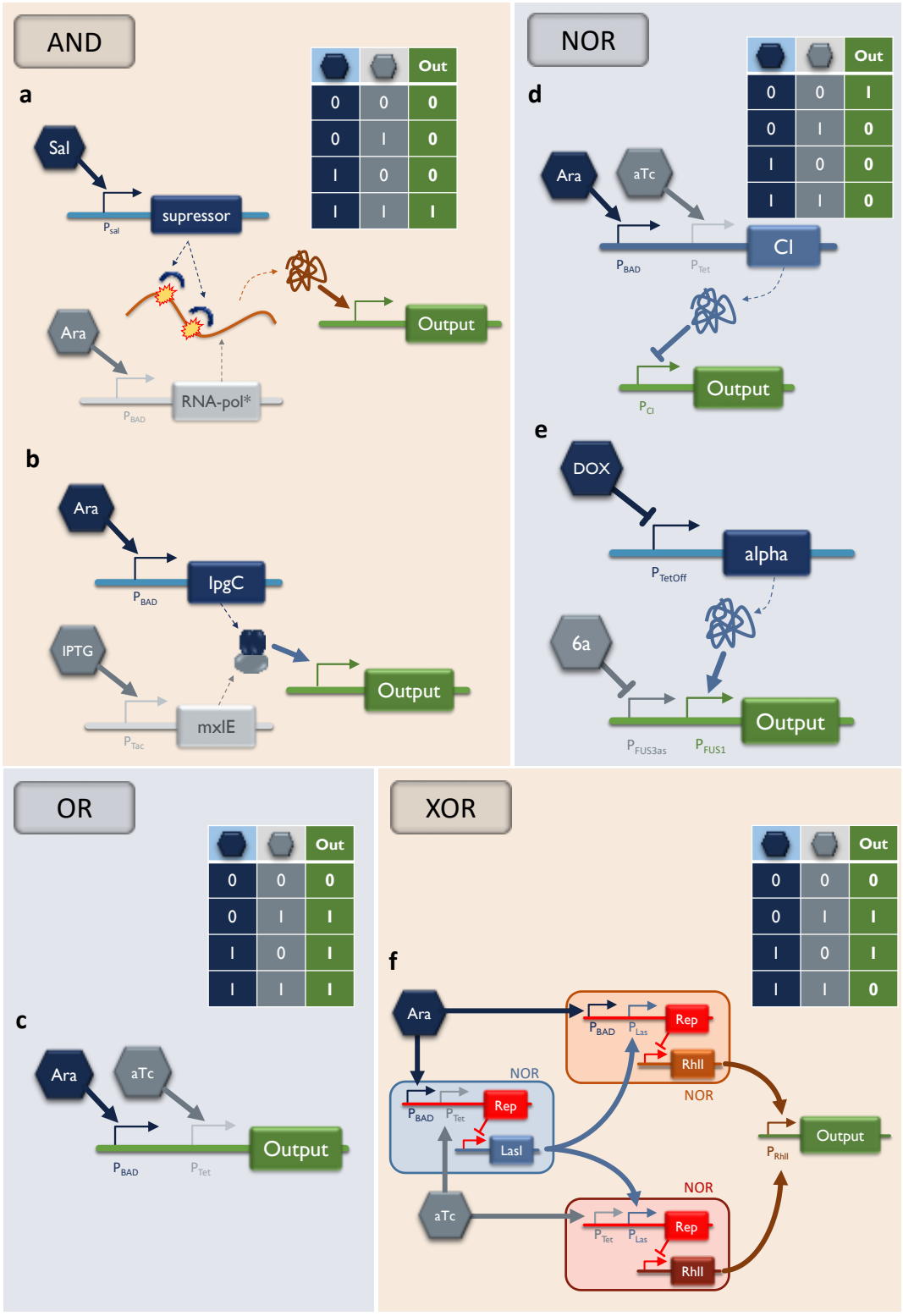


Figure 4 Implémentation des portes logiques classique en RRG : portes AND, NOR, OR et XOR. RRG a est basé sur une séquence corrompue de la T7-polymérase (Anderson, Voigt, and Arkin 2007). RRG b code pour un régulateur qui est activé en presence de sa preotéine chaperone IpgC (Moon et al. 2012). RRG c utilize un promoteur active par ou l'arabinose (Ara) ou l' anhydrotetracycline (aTc) (Tamsir, Tabor, and Voigt 2011). RRG d est une extension de RRG c: Le premier gène code pour un répresseur (CI) qui inhibe l'expression de la sortie. En son absence, la sortie est produite (Tamsir, Tabor, and Voigt 2011). Le RRG e utilize 2 inhibtions (modifié de (Regot et al. 2011)). Le RRG f est compose de 3 portes NOR similaires (cf RRG d) séparées sur 4 cellules "connectées". En effet, elles utilisent les systèmes de communication intercellulaires de *Pseudomonas aeruginosa* (LasI et RhlI) pour faire passer le signal d'une porte NOR à l'autre.

2.3.2. Equations différentielles ordinaires

L'abstraction digitale ne permet de manipuler que des concentrations quantifiées. Pour avoir des concentrations qui varient de manière continuent, il faut utiliser un modèle d'équations différentielles ordinaires. Ce niveau de description est plus précis. En biologie, la formalisation de ce genre de description est souvent réalisée via un fichier SBML (Systems Biology Markup Language). SBML est un langage introduit en 2003 (Hucka et al. 2003) et est composé d'une liste conséquente de balises correspondant aux différents éléments d'un modèle biologique. Par exemple, le SBML permet la définition d'unités, de compartiments où les différentes réactions peuvent avoir lieu, d'espèces impliquées dans ces réactions, de paramètres de ces réactions, d'évènements à appliquer pendant la simulation du modèle…

2.4. Simulateurs

Différents simulateurs existent pour la biologie. L'un des plus utilisés est COPASI. Il permet la manipulation de fichiers SBML, a une interface graphique et propose plusieurs types de simulation. COPASI permet la spécification de compartiments de différentes tailles mais ne propose pas de simulation spatiotemporelle.

BioCham est aussi capable de lire un fichier SBML ainsi que des EDO écrites dans un format spécifique. BioCham propose des fonctionnalités similaires à COPASI mais n'a pas d'interface graphique. En particulier, BioCham propose un ensemble fourni d'analyses qualitatives : le RRG est analysé au niveau booléen. La simulation spatiale n'est pas disponible avec BioCham.

Virtual Cell est aussi un outil majeur pour la simulation des systèmes biologiques. Il dispose d'une interface graphique et propose un ensemble complet d'analyses. Cet outil a un simulateur spatial, qui sera repris dans la partie résumant le chapitre 6.

Pour autant, aucun de ces outils n'est assez robuste pour supporter de très grands nombres d'équations. C'est pourquoi un formalisme basé sur l'électronique et la biologie a été dévelopé par Madec et al. pour simuler des RRGs avec des simulateurs de l'électronique (Madec, Lallement, and Haiech 2017). BB-SPICE est capable de lire un fichier SBML et de produire un fichier près à la simulation avec SPICE. Il a été démontré que BB-SPICE peut gérer des modèles de RRGs avec plus de 10000 réactions tandis que COPASI plante à cause de manque de mémoire avec seulement 1000 réactions. C'est sur ce formalisme que s'appuie notre outil, décrit dans le chapitre 6 de la thèse.

2.5. Design de RRG

Plusieurs outils de CAO existent pour la biologie synthétique. Ils facilitent la manipulation des modèles biologiques et des données associées. L'automatisation du desgin est le prochain point clé à atteindre : l'outil serait alors capable de simuler mais aussi créer le RRG en s'appuyant sur un cahier des charges donné.

2.5.1. Outil d'assistance au design

Plusieurs outils existent pour la biologie synthétique : TinkerCell (Chandran, Bergmann, and Sauro 2009), CellDesigner (Funahashi et al. 2003), BioJADE (Goler 2004), ProMot (Mirschel et al. 2009),

GenoCAD (Czar, Cai, and Peccoud 2009). La plupart d'entre eux s'appuie sur le "design par *parts*" rendu possible par le standard BioBricks. Ces outils ont souvent une interface graphique pour une manipulation facilitée des *parts*, ainsi facilitant le processus de design des RRGs.

Le Tableau 1 résume les différentes fonctionnalités de ces outils.

Tableau 1 Résumé des outils de CAO pour la biologie

	Input	*Output*	*DNA sequence support*	*Possibility to implement custom parts/functions*	*Automated design*
TinkerCell	Graphical interface, DNA sequence	SBML, Octave, Matlab	yes	yes	no
BioJADE			yes		
ProMoT	SBML	SBML, Matlab	no		
GenoCAD	Graphical interface	SBML	yes	yes	no

Les outils étudiés ici ne représentent qu'une fraction de tous les outils existant. Leur étude nous permet de spécifier les fonctionnalités que devrait avoir l'outil idéal. Ainsi, la plupart de ces outils peut gérer des séquences ADN, une fonctionnalité utile pour une implémentation directe à la paillasse. Certains outils (principalement BioJADE et GenoCAD) sont connectés à une database de *parts*. Le retrait automatique de *parts* nouvellement ajoutées à la database n'a pas été observé cependant. La possibilité d'ajouter des *parts* personnalisées est aussi intéressante, étant donné que les constructions biologiques évoluent constemment et de nouvelles construction aparaissent. Comme attendu, le langage SBML est un point commun à tous les formats utilisés pour échanger des données. Il est aussi important de mentionner le dépôt de modèles informatiques de processus biologiques BioModels, une initiative de l'Institut Européen pour la Bioinformatique (Juty et al. 2015). Les modèles y sont écrits en SBML. Dans cette database, plus de 7000 modèles publiés sont disponibles dans un formatage standard.

2.5.2. Automatisation du design

Avec les outils mentionnés précédemment, l'approche traditionnelle d'essai-erreur « à la main » pour l'optimisation de systèmes est améliorée. La prochaine étape reste l'automatisation de ce processus. Pour une spécification donnée, un outil de design automatique trouve le système optimal répondant à cette spécification. Une réponse à ce besoin est décrite en chapitre 4 et 5 de la thèse, dans la seconde partie du manuscrit.

3. Partie 2 – Automatisation du design de systèmes biologioques

Comme mentionné dans la section précédente, on constante un manque en outils de design automatisé de système biologiques. Cependant, la sélection automatique de *parts* à partir d'une spécification haut-niveau a connu plusieurs percées, toutes limitées à l'abstraction Booléenne des RRGs. Dévelopé par le MIT, TASBE (Beal et al. 2012) est l'un de ces outils. TASBE consiste en une suite d'outils (qui comprend plusieurs outils nouveaux ou préexistants comme Proto, BioCompiler, MAtchMaker et BioCAD) qui part d'une specification haut-niveau donnée dans un langage spécifique et produit un assemblage AND qui répond à la specification. Les premières étapes de TASBE incluent une description haut-niveau et une "bio-compilation" (la description haut-niveau est interprétée comme un ensemble de fonctions élémentaires biologiques) (Densmore et al. 2010; Bilitchenko et al. 2011; Beal, Lu, and Weiss 2011; Yaman et al. 2012). Une autre approche intéressante, elle aussi basée sur une abstraction Booléenne des RRGs, a été proposée par Marchisio *et al.* (Marchisio and Stelling 2011).

Il est aussi fait mention dans la première section que plusieurs RRGs synthétiques peuvent être décrits par une équation Booléenne. Dans ce cas, les problèmes que l'on rencontre pour leur design sont très similaires aux problèmes rencontrés lors du design digital en microélectronique. Pour bénéficier des années d'expérience de la microélectronique dans ce problème, on réutilise et adapte leurs outils à notre problématique. C'est ce qui est décrit plus précisément dans le chapitre 4 de la thèse. Tout d'abord, la suite d'outils GeNeDA y est présentée. GeNeDA est adapté de l'électronique et permet le design automatique de RRGs décrits au niveau digital. Dans un premier temps, GeNeDA est testé et validé sur des circuits combinatoires. Dans un second temps, le design de fonctions séquentielles est abordé. La robustesse des systèmes conçus envers la variation de ses paramètres biochimiques est notamment testée. Comme pour la microélectronique, la réalisation de circuits synchrones est souvent nécessaire pour éviter les risques de malfonctionnements et d'instabilité. Pour ce faire, il est nécessaire de pouvoir utiliser une D-flip-flop biologique; ce travail-là est aussi présenté dans le chapitre 4 de la thèse.

Le second chapitre de cette partie (chapitre 5) est consacré à l'automatisation du design de systèmes biologiques qui ne peuvent être représentés par une fonction Booléenne. Cette problématique nous amène dans le domaine de la synthèse analogue, un champ d'investigation pour lequel l'électronique a proposé plusieurs solutions mais auncune qui n'ait été capable de s'imposer comme standard. Dans le passé, plusieurs algorithmes et méthodes d'optimisation ont été testés. Parmi eux, des algortihmes inspirés de la nature (et en particulier les algorithmes évolutionnaires) ont montré des résultats prometteurs (Koza et al. 1997). L'adaptation de ces algorithmes à la biologie fait l'objet du chapitre 5. Dans un premier temps on fixe la topologie du RRG a priori et on fait évoluer les paramètres du modèle dans le but d'obtenir le comportement cible. Ensuite, on utilise la programmation génétique pour faire évoluer à la fois le réseau et ses paramètres, dans le but d'obtenir un véritable algorithme de design automatisé.

3.1. Conclusion obtenues sur le design au niveau digital

Les résultats du chapitre 5 de la thèse ouvrent la voie vers l'automatisation du design de RRGs. On a démontré la possiblité de faire du design automatique de RRG combinatoires en utilisant des outils issus de la microélectronique. A partir d'une spécification haut-niveau (table de vérité, fichier Verilog), l'outil conçoit un RRG et produit les modèles SBML et SystemC-AMS associés. Une attention particulière a été portée à la construction de la librairie générique de *parts*, qui représente le pilier de ce software. Elle a été rendue le plus réaliste possible en tenant compte des possibilités offertes par la biologie synthétique et a été évaluée sur des circuits standard.

La question du desgin de RRGs séquentiels est aussi abordée. Pour les systèmes séquentiels asynchrones, la méthode d'Huffmann fournit 2 ensembles d'équations Booléennes, qui peuvent être fournies en entrée à GeNeDA pour obtenir le RRG équivalent. Pourtant, les résultats de simulations mettent au jour plusieurs déficiences majeures des RRGs asyncrhones. En effet, le délai introduit par l'activation/inhibition d'un gène lors des boucles de retour provoque des malfonctionnements et/ou instabilités.

Pour les circuits asyncrhones, une D-flip-flop est requise. S'appuyant sur l'électronique, Hoteit *et al.* ont dévelopé un RRG composé de nombreux gènes et promoteurs ayant le comportement d'une D-flip-flop (Hoteit, Kharma, and Varin 2012). Cependant, la fesabilité de leur circuit avec du véritable matériel biologique n'a pas été démontrée. Ainsi, nous proposons une D-flip-flop biologique encore plus compacte. La robustesse du circuit, qui implique des boucles de retour, a été validée. Parce que le synthétiseur digital de GeNeDA peut manipuler des fichiers Verilog (décrivant le système asynchrone) directement, l'ajout de la D-flip-flop à la libraire de *parts* permet le design de n'importe quel RRG Booléen, au moins d'un point de vue théorique.

En pratique, la réalisation de ces RRGs pourrait s'avérer plus délicate. Le premier problème rencontré concerne le nombre de *parts* requis pour construire un système synchrone. Un exemple est donnée dans (Madec et al. 2013). Un simple système synchrone à 3 états a été généré avec GeNeDA. Ce système est composé de 11 promoteurs et 2 D-flip-flop, ce qui compose un RRG assez important compte tenu du savoir-faire technique courrant. Un autre problème pourrait se présenter à propos du D-flip-flop en lui-même. En biologie synthétique, toutes les *parts* sont présentes dans le même compartiment, la cellule. Le couplage entre les gènes est par conséquent innévitable (il n'y a pas de fil pour connecter un composant à un autre, à l'instar de l'électronique). Ce couplage peut mener aux mêmes problèmes soulevés plus haut pour les systèmes asyncrhones.

Pour un système contenant de multiples D-flip-flop, une solution consiste à utiliser un ensemble différent de promoteurs et de régulateurs qui n'interagissent pas ensemble pour chaque instance de D-flip-flop. Cette solution pouvant se révéler ardue, une alternative à privilégier consiste à divisier le système en plusieurs sous-systèmes qui sont implémentés dans différentes populations de cellules. L'étude de tels systèmes est plus compliquée et requiert un simulateur capable de prendre en compte la localisation spatiale des éléments biologiques. Notamment, le problème d'un signal d'horloge synchrone distribué à chaque cellule est soulevé.

Ces derniers points sont discutés dans la troisième partie du manuscrit de thèse. Auparavant, il faut s'intéresser à la question de l'automatisation du design de RRGs qui ne peuvent être représentés à un niveau Booléen d'abstraction (typiquement, un système qui exhibe une réponse en forme de cloche). C'est le sujet du chapitre 5 de la thèse.

3.2. Automatisation du design au niveau analogue

Le chapitre 5 de la thèse se concentre sur le design automatique de systèmes biologiques qui sont décrits par un comportement analogue. Cette approche est motivée par deux choses. Tout d'abord, il existe plusieurs systèmes qui ne peuvent être décrits par une fonction Booléenne. Par exemple le passe-band de Basu *et al.* (Basu et al. 2005) dans lequel la protéine qui fait office de signal de sortie n'est synthétisée que pour un niveau intermédiaire de la concentration de la protéine en entrée. Ensuite, par opposition avec l'électronique, le fossé entre l'abstraction d'un RRG et son comportement effectif peut être assez grand. Les connections entre les gènes sont en particulier peu évidentes à régler. Prenons par exemple un gène #1 qui produit un activateur du gène #2. Même si le gène #1 est actif, la quantité de protéine synthétisée pourrait ne pas être suffisante pour activer le gène #2. Ainsi, il est souvent nécessaire de supplémenter le design à haut-niveau par un design à un niveau d'abstraction plus bas. Un autre exemple est la fonction d'amplification decrite par Xie *et al.* (Xie et al. 2011) où il faut deux couches de régulation pour générer une transition suffisemment précise entre les différents états du système.

La logique multivaluée est un niveau d'abstraction plus bas que la logique Booléenne, mais cependant plus haute que le niveau analogue. Il a été montré qu'il est possible de faire du design à un niveau intermédiaire d'abstraction. Les recherches de René Thomas dans ce domaine méritent d'être soulignées (Thomas, Thieffry, and Kaufman 1995). En effet, il a dévelopé un formalisme pour la modélisation de comportement dynamique de RRG via des variables logiques multivaluées, des règles, des graphes et une représentation graphique des différents états. Quelques années plus tard, Gilles Bernot a étendu son approche pour y inclure les propriétés temporelles des RRGs (Bernot et al. 2004). Plus récemment, des recherches ont porté sur une altérnative basée sur la logique floue, utilisée pour décrire les lois qui gouvernent les relations entre concentrations de protéines et état des gènes (Gendrault, Madec, Lemaire, et al. 2014). Le principal atout de la logique floue sur la logique multivaluée est que le lien entre la valeur floue et la véritable concentration n'est jamais perdu. La logique floue peut être utilisée non seulement pour décrire les systèmes à un niveau intermédiaire d'abstraction, mais aussi à des fins de design. Dans ce cas-là, un algorithm teste toutes les combinaisons de matrices de règles possibles (piochée dans une librairie) et trouve celle qui correspond au mieux au comportement désiré. Chaque matrice de règles correspond à une interaction gène-protéine et son contenu fournit au designer des pistes de design importantes pour implémenter ces dernières.

L'automatisation du design de RRG au niveau analogue est très proche de la synthèse analogique en microélectronique. Ce sujet a été largement étudié depuis le début des années 80 mais la question reste toujours ouverte. Plusieurs outils et méthodes ont été mises au point en utilisant des formalismes

spécifiques et des calculs formels (Doboli and Vemuri 2003; Lohn and Colombano 1998). Néanmoins, ces développements n'ont pas conduit à un outil générique qui aurait été largement répandu dans la communauté des designers analogiques. La raison principale est qu'ils étaient trop complexes et trop spécifiques pour être utilisés pour une large gamme de circuits. En outre, ils exigent des bibliothèques et/ou des méthodes d'apprentissage artificielles ou l'exploitation de l'expérience du designer pour formaliser des règles, ce qui n'est pas simple. D'une manière plus générale, il a été observé que le rapport entre la difficulté de mise en œuvre de tels algorithmes et la complexité des circuits pouvant être synthétisés était médiocre par rapport à une conception faite à la main.

L'une des percées les plus remarquables dans le domaine de la synthèse analogique a été faite par Koza en 1997 (Koza et al. 1997) à partir d'algorithmes de programmation génétique. Il démontre le potentiel de sa méthode sur un grand nombre de circuits électroniques (filtres, amplificateurs, contrôleurs) pour lesquels les algorithmes génétiques ont fourni des solutions (topologie des circuits et dimensionnement des composants) très compétitives par rapport à l'intelligence humaine (Koza and Stancalie 2003). À l'époque, le principal défaut des algorithmes évolutifs était qu'ils nécessitaient une puissance de calcul qui ne pouvait être fournie que par des superordinateurs. Ce n'est plus le cas avec les technologies actuelles: l'exploitation du calcul parallélisé sur de petits réseaux et/ou l'exploitation de la performance des Graphical Processor Units (GPU) permet d'obtenir de tels résultats avec des systèmes informatiques de prix raisonnable.

D'une manière générale, la synthèse analogique est un sous-ensemble des problèmes inverses. Les problèmes inverses consistent à rechercher les fonctions (dans ce cas le modèle du système) qui donnent une réponse aussi proche que possible à celle attendue (dans ce cas, la spécification). Plusieurs méthodes existent pour s'attaquer à ce genre de problèmes. La plupart d'entre eux exigent que la topologie du système soit connue *a priori* et optimise les paramètres du système pour converger vers une cible. L'accent est mis sur les algorithmes évolutifs qui semblent prometteurs dans notre contexte. Dans le chapitre 5, on réalise l'application de différentes familles d'algorithmes évolutifs au contexte de la biologie synthétique. Tout d'abord, on considère les RRGs pour lequel la topologie (c'est-à-dire l'assemblage des mécanismes élémentaires) a été précédemment fixée et pour laquelle nous cherchons à trouver le bloc de construction le plus approprié pour réaliser chaque mécanisme. Pour ce faire, deux approches sont possibles. L'optimisation combinatoire peut être utilisée pour essayer différentes combinaisons de blocs de construction et trouver ceux qui présentent la meilleure réponse. D'un autre côté, l'optimisation continue peut être utilisée pour trouver les ensembles de paramètres pour chaque modèle de chaque mécanisme qui donne la meilleure réponse. Cet ensemble de paramètres guide ensuite le concepteur dans le choix des blocs de construction qui seront utilisés pour effectuer chaque mécanisme. Cette deuxième méthode est illustrée dans le contexte d'un RRG dans le chapitre 5 de la thèse. Enfin, nous abordons le problème beaucoup plus complexe de l'automatisation de la conception d'un RRG pour lequel la topologie du réseau n'est pas définie *a priori*. Pour ce faire, des méthodes de programmation génétique doivent être appliquées.

3.3. Conclusion du chapitre 5

Nous avons évalué notre algorithme sur une cible passe-bande. Nous voyons qu'avec seulement 4 points orientant l'évolution des réseaux, notre algorithme fournit une grande variété de réponses passe-bande et d'implémentations possibles de celles-ci. Une cible sous-définie a l'avantage de calculer des solutions dans un temps réduit, car l'évaluation est plutôt rapide. Fait intéressant, nous pourrions obtenir un système en concurrence avec le système Basu en termes de nombre de gènes. Une analyse grossière a révélé les principales caractéristiques des différents réseaux élaborés par l'algorithme. Une analyse plus fine complétée par des essais sur banc humide pourrait conduire à un passe-bande biologique très compact.

Avec une cible plus fine ayant plus de points à correspondre, nous n'avons pas pu trouver la paramétrisation correcte de l'algorithme. Seuls quelques paramètres ont été modifiés. Une piste d'investigation serait de donner à l'algorithme la possibilité d'ajouter des complexes dans les réseaux. La population initiale pourrait également être initialisée au hasard.

4. Partie 3 – Prototypage virtuel des systèmes biologiques dépendant du temps et de l'espace

La partie 2 a mis en évidence une observation: dès que nous souhaitons réaliser des RRGs qui exécutent une fonction non-élémentaire (systèmes séquentiels, machines à états finis, compteurs, etc.), ces RRGs ne peuvent pas être intégrés dans une seule cellule en raison de limitations technologiques. Ainsi, nous devons les diviser en plusieurs sous-réseaux qui interagissent, chacun d'entre eux étant implémenté dans une cellule différente. Cette nouvelle façon de faire des RRGs a été suggérée il y a quelques années et validée expérimentalement sur un circuit à petite échelle (soit une porte XOR composée de 4 cellules) (Brenner, You, and Arnold 2008; Tamsir, Tabor, and Voigt 2011). La conception à plus grande échelle de tels systèmes nécessite un outil qui permet la simulation combinée des mécanismes biologiques intracellulaires et de la carte de la concentration des molécules impliquées dans les communications de cellule à cellule. C'est la question abordée dans la partie 3 de la thèse.

Une liste non exhaustive des exigences pour un tel outil est donnée ci-dessous:

- L'outil devrait être capable de combiner des phénomènes locaux et globaux
- Pour les phénomènes locaux, il faut prendre comme entrée un formalisme commun, tel qu'un fichier SBML, sous lequel existent déjà des modèles de systèmes biologiques
- Il devrait être entièrement intégré dans notre environnement de conception
- Il devrait être configurable de manière à ce que l'utilisateur puisse jouer sur le compromis entre la précision et le temps de calcul
- Il devrait être open-source.

Plusieurs solutions existent déjà pour gérer de tels modèles et sont décrites dans le manuscrit. Cependant, cet état de l'art n'a pas mis en évidence un outil qui répond à nos exigences. D'un autre côté, nos collègues ont récemment développé un simulateur dédié à un problème similaire, mais dans

un domaine très différent. Leur but était la simulation électrothermique des circuits intégrés (Krencker et al. 2010). L'outil que nous avons développé au cours de cette thèse est une adaptation de ce simulateur à un contexte biologique.

Tout comme la partie 2, la partie 3 du manuscrit est divisée en deux chapitres. Le premier (chapitre 6) décrit la façon dont l'outil a été développé. Il commence par un état de l'art des approches et des outils existants qui peuvent être utilisés pour prendre en compte l'emplacement de l'espace dans les modèles biologiques. L'accent est mis sur trois outils existants: HSIM, COMSOL et Virtual Cell. Dans la section 2, le contexte théorique sur l'équation de réaction-diffusion (loi de Fick utilisée pour modéliser les systèmes biologiques dépendant de l'espace et du temps) et sa contrepartie, l'équation de la chaleur, sont rappelés. Ensuite, notre outil est décrit. Il est principalement composé d'un mailleur, d'un générateur de circuit et d'un modèle de maille élémentaire. Deux schémas de discrétisation ont été évalués pour calculer ce modèle: les méthodes des différences finies d'une part et des éléments finis d'autre part.

Le deuxième chapitre de cette partie (Chapitre 7) est consacré à la validation de l'outil et à son application sur plusieurs cas d'utilisation:

- Le générateur de motifs de Basu, déjà présenté dans la deuxième partie, mais cette fois dans sa version complète incluant les cellules émettrices et réceptrices
- Un XOR réalisé avec un consortium de 3 cellules, chacune effectuant une fonction NOR et communiquant entre elles via les acyl homosérine lactones (AHL)
- Un système simplifié proie-prédateur (Balagaddé et al. 2008)
- L'étude de la synchronisation d'oscillateurs biologiques (Garcia-Ojalvo, Elowitz et Strogatz 2004)

4.1. Aperçu de notre simulateur

L'outil que nous avons développé est basé sur un simulateur SPICE. La capacité de ce langage à gérer à la fois les systèmes biologiques (Madec, Lallement, and Haiech 2017) et les problèmes de convection-diffusion thermique (Garci, Kammerer, and Hebrard 2014) a déjà fait ses preuves. Nous bénéficions ainsi de près de 50 ans d'expérience et de curation. Deux distributions de SPICE sont utilisées. Spectre MMSIM, un simulateur commercial intégré dans la suite de conception de circuits intégrés Cadence (https://www.cadence.com) qui offre différents types d'analyse (point de fonctionnement, transitoire, analyse CC, analyse AC, analyse de bruit, balayage de paramètres...). De plus, il est parallélisé et peut ainsi simuler des systèmes à grande échelle (avec des milliers d'équations) dans un temps de calcul très faible par rapport aux autres logiciels. Par ailleurs, un simulateur open source SPICE, à savoir NgSpice a également été testé et intégré dans l'outil suivant afin de garder l'outil complet gratuit. L'outil est composé d'une suite de cinq modules principaux écrits en différents langages (voir Figure 5): un mesher écrit en C ++, un générateur de netlist SPICE écrit en Python, un modèle générique d'une maille élémentaire décrit en Verilog-A ou directement dans SPICE, un simulateur SPICE et un script Python pour lire des fichiers de sortie de simulation et afficher les résultats. Le workflow complet est détaillé dans la thèse.

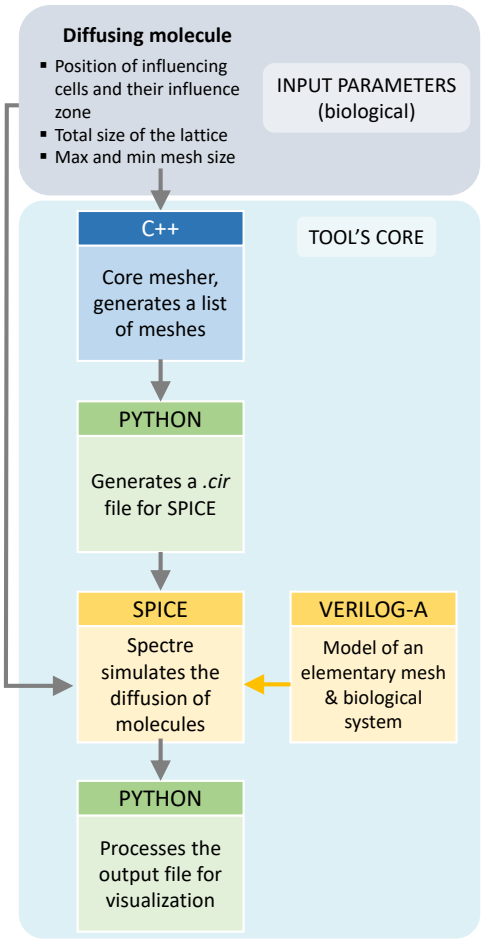


Figure 5 Modules principaux de l'outil dévelopé. Les étiquettes indiquent dans quelle langue ont été écrits les modules.

4.2. Validation du simulateur et résultats obtenus

Le chapitre 7 de la thèse fait l'objet de la validation des modèles proposés. A cet effet, deux cas sont considérés. Le premier est la diffusion transversale d'une frontière (où les molécules sont synthétisées) à l'autre. C'est un problème 1-D pour lequel une solution analytique peut être trouvée à l'état stable. Le second cas est la diffusion radiale d'une source ponctuelle au centre du réseau. Encore une fois, il existe une solution analytique qui servira de référence pour le processus de validation. Ensuite, le modèle validé est appliqué à de véritables problèmes biologiques.

4.2.1. Comparaison entre le modèle éléments finis et différences finies

Le modèle de différence finie est correct pour chaque configuration du maillage. Le modèle des éléments finis est plus précis que son homologue car le calcul du flux dans un nœud dépend de tous ses voisins. Cependant, cette version du modèle n'est stable que pour un ensemble restreint de configurations. Par conséquent, c'est la version initiale du modèle qui correspond à un modèle de différence finie qui est utilisée dans la thèse pour produire les résultats présentés. Tous les résultats ont été générés avec le simulateur Spectre.

4.2.2. Résultats obtenus sur des cas d'utilisation biologiques

Le simulateur est utilisé pour montrer quel type de résultats il peut apporter à des problèmes de la biologie synthétique. Au préalable, deux problématiques sont abordées: l'ajout de composants

externes (cellule, réaction localisée, membranes...) et la description de réseaux multicouches dans le cas d'un système composé de plusieurs espèces de diffusions.

Pour le premier point, on a besoin de descriptions de modèles biologiques en SPICE ou en Verilog-A. Un modèle SPICE peut être généré à partir d'une description SBML ou d'un fichier d'entrée propriétaire (composé d'une liste de paramètres, d'une liste de réaction et d'une liste de réactions avec équation de vitesse prédéfinie) avec BB-SPICE. De plus, écrire de A à Z un modèle Verilog-A associé à un mécanisme biologique n'est pas si difficile. Ceci est fait à l'aide de l'analogie entre l'électronique et la biologie décrite au chapitre 2. L'écriture d'un modèle biologique en langage VHDL a également été démontrée dans (Gendrault, Madec, Lallement, et al. 2014). La transposition de la méthodologie à du Verilog-A est assez simple.

Deuxièmement, la description des réseaux multicouches a déjà été prise en compte dans le générateur de netlist. La version actuelle du générateur duplique le maillage de base autant de fois que nécessaire en créant de nouveaux nœuds dont l'ID est incrémenté du nombre total de nœuds dans un maillage. Ainsi, les noeuds de chaque couche sont uniques et les maillages sont indépendants les uns des autres. De cette façon, ils sont superposables, ce qui facilite leur interconnexion via des modèles de composants externes.

Les résultats obtenus prouvent la validité de notre approche. De plus, ils montrent que notre outil peut être utilisé pour l'étude et l'optimisation de systèmes biologiques simples. L'utilisation de l'outil dans des cas réels a également mis en lumière certaines limites de l'outil ainsi que certaines améliorations. Ceux-ci sont discutés ci-après.

4.3. Résumé de la partie 3

Comme mentionné précédemment, il est devenu nécessaire de rendre compte du comportement spatio-temporel des systèmes biologiques. En effet, avec la croissance et la complexité des systèmes, de nombreux biologistes divisent maintenant leurs systèmes en sous-systèmes mis en œuvre dans différentes populations cellulaires. Comme ces populations cellulaires sont souvent dispersées dans l'espace, elles communiquent en envoyant et en recevant un signal moléculaire, dont la diffusion dans l'espace et le temps est d'une importance capitale pour le bon fonctionnement du système.

Les outils actuels pour la simulation des systèmes biologiques dans l'espace et le temps comprennent VirtualCell et COMSOL. Les principales conclusions de notre état de l'art concernant ces outils sont résumées par la suite.

D'une part, Virtual Cell dispose d'une interface graphique intuitive qui peut être utilisée pour définir la géométrie du problème et les réactions biologiques impliquées dans le système. Plusieurs solveurs sont disponibles, y compris des simulations déterministes et stochastiques pour les problèmes 0D à 3D. Il a été testé et a conduit à des résultats comparables à ceux obtenus avec notre outil. Néanmoins, Virtual Cell n'effectue que des simulations transitoires. Des résultats de simulation statique ont été obtenus en effectuant une simulation transitoire jusqu'à atteindre l'état stationnaire, ce qui prend du temps. Les calculs ne sont pas effectués localement sur l'ordinateur mais sont distribués et exécutés

sur un serveur distant. Ainsi, la comparaison du temps de calcul n'est pas très pertinente. Les résultats donnés dans le Tableau 2 correspondent à la durée effective de l'opération et sont indépendants de l'ordinateur sur lequel ils sont effectués. Virtual Cell utilise également un maillage pour résoudre les équations aux dérivées partielles mais la taille du maillage est fixe. Ainsi, pour obtenir une résolution spatiale de 1mm au centre, il est nécessaire d'avoir un maillage de 1mm sur toute la surface, ce qui conduit à 10201 nœuds. C'est entre 3 et 4 fois plus que le nombre de nœuds requis pour obtenir la même résolution spatiale près de la source et la même précision avec un réseau adaptatif. Enfin, une autre limitation de Virtual Cell est la difficulté de coupler des modèles biologiques avec des modèles d'autres domaines de la physique. Sans être impossible, cela nécessite une traduction du problème physique en un problème biochimique équivalent.

D'autre part, COMSOL est un logiciel dédié à la simulation de systèmes multi-physiques. Il propose des modules dédiés à la diffusion d'espèces chimiques et à l'ingénierie de réactions pouvant être couplées entre elles et avec d'autres modules issus d'autres domaines de la physique. Il a déjà été utilisé pour l'étude des systèmes biochimiques (Dreij et al. 2011; Vollmer, Menshykau, and Iber 2013). Le solveur d'équations aux dérivées partielles (EDP) ainsi que le mailleur implémenté dans COMSOL sont plus sophistiqués que celui proposé dans notre approche. Le réseau adaptatif peut être calculé automatiquement par COMSOL en fonction de la géométrie et du problème lui-même (détection des points chauds pour déterminer automatiquement les zones de raffinement) alors qu'il doit être défini par l'utilisateur dans notre cas. La définition de la géométrie et les équations de diffusion sont simplifiées par une interface graphique. Au contraire, la définition des réactions est moins intuitive. L'importation de modèles biologiques existants ou l'intégration de modèles de réactions complexes est douloureuse. Les algorithmes de résolution d'EDP sont variés et optimisés pour une implémentation multicœur, ce qui conduit à des temps de calcul bas comme le montre le Tableau 2. Les types de simulation les plus courants (statique, dynamique, paramétrique) sont disponibles et COMSOL peut être couplé avec MATLAB pour obtenir un banc d'essai amélioré. Malheureusement, COMSOL est un outil commercial coûteux, ce qui limite sa gamme d'applications, en particulier dans le domaine académique.

Par rapport à eux, notre approche offre une solution qui se positionne comme un bon compromis entre précision et rapidité de calcul, s'appuie sur des outils qui ont prouvé leur efficacité depuis des années (pour simuler des microprocesseur composés de 1 milliard de transistors) et offre un couplage avec d'autres domaines de la physique à travers les lois généralisées de Kirchhoff. La comparaison de plusieurs caractéristiques de notre approche, COMSOL et Virtual Cell, est présentée dans le Tableau 2. La comparaison entre les résultats de simulation et le temps de calcul n'est pas simple en raison des différentes implémentations des simulateurs et des différents ordinateurs sur lesquels les résultats ont été générés. Le point de référence utilisé pour la comparaison est celui décrit dans le chapitre 7, section 1, tableau 2, à savoir un carré de 100 x 100 mm² avec une source à (50, 50) et des conditions aux limites sans flux. La résolution spatiale près de la source doit être d'au moins 1 mm. Comme Virtual Cell ne possède pas de mailleur adaptatif, un réseau de 100x100 est utilisé pour discrétiser toute la surface. Avec COMSOL, un maillage adaptatif de 1858 nœuds est généré afin que les éléments les plus petits

soient inférieurs à 1 mm au centre. Enfin, avec nos outils, différents maillages ont été testés. La réduction du nombre de nœuds dégrade évidemment la qualité des résultats mais, en contrepartie, réduit drastiquement le temps de calcul. Les résultats retenus pour la comparaison sont ceux du maillage n°7 (un maillage régulier 100x100) et maillage n°9 (un maillage adaptatif 25x25 affiné une fois dans un cercle centré de 30mm et deux fois dans un cercle centré de 5mm) qui peuvent être considérés comme ayant un bon compromis entre temps de calcul et précision. De plus, nous avons comparé les performances de Spectre et NGSPICE pour notre approche. Le simulateur commercial a de meilleures performances que celui open-source lorsque le nombre de nœuds dépasse 2000. Ceci est encore plus marqué en simulation transitoire. Ceci est probablement dû à la fonctionnalité multi-thread proposée par Spectre et qui n'est pas encore supportée par NGSpice.

Tableau 2 Comparaison des fonctionnalités et des performances de COMSOL, Virtual Cell et de notre approche (à la fois avec Spectre et NGSpice).

	COMSOL	Virtual Cell	Spectre	NGSPICE
Mesher	Triangle Adaptive mesh Driven by physics	Rectangular Fixed	Rectangular Adaptive Defined by user	
Discretization sheme	Finite element (various scheme)	Finite Differences	Finite Differences Finite Element (*still under dev*.)	
Multiphysic interface	Yes Coupling with other COMSOL module	Not straightforward	Yes, using Kirchhoff-based equivalent circuits and HDL	
Simulation type	Deterministic Steady State, Transient, Parametric	Deterministic or Stochastic Transient	Deterministic Steady State, Transient, Parametric, Frequency analysis, Noise analysis	
Scripting for the development of complex testbenchs	Yes, with MATLAB	No	Yes, with SPICE simulation control directives	
Software type	Commercial	Freeware	Open-source model generator Commercial simulator	Fully open-source
Computation time at the steady state	< 1 sec for a 1858-nodes model		14.5 sec for the Mesh #7 (regular, 10201 nodes) 1.29 sec for Mesh #9 (adaptive, 1969 nodes).	20.5 sec for the Mesh #7 (regular, 10201 nodes) . 2.19 sec Mesh #9 (adaptive, 1969 nodes).
Computation time in transient	19 sec for a 1858-nodes model and 20 time steps	140sec for a 10201-nodes model and 20 time steps	18.7 sec for the Mesh #7 and 20 time steps 3.44 sec for The Mesh #9and 20 time steps	80.8 sec for the Mesh #7 and 20 time steps 13.18 sec for the Mesh #9 and 20 time steps

En résumé, notre simulateur présente quatre avantages principaux par rapport aux outils existants:

- il est basé sur un algorithme très simple pour la discrétisation de l'espace, ce qui facilite la description des phénomènes de diffusion avec des modèles compacts simples
- il fournit un couplage direct entre le modèle de diffusion des molécules, les modèles de systèmes biologiques jouant un rôle dans le milieu de diffusion et les modèles issus d'autres domaines de la physique
- il utilise un cœur de simulation SPICE, qui a prouvé son efficacité depuis des années, notamment pour les systèmes avec un nombre élevé d'équations différentielles et qui sera amélioré dans un futur proche pour faire face aux nouveaux défis de la microélectronique
- il est open-source.

Avec ces caractéristiques, c'est un outil très puissant pour la simulation et le prototypage virtuel de systèmes biologiques à l'intérieur des cellules ou impliquant divers types de cellules qui communiquent entre elles par l'intermédiaire de messagers chimiques. Par exemple, il peut être couplé avec des algorithmes évolutifs pour explorer de nouvelles solutions qui exploitent des consortiums de cellules en biologie synthétique ou pour acquérir un niveau de complexité dans la modélisation de systèmes biologiques.

Bien que le modèle soit simplifié en raison de l'algorithme de discrétisation mis en œuvre (en comparaison de modèles COMSOL par exemple), la simulation d'un modèle complet peut prendre beaucoup de temps, surtout lorsque le nombre d'espèces augmente. Compte tenu des propriétés des équations à résoudre, le déploiement sur GPU pourrait fournir une solution pour accélérer le calcul. Des versions récentes du simulateur SPICE open-source optimisé par GPU ont été publiées (Keiter et al. 2014; Lannutti 2014) et leur couplage avec notre outil est actuellement à l'étude. Une autre perspective de ce travail est d'améliorer la façon dont plusieurs couches de maillage peuvent être interconnectées pour des systèmes avec plusieurs espèces diffusantes. Lorsque plusieurs couches de maillage sont implémentées (par exemple dans l'exemple de proie-prédateur simplifié), le même raffinement est appliqué sur chaque couche pour faciliter l'interconnexion entre elles (les maillages de chaque couche se chevauchent et les coordonnées des nœuds sont identiques sur chaque couche). Appliquer un raffinement spécifique sur chaque couche rendrait la génération de la netlist plus complexe mais, d'un autre côté, réduirait le nombre de nœuds dans le modèle et accélèrerait le calcul.

Jusqu'à maintenant, notre simulateur n'était utilisé que dans notre équipe. L'expertise sur l'outil était donc à proximité et aucune interface graphique n'était requise pour exécuter correctement une simulation. Cependant, une interface graphique aurait facilité la mise en place et le fonctionnement de notre outil.

De plus, pour élargir le champ d'application, une validation sur un cas d'utilisation à une échelle différente serait la prochaine étape. En effet, dans nos exemples, les cellules (et donc leurs compartiments) sont ponctuelles. Dans certains systèmes biologiques, une simulation au niveau cellulaire des flux de molécules est nécessaire. Un exemple notable est l'étude des oscillations de calcium dans une cellule. Ces oscillations sont impliquées dans de nombreux processus cellulaires et sont donc largement étudiées par la communauté de la biologie. Le calcium est stocké dans différents

compartiments de la cellule (par exemple le cytoplasme, le réticulum endoplasmique, les mitochondries...) et les flux de calcium entre ces compartiments (les motifs d'oscillations calciques) déterminent le comportement d'une cellule en réaction à un signal externe. Être capable de simuler ces échanges de calcium au niveau d'une cellule apporterait sûrement un nouveau savoir dans le vaste monde des motifs d'oscillation du calcium. De plus, cela ouvrirait la porte à la prédiction de modèles calcium encore non observés. Avec cette possibilité à portée de main, les concepteurs disposeraient d'un outil dédié pour moduler le comportement d'une cellule. Comme les dérèglements dans les oscillations du calcium sont d'une importance capitale dans plusieurs maladies, une application évidente est la thérapeutique. Une extension utile à une utilisation à plus petite échelle de notre outil serait la mise en œuvre d'un modèle 3D.

En outre, comme mentionné ci-dessus, l'outil a déjà été utilisé dans un autre projet qui n'est pas directement lié à la biologie synthétique. L'objectif était de simuler le transport de diffusion entre gouttelettes dans une puce microfluidique. Dans un tel dispositif, chaque gouttelette peut être vue comme un seul bioréacteur indépendant. Cependant, la fuite de produits chimiques d'une gouttelette à l'autre modifie la composition chimique à l'intérieur du réacteur et introduit des interactions parasites entre chaque réaction. Notre simulateur a été utilisé pour simuler le phénomène de diffusion. Plus précisément, les gouttelettes sont décrites comme deux disques 2D à fort coefficient de diffusion séparés par de l'huile qui est modélisée par un faible coefficient de diffusion. Les résultats de la simulation sont prometteurs et peuvent être utilisés pour optimiser le dispositif dans le futur.

Pour conclure, nous avons développé un outil open-source qui peut être utilisé pour la modélisation et la simulation de systèmes biologiques qui dépendent de l'espace et du temps. Les applications sont nombreuses en biologie des systèmes, en biologie synthétique en l'associant à notre environnement de conception de prototypage virtuel et/ou automatique et pour des applications à l'interface avec d'autres domaines de la physique dont les phénomènes peuvent être modélisés par des réseaux de diffusion et/ou de Kirchhoff, soit des dispositifs électroniques, des phénomènes thermiques, de la microfluidique en laboratoire sur puce et biocapteurs.

5. Conclusion générale

5.1. Notre contribution au monde merveilleux de la biologie synthétique

Dans la partie 1 de lathèse, nous présentons un design flow pour la biologie. Notre travail pallie aux deux éléments manquants identifiés en fournissant une solution pour la conception automatique de systèmes biologiques et un simulateur spatio-temporel open-source (presque) prêt à l'emploi. Comme nos outils ne sont pas encapsulés par une interface graphique limitative ou par une licence commerciale dissimulée (parfois vous ne savez pas vraiment ce que ces logiciels font vraiment...), notre environnement est ouvert à l'extension. Par notre choix du langage de description, simuler des systèmes mixtes comprenant des éléments d'autres domaines que la biologie est plus facile avec notre outil qu'avec les outils dédiés à la biologie. De plus, notre traducteur de manipulation SBML, BB-SPICE, permet de spécifier facilement de nouveaux systèmes biologiques dans le format Verilog. La plate-

forme EASEA, comme son nom l'indique, facilite également l'adaptation des algorithmes évolutionnaires à tout problème biologique, même pour les padawan de la programmation.

5.2. Faciliter le développement de systèmes innovants

Notre flux de conception est maintenant presque complet et permet la manipulation de systèmes biologiques complexes, de la conception à la simulation. Un exemple récent (Müller et al. 2017) possède toutes les caractéristiques pour illustrer la contribution potentielle de notre outil. Müller *et al.* ont conçu un convertisseur biologique analogique-numérique. Leur système comporte une partie analogique, le biocapteur et une partie numérique composée d'un circuit booléen. Ils divisent également leur système sur différentes populations cellulaires séparées par une interface liquide où la diffusion se produit. Avec notre outil, ils auraient pu réaliser le prototypage virtuel du GRN pour l'implémenter dans ces différentes populations cellulaires. Différentes solutions auraient probablement été trouvées par l'algorithme, leur donnant des alternatives entre lesquelles choisir. Avoir un choix permet d'ordonner les solutions suggérées par des critères qui n'auraient autrement pas été pris en compte. En plus d'élargir l'horizon des designers, notre environnement leur aurait probablement permis de gagner du temps à la paillasse, en faisant des simulations spatiotemporelles des systèmes à tester.

5.3. Connexions manquantes

Comme le passage d'une simulation *in silico* à l'expérience à la paillasse nécessite une mise en œuvre effective du système conçu, nous devons mentionner ici les limites de l'environnement actuel. En effet, plusieurs connexions manquent, notamment une libraire de *parts*. Comme mentionné dans la partie 1, il n'y a pas de base de données standard mature pour la biologie synthétique. La bibliothèque la plus avancée est actuellement le référentiel BioBrick, même si une description standard d'une BrioBrick n'est pas encore implémentée. Comme cette base de données idéale n'existe pas, nous n'avons pas eu la possibilité de connecter l'outil de conception à un tel référentiel. Une première perspective consisterait à gérer manuellement une telle bibliothèque et à y connecter notre outil. De plus, une implémentation réelle d'un GRN nécessite plus que de pouvoir fournir à l'utilisateur les séquences d'ADN correspondant aux parties du système. Les contraintes expérimentales liées à la réalisation *in vivo* des systèmes conçus devraient probablement être prises en compte. Comme le côté expérimental de cette thèse était plutôt rare, je suis progressivement passée du côté obscur de la biologie synthétique, qui correspond au travail en amont (à savoir le développement d'outils) qui doit être réalisé pour que les expérimentateurs et les concepteurs aient à leur disposition des outils aptes à les aider.

5.4. L'outil idéal

Au-delà de ces liens manquants, d'autres améliorations ont été envisagées pour le flux de conception. Les connexions entre les différents outils pourraient être renforcées. En effet, l'outil de conception repose sur la capacité à simuler le système biologique à réaliser. Comme montré dans le chapitre 5 de la thèse, une simulation est nécessaire pour évaluer le score de fitness utilisé pour évaluer les

différentes solutions à un problème biologique. Jusqu'à présent, seuls les EDO étaient utilisées pour simuler les RRGs dans l'outil de conception. Lors de la conception d'un système ayant un comportement spatio-temporel à prendre en compte, l'outil de simulation que nous avons développé devrait être connecté à l'outil de conception. Une première perspective consisterait à créer une faible interaction entre ces outils: l'outil de conception ne pourrait exécuter des simulations SPICE que sur un maillage donné. Une seconde perspective nécessiterait d'apporter de l'intelligence dans le choix du maillage. Une première étape consisterait à faire en sorte que l'algorithme choisisse des maillages de plus en plus fins pour simuler les systèmes au cours de la convergence (l'affinement du maillage augmente avec les itérations de l'algorithme évolutionnaire). Dans un deuxième temps, on pourrait imaginer que l'algorithme serait aussi capable d'optimiser le maillage en ce qui concerne le rapport de la précision sur le nombre de nœuds (comme mentionné au chapitre 7, plus le réseau est fin, plus les résultats sont précis). Ces sentiers sont laissés ouverts aux nouveaux venus en biologie synthétique.

6. References

Anderson, J Christopher, Christopher A Voigt, and Adam P Arkin. 2007. "Environmental Signal Integration by a Modular AND Gate." *Molecular Systems Biology* 3. European Molecular Biology Organization: 133. doi:10.1038/msb4100173.

Balagaddé, Frederick K, Hao Song, Jun Ozaki, Cynthia H Collins, Matthew Barnet, Frances H Arnold, Stephen R Quake, and Lingchong You. 2008. "A Synthetic Escherichia Coli Predator-Prey Ecosystem." *Molecular Systems Biology* 4 (1): 187. doi:10.1038/msb.2008.24.

Basu, Subhayu, Yoram Gerchman, CH Collins, FH Arnold, and R Weiss. 2005. "A Synthetic Multicellular System for Programmed Pattern Formation." *Nature* 434 (April). http://www.nature.com/nature/journal/v434/n7037/abs/nature03461.html.

Beal, Jacob, Ting Lu, and Ron Weiss. 2011. "Automatic Compilation from High-Level Biologically-Oriented Programming Language to Genetic Regulatory Networks." Edited by Eshel Ben-Jacob. *PLoS ONE* 6 (8). Public Library of Science: e22490. doi:10.1371/journal.pone.0022490.

Beal, Jacob, Ron Weiss, Douglas Densmore, Aaron Adler, Evan Appleton, Jonathan Babb, Swapnil Bhatia, et al. 2012. "An End-to-End Workflow for Engineering of Biological Networks from High-Level Specifications." *ACS Synthetic Biology* 1 (8). American Chemical Society: 317–31. doi:10.1021/sb300030d.

Bernot, Gilles, Jean-Paul Comet, Adrien Richard, and Janine Guespin. 2004. "Application of Formal Methods to Biological Regulatory Networks: Extending Thomas' Asynchronous Logical Approach with Temporal Logic." *Journal of Theoretical Biology* 229 (3). Academic Press: 339–47. doi:10.1016/J.JTBI.2004.04.003.

Bilitchenko, Lesia, Adam Liu, Sherine Cheung, Emma Weeding, Bing Xia, Mariana Leguia, J Christopher Anderson, and Douglas Densmore. 2011. "Eugene-a Domain Specific Language for Specifying and Constraining Synthetic Biological Parts, Devices, and Systems." *PloS One* 6 (4). Public Library of Science.

Brenner, Katie, Lingchong You, and Frances H Arnold. 2008. "Engineering Microbial Consortia: A New Frontier in Synthetic Biology." *Trends in Biotechnology* 26 (9): 483–89. doi:10.1016/j.tibtech.2008.05.004.

Chandran, Deepak, Frank T Bergmann, and Herbert M Sauro. 2009. "TinkerCell: Modular CAD Tool for Synthetic Biology." *Journal of Biological Engineering* 3 (1): 19. doi:10.1186/1754-1611-3-19.

Czar, Michael J, Yizhi Cai, and Jean Peccoud. 2009. "Writing DNA with GenoCAD." *Nucleic Acids Research* 37 (Web Server issue). Oxford University Press: W40-7. doi:10.1093/nar/gkp361.

Densmore, Douglas, Joshua T. Kittleson, Lesia Bilitchenko, Adam Liu, and J. Christopher Anderson. 2010. "Rule Based Constraints for the Construction of Genetic Devices." In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 557–60. IEEE. doi:10.1109/ISCAS.2010.5537540.

Doboli, Alex, and Ranga Vemuri. 2003. "Exploration-Based High-Level Synthesis of Linear Analog Systems Operating at Low/medium Frequencies." *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 22 (11). IEEE: 1556–68.

Dreij, Kristian, Qasim Ali Chaudhry, Bengt Jernström, Ralf Morgenstern, and Michael Hanke. 2011. "A Method for Efficient Calculation of Diffusion and Reactions of Lipophilic Compounds in Complex Cell Geometry." *PloS One* 6 (8). Public Library of Science: e23128. doi:10.1371/journal.pone.0023128.

Funahashi, Akira, Mineo Morohashi, Hiroaki Kitano, and Naoki Tanimura. 2003. “CellDesigner: A Process Diagram Editor for Gene-Regulatory and Biochemical Networks.” *BIOSILICO* 1 (5): 159–62. doi:10.1016/S1478-5382(03)02370-9.

Garci, Maroua, Jean-Baptiste Kammerer, and Luc Hebrard. 2014. “Compact Modeling and Electro-Thermal Simulation of Hot Carriers Effect in Analog Circuits.” In *2014 IEEE 12th International New Circuits and Systems Conference (NEWCAS)*, 125–28. IEEE. doi:10.1109/NEWCAS.2014.6933999.

Gendrault, Yves, Morgan Madec, Christophe Lallement, and Jacques Haiech. 2014. “Modeling Biology with HDL Languages: A First Step toward a Genetic Design Automation Tool Inspired from Microelectronics.” *IEEE Transactions on Biomedical Engineering* 61 (4). IEEE Computer Society: 1231–40.

Gendrault, Yves, Morgan Madec, Martin Lemaire, Christophe Lallement, and Jacques Haiech. 2014. “Automated Design of Artificial Biological Functions Based on Fuzzy Logic.” In *Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE*, 85–88.

Goler, Jonathan Ari. 2004. “BioJADE: A Design and Simulation Tool for Synthetic Biological Systems.” https://dspace.mit.edu/handle/1721.1/30475.

Hoteit, Imad, Nawwaf Kharma, and Luc Varin. 2012. “Computational Simulation of a Gene Regulatory Network Implementing an Extendable Synchronous Single-Input Delay Flip-Flop.” *Bio Systems* 109 (1). Elsevier Ireland Ltd: 57–71. doi:10.1016/j.biosystems.2012.01.004.

Hucka, M., A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, et al. 2003. “The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models.” *Bioinformatics* 19 (4): 524–31. doi:10.1093/bioinformatics/btg015.

Juty, N, R Ali, M Glont, S Keating, N Rodriguez, MJ Swat, SM Wimalaratne, et al. 2015. “BioModels: Content, Features, Functionality, and Use.” *CPT: Pharmacometrics & Systems Pharmacology* 4 (2): 55–68. doi:10.1002/psp4.3.

Keiter, ER, T Mei, TV Russo, and RL Schiek. 2014. “Xyce Parallel Electronic Simulator Users Guide, Version 6.1.”

Koza, John R., and Gheorghe. Stancalie. 2003. *Genetic Programming IV : Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers.

Koza, John R, Forrest H Bennett, David Andre, Martin A Keane, and Frank Dunlap. 1997. “Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming.” *Evolutionary Computation, IEEE Transactions on* 1 (2). IEEE: 109–28.

Krencker, Jean-Christophe, Jean-baptiste Kammerer, Yannick Hervé, and Luc Hébrard. 2010. “Direct Electro-Thermal Simulation of Integrated Circuits Using Standard CAD Tools.” IEEE, 1–4. http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5636296.

Lannutti, Francesco. 2014. “CUSPICE (NGSPICE on CUDA Platform) User Guide.”

Le Duc, Stéphane. 1910. *Théorie Physico-Chimique de La Vie et Générations Spontanées.* Paris: A. Poinat. doi:10.5962/bhl.title.32591.

Lohn, Jason D, and Silvano P Colombano. 1998. “Automated Analog Circuit Synthesis Using a Linear Representation.” In *Evolvable Systems: From Biology to Hardware*, 125–33. Springer.

Madec, Morgan, Christophe Lallement, and Jacques Haiech. 2017. “Modeling and Simulation of Biological Systems Using SPICE Language.” *PloS One* 12 (8). Public Library of Science: e0182385. doi:10.1371/journal.pone.0182385.

Madec, Morgan, Francois Pecheux, Yves Gendrault, Loïc Bauer, Jacques Haiech, and Christophe Lallement. 2013. “EDA Inspired Open-Source Framework for Synthetic Biology.” In *2013 IEEE Biomedical Circuits and Systems Conference, BioCAS 2013*, 374–77. doi:10.1109/BioCAS.2013.6679717.

Marchisio, Mario A., and Jörg Stelling. 2011. “Automatic Design of Digital Synthetic Gene Circuits.” Edited by Jason A. Papin. *PLoS Comput. Biol* 7 (2): e1001083. doi:10.1371/journal.pcbi.1001083.

Mirschel, Sebastian, Katrin Steinmetz, Michael Rempel, Martin Ginkel, and Ernst Dieter Gilles. 2009. “ProMoT: Modular Modeling for Systems Biology.” *Bioinformatics* 25 (5): 687–89. doi:10.1093/bioinformatics/btp029.

Müller, Marius, Simon Ausländer, Andrea Spinnler, David Ausländer, Julian Sikorski, Marc Folcher, and Martin Fussenegger. 2017. “Designed Cell Consortia as Fragrance-Programmable Analog-to-Digital Converters.” *Nature Chemical Biology* 13 (3). Nature Publishing Group: 309–16. doi:10.1038/nchembio.2281.

“Registry of Standard Biological Parts.” 2015. Accessed November 6. http://parts.igem.org/Main_Page.

Tamsir, Alvin, Jeffrey J Tabor, and Christopher A Voigt. 2011. “Robust Multicellular Computing Using Genetically Encoded NOR Gates and Chemical ‘Wires’.” *Nature* 469 (7329): 212–15. doi:10.1038/nature09565.

Thomas, René, Denis Thieffry, and Marcelle Kaufman. 1995. “Dynamical Behaviour of Biological Regulatory networks—I. Biological Role of Feedback Loops and Practical Use of the Concept of the Loop-Characteristic State.” *Bulletin of Mathematical Biology* 57 (2). Springer: 247–76.

Vollmer, Jannik, Denis Menshykau, and Dagmar Iber. 2013. “Simulating Organogenesis in COMSOL: Cell-Based Signaling Models.” In *Proceedings of COMSOL Conference*.

“What Is Synthetic Biology? - Synthetic Biology Project.” 2017. Accessed October 6. http://www.synbioproject.org/topics/synbio101/definition/.

Xie, Zhen, Liliana Wroblewska, Laura Prochazka, Ron Weiss, and Yaakov Benenson. 2011. “Multi-Input RNAi-Based Logic Circuit for Identification of Specific Cancer Cells.” *Science (New York, N.Y.)* 333 (6047). American Association for the Advancement of Science: 1307–11. doi:10.1126/science.1205527.

Yaman, Fusun, Swapnil Bhatia, Aaron Adler, Douglas Densmore, and Jacob Beal. 2012. “Automated Selection of Synthetic Biology Parts for Genetic Regulatory Networks.” *ACS Synthetic Biology*, July. American Chemical Society, 120706084748008. doi:10.1021/sb300032y.

Université de Strasbourg

Elise ROSATI

Outils d'aide à la conception pour l'ingénierie de systèmes biologiques

iCUBE

Résumé

En biologie synthétique, il existe plusieurs manières d'adresser les problèmes soulevés dans plusieurs domaines comme la thérapeutique, les biofuels, les biomatériaux ou encore les biocapteurs. Nous avons choisi de nous concentrer sur l'une d'entre elles : les réseaux de régulation génétique (RRG). Un constat peut être fait : la diversité des problèmes résolus grâce aux RRGs est bridée par la complexité de ces RRGs, qui a atteint une limite. Quelles solutions s'offrent aux biologistes, pour repousser cette limite et continuer d'augmenter la complexité de leur système ? Cette thèse a pour but de fournir aux biologistes les outils nécessaires à la conception et à la simulation de RRGs complexes. Un examen de l'état de l'art en la matière nous a mené à adapter les outils de la micro-électronique à la biologie ainsi qu'à créer un algorithme de programmation génétique pour la conception des RRGs. D'une part, nous avons élaboré les modèles Verilog A de différents systèmes biologiques (passe-bande, proie-prédateur, repressilator, XOR) ainsi que de la diffusion spatiotemporelle d'une molécule. Ces modèles fonctionnent très bien avec plusieurs simulateurs électroniques (Spectre et NgSpice). D'autre part, les premières marches vers l'automatisation de la conception de RRGs ont été gravies. En effet, nous avons développé un algorithme capable d'optimiser les paramètres d'un RRG pour remplir un cahier des charges donné. De plus, la programmation génétique a été utilisée pour optimiser non seulement les paramètres d'un RRG mais aussi sa topologie. Ces outils ont su prouver leur utilité en apportant des réponses pertinentes à des problèmes soulevés lors du développement de systèmes biologiques. Ce travail a permis de montrer que notre approche, à savoir adapter les outils de la micro-électronique et utiliser des algorithmes de programmation génétique, est valide dans le contexte de la biologie synthétique. L'assistance que notre environnement de développement fournit au biologiste devrait encourager l'émergence de systèmes plus complexes.

Mots-clefs: biologie synthétique; automatisation de la conception; conception assistée par ordinateur; modélisation et simulations; réseaux de régulation génétiques; microélectronique.

Résumé en anglais

In synthetic biology, Gene Regulatory Networks (GRN) are one of the main ways to create new biological functions to solve problems in various areas (therapeutics, biofuels, biomaterials, biosensing). However, the complexity of the designed networks has reached a limit, thereby restraining the variety of problems they can address. How can biologists overcome this limit and further increase the complexity of their systems? The goal of this thesis is to provide the biologists with tools to assist them in the design and simulation of complex GRNs. To this aim, the current state of the art was examined and it was decided to adapt tools from the micro-electronic field to biology, as well as to create a Genetic Programming algorithm for GRN design. On the one hand, models of diffusion and of other various systems (band-pass, prey-predator, repressilator, XOR) were created and written in Verilog A. They are already implemented and well-functioning on the Spectre solver as well as a free solver, namely NgSpice. On the other hand, the first steps of automatic GRN design were achieved. Indeed, an algorithm able to optimize the parameters of a given GRN according to a specification was developed. Moreover, Genetic Programming was applied to GRN design, allowing the optimization of both the topology and the parameters of a GRN. These tools proved their usefulness for the biologists' community by efficiently answering relevant biological questions arising in the development of a system. With this work, we were able to show that adapting micro-electronics and Genetic Programming tools to biology is doable and useful. By assisting design and simulation, such tools should promote the emergence of more complex systems.

Keywords: synthetic biology; design automation; computer-aided design; modelling and simulations; gene regulatory networks; microelectronics.