Fluxes and Fluctuations in Biochemical Models

Timo Maarleveld

FLUXES AND FLUCTUATIONS IN BIOCHEMICAL MODELS

TIMO RIAN MAARLEVELD

"WORK HARD, DREAM BIG, HAVE A BEER."

Members of the Doctoral Examination Committee:

Prof.dr. David Fell, Oxford Brookes University, UK;

Prof.dr. Jaap Heringa, Vrije Universiteit Amsterdam;

Prof.dr. Gunnar Klau, Centrum Wiskunde & Informatica;

Prof.dr. Klaas Hellingwerf, University of Amsterdam;

Prof.dr. Pieter-Rein ten Wolde, AMOLF; and

Prof.dr. Vítor Martins dos Santos, Wageningen UR.

The work presented in this thesis has been carried out at Centrum Wiskunde & Informatica and at the department of Systems Bioinformatics at the Vrije Universiteit Amsterdam.
I acknowledge funding from the project BioSolar Cells, co-financed by the Dutch Ministry of Economic Affairs.

ISBN: 978-94-6233-113-6

Document prepared with LaTeX 2$_\varepsilon$ and typeset by pdfTeX (Minion Pro font)
Cover: The metabolic network of *Synechocystis* (design by Timo Maarleveld)
Printed by: Gildeprint – Enschede

© 2015, Timo Maarleveld. All rights reversed.
No part of this publication may be reproduced in any form or by any electronic or mechanical means including information storage and retrieval systems without permission in writing from the author.

VRIJE UNIVERSITEIT

Fluxes and Fluctuations in Biochemical Models

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. V. Subramaniam,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Aard- en Levenswetenschappen
op donderdag 17 december 2015 om 9.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Timo Rian Maarleveld

geboren te Amsterdam

promotoren:	prof.dr. F.J. Bruggeman prof.dr. B. Teusink
copromoter:	dr. B.G. Olivier

Acknowledgments

> Alone we can do so little, together we can do so much.
>
> *Helen Keller*

This Ph.D. dissertation has only one name on its cover, but this dissertation is the result of the efforts of many people, for which I am grateful. In these acknowledgments I would like to thank everyone who contributed to my Ph.D. dissertation.

First and foremost, I would like to thank my promoters Bas and Frank—for advising, guiding, and stimulating my research—but also for giving me the freedom to work on many different topics. It has been a privilege to learn and grow under the supervision of both of you. I am also very grateful for giving me the opportunity to lecture during various courses and to supervise student internships. Frank, I also would like to say "thanks" for the countless times we discussed about other common interests: cycling, running, and playing tennis.

Next, I would like to thank my co-promoter Brett. Your knowledge about scientific software development and in particular of Python was of tremendous help. I would also like to thank all my co-authors with a special thanks to Anne, Joost, Mannus, and Meike for the pleasant and useful collaborations. Alone we can do so little, together we can do so much.

My gratitude also goes to my internship students (Coco, Daniel, Katy, Larissa, Laura, Maxim, Philip, Remco, Sylvie, and Thomas). I enjoyed supervising each of you. The progress that each of you made during your internship makes me proud. I am confident that each you will do great in the future—whether inside or outside the academic world. Katy and Maxim, I would like to highlight both of you briefly. Each of you did an outstanding internship. Katy, I included your stochastic modeling and simulation of a two-component signaling system as a prime example of why stochastic modeling and simulation can be extremely useful. Maxim, you laid the foundation of Chapter 8.

Many thanks to the colleagues from both the Life Sciences group at the Centrum Wiskunde & Informatica (CWI) and the Systems Bioinformatics group at the Vrije Universiteit Amsterdam for the pleasant four years. I would like to thank Mark in particular for critically reading many of my manuscripts prior to submission to a scientific journal. Oh, and by the way, Mark, I enjoyed watching you, *in my vintage ski-outfit*, making progress on the slopes of Axamer Lizum during the advanced lecture course on systems biology in

Innsbruck. I also enjoyed being part of the CWI running team; Peter thanks for organizing this. Gunnar, I will never forget your smile when we played tennis for the first time. This are the little things in life that make me feel great. Your enthusiasm is inspiring.

Drs. R.P. Machielsen, I would like to thank you for constantly motivating me to further develop your favorite software project StochPy.

While I should formally only thank those that contributed to this work, I also would like to thank all of my friends. And lets be honest, some of you had some direct influence on my work. For instance, Jasper, Nicolaas, Robbert, and Stef we had a great week in Playa del Inglés, but this holiday also delayed my research for about two additional days. Just because I was too tired of our adventures to be really productive. Fortunately, some of you also had a positive contribution to my work. In particular, I would like to thank Bas, Hetty, Leon, Jozien, Mark, and Robbert for reading and providing useful comments to the introductory chapter of this Ph.D. dissertation. I aimed at making the introductory chapter readable and understandable for a broad(er) audience. I think *we* succeeded!

Of course, I would like to thank my parents, Hetty and Ron, and sister, Susanne. Special thanks goes to my parents for their love, support, and encouragement. I would never have made it here without the both of you. Last but not least, Jozien thank you for coming into my life. You give me the strength to reach for the stars and chase my dreams. Your arms feel like home.

—Timo, August 25, 2015.

Contents

I Part I: Prologue 1

1 Essentially, all models are wrong, but some are useful 3

II Part II: Fluxes in Biochemical Models 13

2 A toolkit for analyzing fluxes in biochemical models: stoichiometric modeling & simulation 15

3 Developing Google Maps for metabolic highways 33

4 Interplay between constraints, objectives, and optimality for metabolic highways 53

5 The TimoTimo: a revolutionary "navigation system" for analyzing metabolic highways 75

III Part III: Fluctuations in Biochemical Models 87

6 A toolkit for analyzing fluctuations in biochemical models: stochastic simulation algorithms 89

7 Simulating fluctuations in biochemical models with StochPy 109

8 Cell growth and division in stochastic simulation algorithms 135

IV Part IV: Epilogue 163

9 Discussion and prospects 165

Bibliography 173

Summary 187

Samenvatting 191

Publications 195

Curriculum Vitae 197

Index 199

Part I

Prologue

1

Essentially, all models are wrong, but some are useful

1.1 Introducing the Key Players

I dedicate the title of the very first chapter of my Ph.D. dissertation to George E. P. Box (1919 – 2013) who was, besides a British statistician, one of the greatest minds of the 20th century and famous for saying:

> “Essentially, all models are wrong, but some are useful.”
>
> *George E. P. Box*

The models George E. P. Box was referring to are one of the key players in my Ph.D. dissertation about fluxes and fluctuations in biochemical models. To understand what George E. P. Box meant here, we first have to understand what kind of models he was referring to.

A Google (images) search mainly returns images of (beautiful) scantily dressed women. This matches one of the fifteen descriptions given by The Oxford English Dictionary: *“A person employed to wear clothes for display, or to appear in displays of other goods.”* The Oxford English Dictionary additionally writes: *“Originally used of women, and still usually understood in this sense unless preceded by a modifying word, as child, male, etc.”* Being a brilliant statistician George E. P. Box was most likely not referring to scantily dressed women in his book “Empirical Model-Building and Response Surfaces” (Box and Draper, 1987). In stead, he was referring to the following definition of a model given by The Oxford English Dictionary.

Key Player #1: Model

A simplified or idealized description or conception of a particular system, situation, or process, often in mathematical terms, that is put forward as a basis for theoretical or empirical understanding, or for calculations, predictions, etc.; a conceptual or mental representation of something.

These types of models are widely used for road maps (Fig. 1.1A), weather predictions (Fig. 1.1B), stock market predictions (Fig. 1.1C), drug discovery, and parking guidance systems of cars, for example.

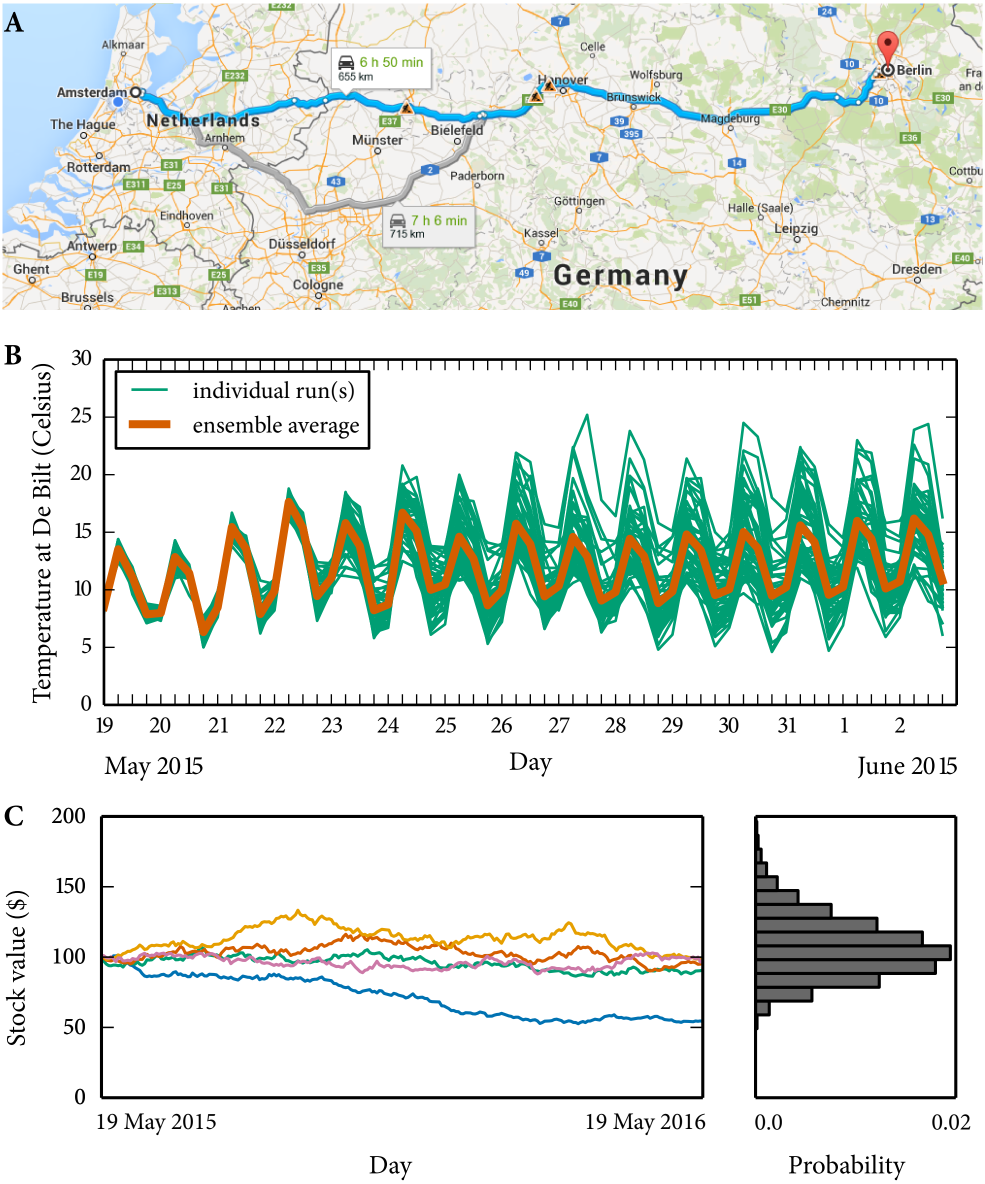


Figure 1.1. Examples of the use of modeling and simulation in "daily" life. (A) road maps can be used to predict optimal (blue) and suboptimal (grey) routes from a starting point (Amsterdam) to a destination (Berlin). (B) ensemble weather predictions at De Bilt (Netherlands). I would like to thank the KNMI (Koninklijk Nederlands Meteorologisch Instituut) for providing the requested data. (C) stock price simulation with Geometric Brownian Motion—the left panel shows five different simulations and the right panel shows the predicted stock price distribution at May 19th 2016 after running 50000 simulations.

Now that we know what kind of models George E. P. Box was referring to, we have to try to understand what he meant by "Essentially, all models are wrong, but some are useful". Each of these models is by definition a simplified description of reality, which is the nature of this type of models. This means that these models are by definition a simplified description and they are, therefore, always wrong. Should we, scientists, try and seek to get an as accurate as possible description of reality? We should not, and how can I better describe this than George E. P. Box who wrote:

> Since all models are wrong the scientist cannot obtain a "correct" one by excessive elaboration. On the contrary following William of Occam he should seek an economical description of natural phenomena. Just as the ability to devise simple but evocative models is the signature of the great scientist so overelaboration and overparameterization is often the mark of mediocrity.
>
> *George E. P. Box*

Being simplified and therefore wrong does not mean that they cannot be useful. Let me use road maps to illustrate this (and because I am going to use metabolic road maps later). Road maps are a simplified representation of the actual roads we drive our cars on. Creating an 1:1 scale road map would not only be cumbersome, but also impractical. Roads maps are further simplified by leaving out many details (see Fig. 1.1A), details that do not aid the driver to get from his starting point to his destination. This is what makes road maps useful.

The biochemical models I developed and used in this Ph.D. dissertation fall exactly into the category of models George E. P. Box was referring to. These biochemical models are used to study the chemical processes within living organisms. We can study this with the second key player of this Ph.D. dissertation: simulation, which is defined by the Oxford English Dictionary as follows:

Key Player #2: Simulation

The technique of imitating the behavior of some situation or process (whether economic, military, mechanical, etc.) by means of a suitably analogous situation or apparatus, esp. for the purpose of study or personnel training.

This means that simulations can be used to predict a certain behavior of your model. Road maps are, for example, exploited by navigation systems to predict the fastest or shortest route from a starting point to a destination (Fig. 1.1A).

These navigation systems use a set of mathematical operations to determine the fastest route from a starting point to a destination. Such a set of mathematical operations is called an algorithm, the third key player of this Ph.D. dissertation.

The Oxford English Dictionary defines an algorithm as follows:

Key Player #3: Algorithm

A procedure or set of rules used in calculation and problem-solving; (in later use spec.) a precisely defined set of mathematical or logical operations for the performance of a particular task.

This definition might give the impression that algorithms are something exclusively for computer scientists and mathematicians. These algorithms are indeed mainly developed by them, but we all profit from these algorithms in our daily lives. Traffic lights are controlled by algorithms and we use them every time we hit the search button on www.Google.com. Besides profiting from algorithms designed by computer scientists and mathematicians, we also develop and use our own algorithms in our daily lives. For example, we have recipes for cooking a specific meal, which also involves a set of rules that need to be followed. Cooking a frozen pizza can involve the following set of rules: remove all packaging and shrink wrap, preheat oven to 225 °C, bake pizza for 12-15 minutes in the center of the oven until the crust is light golden brown.

While we can perform simple algorithms by hand, simulating more advanced algorithms requires (automated) software packages: We do not need an advanced navigation system with fancy algorithms to tell us the fastest way from our desk to the coffee machine, but we probably need this navigation system to tell us the fastest way from Amsterdam to Berlin (Fig. 1.1A). This brings me to the fourth and final key player in my Ph.D. dissertation: software, which is defined by the Oxford English Dictionary as:

Key Player #4: Software

The programs and procedures required to enable a computer to perform a specific task, as opposed to the physical components of the system.

Software is indispensable in the era of the (personal) computer. We use anti-virus software, web-browser software, audio player software, and e-mail client software on a daily basis.

In the next section, I explain how we exploited these key players—models, simulation, algorithms, and software—in the emerging field of (computational) systems biology; a field where software tools are nowadays also indispensable (Ghosh et al., 2011). I close this section with another quote of George E. P. Box:

> “Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful.”
>
> *George E. P. Box*

1.2 Exploiting these key players in systems biology

Before we can start exploiting the key players of this thesis—models, simulation, algorithms, and software—in systems biology, we first have to understand what systems biology exactly is. Systems biology uses a "systems approach" to study biology. We know biology from high school as the study of life and living organisms and is defined by the Oxford English Dictionary as:

> **Biology**
>
> The branch of science that deals with living organisms as objects of study.

Biological systems are complex systems that display emergent properties which cannot be understood from a full understanding of the individual parts (Kitano, 2002b; Aderem, 2005). This reasoning is nicely illustrated by Kitano's example of an airplane: *"identifying all the genes and proteins in an organism is like listing all the parts of an airplane. While such a list provides a catalog of the individual components, by itself this list is not sufficient to understand the complexity underlying the engineered object."* We need a systems approach to obtain an understanding of the emergent properties of complex systems, thus we need a systems biology approach to obtain an understanding of the emergent properties of complex biological systems.

Of systems biology there exist, confusingly, many definitions. I present here three often used, but different, definitions of systems biology:

- *"Systems biology studies biological systems by systematically perturbing them (biologically, genetically, or chemically); monitoring the gene, protein, and informational pathway responses; integrating these data; and ultimately, formulating mathematical models that describe the structure of the system and its response to individual perturbations."* (Ideker et al., 2001)
- *"To understand complex biological systems requires the integration of experimental and computational research—a systems biology approach."* (Kitano, 2002a)
- *"The new discipline of systems biology examines how biological cell components interact and form networks and how the networks generate whole cell functions corresponding to observable phenotypes."* (Palsson, 2006)

Each of these definitions focuses on different aspects. Luckily, they have something in common: Systems biology uses different approaches to understand complex biological systems, not individual pieces.

The explicit definition of systems biology should not come from me, but if you ask me, I would define systems biology as:

Systems Biology

Systems biology is an integrated approach of computational, experimental, and theoretical methods that aims to understand complex biological systems in terms of the properties of its components and their interactions.

This is the definition of systems biology that I use throughout this Ph.D. dissertation. Now we know what systems biology is, we can start using the key players of this thesis—models, simulation, algorithms, and software—to understand complex biological systems. Despite that I defined systems biology as an integrated approach I focus on computational (and theoretical) approaches—computational systems biology.

Unraveling the mysteries of micro-organisms with computational systems biology

Computational systems biology approaches can be used to study a wide variety of complex biological systems. When applied to human health, these approaches can be used to study health and disease (Butcher et al., 2004). Cancer, for example, is argued to be a systems biology disease (Hornberg et al., 2006). This led to the development of a specific research field: cancer systems biology (Kreeger and Lauffenburger, 2010; Werner et al., 2014).

In this Ph.D. dissertation, I focus primarily on micro-organisms—

Micro-organism

An organism so small as to be visible only under a microscope; esp. a bacterium, fungus, or alga.

—of which bacteria, viruses, archaea, and some fungi and algae are examples. We have about 1.5 kg of these micro-organisms in and on us which corresponds to more than hundred trillion micro-organisms (Xu and Gordon, 2003; Ley et al., 2006); they are together called the human microbiota. Bacteria are the most abundant type of micro-organism in the human microbiota. The majority of these micro-organisms help us to stay healthy as may be illustrated by the following examples. The bacterium *Escherichia coli* protects the colon against harmful micro-organisms and produces vitamin K2. *Methanobrevibacter smithii* is an archaeum that aids the efficient digestion of polysaccharides in the human gut.

Despite the fact that we cannot see these micro-organisms by the naked eye, there exist numerous industrial applications of these micro-organisms. Food manufacturers exploit that micro-organisms are able to ferment, which not only is required to make the products we love most—e.g. beer, bread, chocolate, cheese, yogurt and wine—but additionally improves their storage life. In health care micro-organisms are used to produce antibiotics and vaccines.

More importantly, micro-organisms are responsible for the composition of our biosphere. They do this by (re)cycling the chemical elements that are a necessity for life, generating oxygen into the atmosphere and fixating nitrogen into a usable form (Dismukes et al., 2001; Canfield et al., 2010). This makes them the base of many food chains by converting nutrients and inorganic elements that other life forms cannot access. Cyanobacteria, for example, have been responsible for a significant fraction of the oxygen release in our atmosphere (Dismukes et al., 2001).

Cyanobacterium as example micro-organism

Nowadays, these cyanobacteria are of increased interest due to the world-wide fossil fuel scarcity. They are defined as:

Cyanobacterium

Any of a division of prokaryotic micro-organisms that contain chlorophyll (green) and phycocyanin (blue), produce free oxygen in photosynthesis, and occur widely in unicellular, filamentous, and colonial forms; also called blue-green alga.

Many different research facilities are currently studying the potential of cyanobacteria for harvesting solar energy for industrial applications. This Ph.D. study was carried out in the BioSolarCells fundamental research program: “Expanding society’s toolbox to harvest solar energy” with a specific cyanobacterium, *Synechocystis* PCC6803, as model organism. While most of the tools we developed are generic (except for the metabolic map of *Synechocystis* presented in Chapter 3), I use the cyanobacterium *Synechocystis* PCC6803 as an example micro-organism in my Ph.D. dissertation.

For industrial applications, cyanobacteria possess many advantages compared to other micro-organisms. The most important advantage is that they use photosynthesis to harvest solar energy for growth. These cyanobacteria have therefore the potential to be used as a cellular factory—fueled by sun light—for the direct conversion of carbon dioxide into compounds of interest to society (Angermayr et al., 2009; Ducat et al., 2011; Quintana et al., 2011; Wijffels et al., 2013; Savakis and Hellingwerf, 2014; Angermayr et al., 2015).

In the recent years, cyanobacterial strains have been developed that can produce reduced fuel compounds such as ethanol (Gao et al., 2012), L-lactic acid (Angermayr et al., 2012; van der Woude et al., 2014; Angermayr et al., 2014), 2,3-butanediol (Oliver et al., 2013; Savakis et al., 2013), glycerol (Savakis et al., 2015), and isobutyraldehyde & isobutanol (Atsumi et al., 2009), although no sufficient productivity has been found yet for large scale production of such reduced fuel compounds.

1.3 Outline of this thesis

By now, you should be able to grasp the aim of my Ph.D. dissertation—using models, simulation, algorithms, and software—to understand complex biological systems in terms of the properties of its components and their interactions. Besides using them to understand complex biological systems, we also want to make them available to the scientific community by developing useful software packages. In the remainder of this introductory chapter, I provide a brief overview of what is presented in the remaining chapters—this can be difficult to understand if you are not familiar with these modeling and simulation (M&S) approaches.

In computational systems biology, many different M&S approaches exists. Each M&S approach has its own advantages and disadvantages, thus we cannot say that one M&S approach is better than the other. A M&S approach can only be more suitable to resolve a specific question. For various scientific questions in the field of computational systems biology, Table 1.1 illustrates—for those interested or familiar with different mathematical methods—which approach is generally the most suitable.

	Model	Method	Examples
Optimal growth yields	stoichiometric	Flux Balance Analysis	Lewis et al. (2010)
Gene additions and deletions	stoichiometric	Flux Balance Analysis	Deutscher et al. (2006)
Network flexibility	stoichiometric	Elementary Flux Modes	Jol et al. (2012)
Metabolic engineering	stoichiometric	OptKnock	Yim et al. (2011)
Metabolic engineering	stoichiometric	Elementary Flux Modes	Trinh et al. (2008)
Optimal specific-flux states	kinetic	Elementary Flux Modes	Wortel et al. (2014)
Single-cell dynamics	stochastic	SSA	Elowitz et al. (2002)
Population dynamics	kinetic	ODEs	van Heerden et al. (2014)
Bistable dynamics	stochastic	SSA	Wei et al. (2014)
Structural dynamics of proteins	force field	Molecular Dynamics	Shaw et al. (2010)
Behavior of cellular structures	cellular Potts	Metropolis algorithm	Szabo and Merks (2013)

Table 1.1. Examples of mathematical methods that are often used to solve scientific questions. *Abbreviations: SSA: stochastic simulation algorithm, ODE: ordinary differential equation. Flux Balance Analysis (in depth discussed in Chapter 2) is most suited to predict optimal growth yields by using genome-scale stoichiometric models of metabolism as input. Stochastic simulation algorithms (SSAs) (in depth discussed in Chapter 6) are most suitable to predict single-cell dynamics, while classical systems of coupled ordinary differential equations (ODEs) are most suitable to predict population dynamics.*

I focus mainly on two important M&S classes: **stoichiometric modeling and simulation** to investigate fluxes in biochemical models (Part II) and **discrete stochastic modeling and simulation** to investigate fluctuations in biochemical models (Part III). Understanding these classes is not yet a requisite—both M&S classes are discussed in detail in Chapters 2 and 6. In short, stoichiometric simulations are deterministic (the output of a specific simulation is fully determined by a set of constraints), whereas stochastic simulations possess inherent randomness (the same set of initial conditions and parameter values leads to an ensemble of different outputs). Fig. 1.2 illustrates the outcome of both M&S classes; note the striking resemblance between these simulations and the examples of daily life simulations shown in Fig. 1.1:

(i) Stoichiometric M&S can be used to predict optimal routes in biochemical models. In Fig. 1.2A, we predict the optimal biochemical route to synthesize glycogen from light which is comparable to predicting the optimal (fastest) route from Amsterdam to Berlin (Fig. 1.1A).

(ii) Discrete stochastic M&S can be used to predict time dynamics and probability distributions of stochastic processes. In Fig. 1.2B, we predict the behavior of molecule X in a specific biological system which is comparable to predicting the future price of a specific stock (Fig. 1.1C).

More specifically, I use stoichiometric M&S to predict and analyze steady-state fluxes in biochemical models (the first part of the title). All processes in nature are, however, fundamentally stochastic. While this stochasticity is often negligible in the macroscopic world because of the law of large numbers, considerable experimental evidence indicates that significant stochastic fluctuations are present and essential. I therefore use discrete stochastic M&S to investigate fluctuations in biochemical models (the second part of the title).

I start both Part II and III by discussing the start-of-the-art mathematical methods for both stoichiometric and discrete stochastic simulations in Chapters 2 and 6, respectively. We developed specific software packages that aid the community in understanding complex biological systems (without fragmenting them into pieces). These software packages are VoNDA (Visualization of Network DAta), CoPE FBA 2.0 (Comprehensive Polyhedron Enumeration FBA), and StochPy (Stochastic modeling and simulation in Python) which are in detail described and illustrated in Chapters 3, 5, and 7, respectively. Each of these software packages is (mainly) developed in Python. Python is a user-friendly, freely available, and powerful programming language that is currently a popular programming language in systems biology. This is demonstrated by the collection of systems biology tools that have been developed in Python during the last years (e.g. PySCeS (Olivier et al., 2005), ScrumPy (Poolman, 2006), CBMPy (Olivier, 2011), PySB (Lopez et al., 2013), and COBRApy (Ebrahim et al., 2013)).

In short, we developed VoNDA as a data integration and visualization resource for metabolic models. As input VoNDA requires a graphical map of metabolism in Support Vector Graphics, which is what we developed and illustrate for *Synechocystis* in Chapter 3. We also developed a plug-in to FAME, the first web-based Flux Analysis and Modeling Environment. In Chapter 4 we first present a better understanding of the interplay between constraints, objectives, and optimization for genome-scale stoichiometric models of metabolism. We subsequently present in Chapter 5 an efficient method, CoPE FBA 2.0, to enumerate optimal solution spaces of these genome-scale stoichiometric models.

In Chapter 7 we present StochPy as our tool of choice for stochastic simulation of molecular control models inside living cells. We extended StochPy by incorporating explicit cell growth and division events which we further describe and illustrate in Chapter 8.

I conclude with a discussion in which I elaborate on the importance of the key players—models, simulation, algorithms, and software—in the emerging field of systems biology, with a special focus on understanding fluxes and fluctuations in biochemical models.

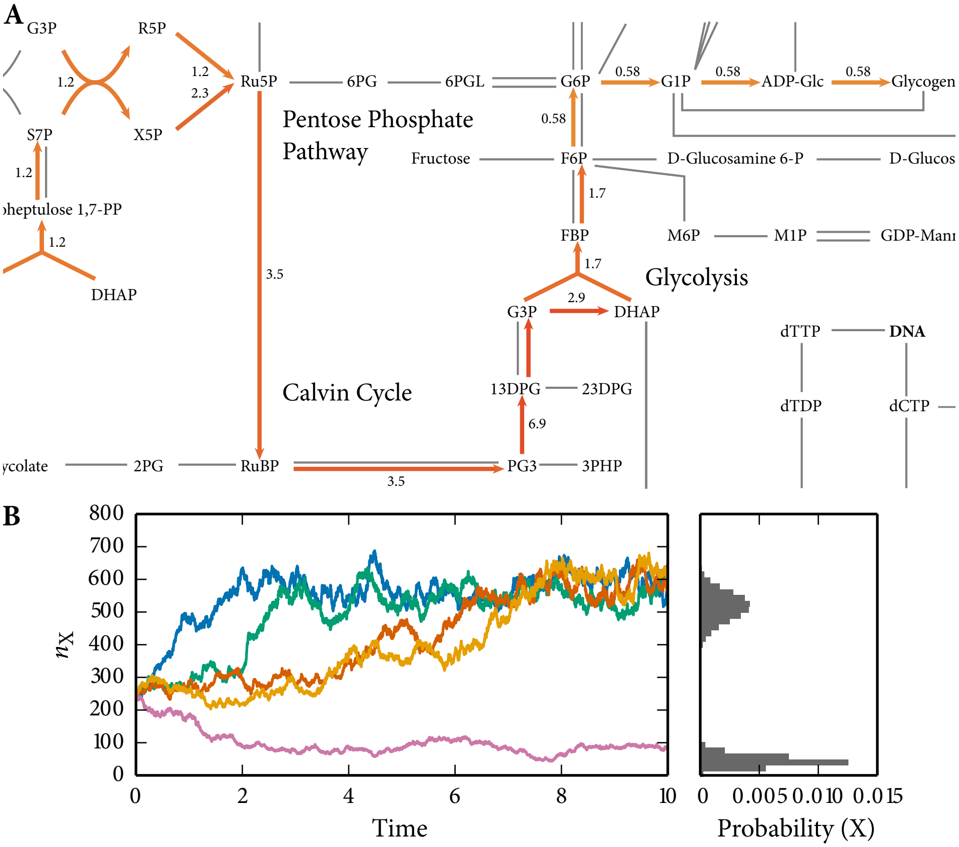


Figure 1.2. Graphical representation of stoichiometric and discrete stochastic simulations. *(A) flux balance analysis predicts steady-state fluxes in Synechocystis' central carbon metabolism (Chapter 3). (B) stochastic simulation of the number of X molecules (n_X) in the Schlögl reaction system—left panel shows five different time trajectory simulations and the right panel shows the n_X distribution at $t = 10$ after running 250000 simulations (Chapter 6).*

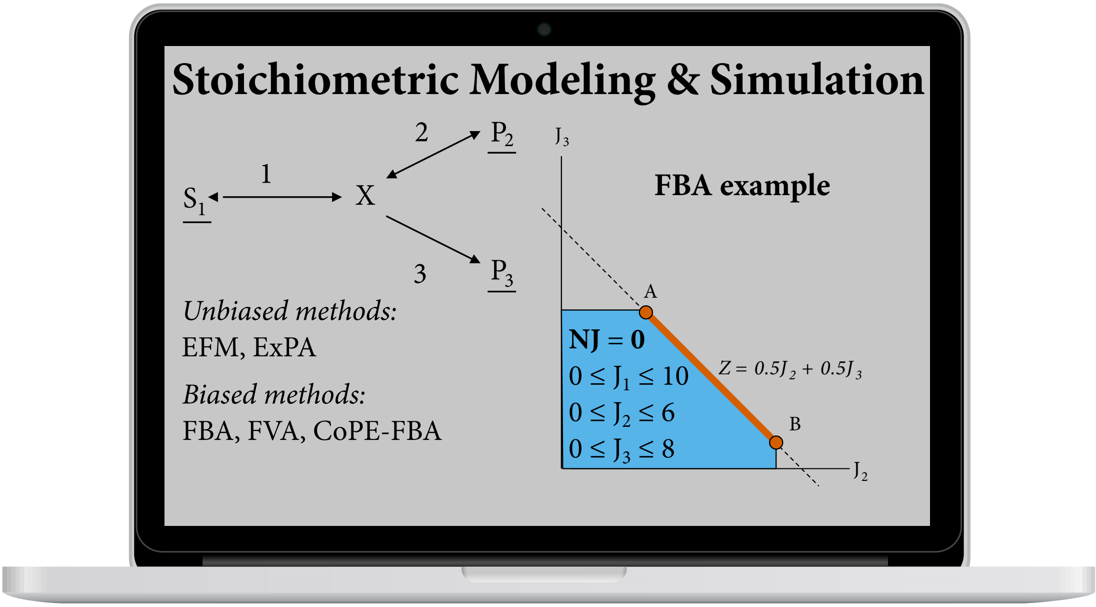
Part II

Fluxes in Biochemical Models:

Stoichiometric Modeling & Simulation

This chapter is based on the publication:

Maarleveld T.R, Khandelwhal R., Olivier B.G., Teusink B., and Bruggeman F.J. (2013) *"Basic concepts and principles of stoichiometric modeling of metabolic networks"*, **Biotechnology Journal**, 8, 9:997-1008.



Stoichiometric Modeling & Simulation

Unbiased methods:
EFM, ExPA

Biased methods:
FBA, FVA, CoPE-FBA

FBA example

NJ = 0
$0 \le J_1 \le 10$
$0 \le J_2 \le 6$
$0 \le J_3 \le 8$

$Z = 0.5J_2 + 0.5J_3$

Stoichiometric Modeling & Simulation: A widely used predictive approach to study metabolism at the genome-scale level. Various mathematical methods exist of which Flux Balance Analysis (FBA) is the most popular method.
The above example shows using FBA to optimize the objective function Z given the constraints.

A toolkit for analyzing fluxes in biochemical models: stoichiometric modeling & simulation

2

Abstract

Metabolic networks supply the energy and building blocks for cell growth and maintenance. Cells continuously rewire their metabolic networks in response to changes in environmental conditions to sustain fitness. Studies of the systemic properties of metabolic networks give insight into metabolic plasticity and robustness, and the ability of organisms to cope with different environments. Stoichiometric modeling and simulation of metabolic networks has become an indispensable tool for such studies. Herein, we review the basic theoretical underpinnings of stoichiometric modeling and simulation of metabolic networks. Basic concepts, such as stoichiometry, chemical moiety conservation, flux modes, flux balance analysis, and flux solution spaces, are explained with simple, illustrative examples. We emphasize the mathematical definitions and their network topological interpretations.

2.1 Introduction

Metabolism is essentially a large network of coupled chemical conversions (reactions) catalyzed mostly by enzymes. In this process, nutrients are converted into building blocks, such as nucleotides, fatty acids, lipids, amino acids, and free-energy carriers, for the synthesis of macromolecules, such as DNA, RNA, and proteins. These macromolecules are required for the maintenance of cellular integrity and formation of new cells. Fundamental processes in metabolism are enzyme-catalyzed reactions. In a single reaction, substrates are converted into products and the number of atoms of a given type, such as C, H, O, N, P, or S, and the net charge should balance on each side of the equation (Roels, 2009).

These balancing principles are followed in genome-scale stoichiometric models (GSSM) of metabolism (Thiele and Palsson, 2010). These GSSMs are defined as:

Genome-scale stoichiometric models

Models that are solely based on stoichiometry (not the kinetics) and that cover the total (metabolic) potential encoded in the genome.

Some aspects of balancing remain ambiguous, such as the protonation states of some of the metabolites, because this may be dependent on intracellular properties, such as pH and ionic strength. Every reaction occurs at a rate that depends on the concentrations of the enzyme reactants, possibly a few effectors, and the enzyme kinetic properties described by enzyme kinetics. Any reaction j can be written as

$$\sum_{i=1}^{m} n_{ij}^{-} x_i \longrightarrow \sum_{i=1}^{m} n_{ij}^{+} x_i \tag{2.1}$$

in which we consider a network with a total of m metabolites (reactants), denoted by x_i. The n_{ij}^{+} and n_{ij}^{-} coefficients denote product and substrate stoichiometric coefficients respectively, and equal the number of molecules produced and consumed per unit reaction rate. We define v_j as the reaction rate and typical units are mM min^{-1} or mmol h^{-1} (g biomass)$^{-1}$. The net stoichiometric coefficient of metabolite i in reaction j is defined as $n_{ij} = n_{ij}^{+} - n_{ij}^{-}$.

The rates of change of the concentration of every metabolite can be equated in terms of reaction rates and net stoichiometric coefficients, which gives rise to the following set of ordinary differential equations:

$$\frac{d}{dt}\boldsymbol{x}(t, \boldsymbol{p}) = \boldsymbol{N}\boldsymbol{v}(\boldsymbol{x}(t, \boldsymbol{p}, \boldsymbol{p}). \tag{2.2}$$

Here the metabolite or state vector $\boldsymbol{x}$ is $m \times 1$ in dimension and the $r \times 1$ rate vector, $\boldsymbol{v}$, contains the rate equations of the r reactions in the network, which are typically expressed in terms of enzyme kinetics. The stoichiometric matrix $\boldsymbol{N}$ is $m \times r$ in dimensions and contains as its i, j-th entry the net stoichiometric coefficient, n_{ij}, of metabolite i in reaction j. The coefficient $n_{ij} < 0$ if metabolite i is a substrate in the net stoichiometry of reaction j and $n_{ij} > 0$ if metabolite i is a product in the net stoichiometry of reaction j. The kinetic and environmental parameters are elements of the vector $\boldsymbol{p}$ and t denotes time. Metabolites that are held at a fixed concentration (boundary metabolites) do not enter the stoichiometry matrix because they do not have a rate of change. They enter as parameters in the parameter vector $\boldsymbol{p}$.

Because our interest is in stoichiometric models, we do not discuss further enzyme kinetics that enter the rate vector $\boldsymbol{v}$, see Cornish-Bowden (2013). The stoichiometric matrix is the principle object of study in stoichiometric modeling and simulation. Herein, we discuss basic analyses of the stoichiometric matrix.

2.2 Chemical moiety conversation

In metabolism, metabolites tend to occur that are solely recycled. Examples of such metabolites include ATP, NAD(P)H, and coenzyme A. As a consequence of recycling, the maximum concentration of those metabolites is constrained by a total concentration of a chemical moiety. For instance, in the case of phosphate and adenosine moiety conservation, the relationships

$$\begin{aligned} P_T &= 3\ ATP + 2\ ADP + AMP + P, \\ A_T &= ATP + ADP + AMP, \end{aligned} \tag{2.3}$$

hold true at any moment in time with total phosphate and adenosine levels given as P_T and A_T. Taking the derivative of Eq. (2.3) with respect to time yields

$$\begin{aligned} 0 &= 3\ \frac{dATP}{dt} + 2\ \frac{dADP}{dt} + \frac{dAMP}{dt} + \frac{dP}{dt}, \\ 0 &= \frac{dATP}{dt} + \frac{dADP}{dt} + \frac{dAMP}{dt}, \end{aligned} \tag{2.4}$$

which indicates the linear relationships between the rows of the stoichiometry matrix and allows for the expression of the rate of change of one metabolite in terms of other rates of change (Hofmeyr, 2001). In matrix form, we can write this as

$$\frac{d\boldsymbol{x^D}}{dt} = \boldsymbol{L_0}\frac{d\boldsymbol{x^I}}{dt} \tag{2.5}$$

for the general case of metabolites of a metabolic network. The vectors $\boldsymbol{x^D}$ and $\boldsymbol{x^I}$ denote the vector of dependent and independent metabolite concentrations. The matrix $\boldsymbol{L_0}$ expresses the linear combinations of the rates of changes of the independent metabolites. In integrated form, Eq. (2.5) becomes

$$\boldsymbol{x_D} - \boldsymbol{L_0 x_I} = \boldsymbol{t} \tag{2.6}$$

with $\boldsymbol{t}$ as the vector of total concentrations of chemical moieties. By way of illustration, for the moiety-conservation relationships given in Eq. (2.3) the vectors $\boldsymbol{x_D}$ and $\boldsymbol{x_I}$ and the matrix $\boldsymbol{L_0}$ are given in the following relationship:

$$\frac{d}{dt}\begin{pmatrix} AMP \\ P \end{pmatrix} = \begin{pmatrix} -1 & -1 \\ -2 & -1 \end{pmatrix} \frac{d}{dt}\begin{pmatrix} ATP \\ ADP \end{pmatrix}. \tag{2.7}$$

Using this $\boldsymbol{L_0}$ matrix, the dynamics of all metabolites (ATP, ADP, AMP, and P) can be obtained from the dynamics of the independent metabolites (ATP, ADP). In other words, the dependent species are redundant for determining the species dynamics. Different combinations of independent metabolites can be chosen. This can intuitively be seen from the relationships given in Eq. (2.3). For example, by choosing ATP and AMP as independent metabolites, the concentration of ADP can be determined from $A_T = ATP + ADP + AMP$. Subsequently, the concentration of P can be determined from $P_T = 3\ ATP + 2\ ADP + AMP + P$.

The relationship given in Eq. (2.5) dictates the decomposition of the stoichiometric matrix into two blocks as is given by

$$\frac{d}{dt}\begin{pmatrix} \boldsymbol{X_I} \\ \boldsymbol{X_D} \end{pmatrix} = \boldsymbol{N v} = \begin{pmatrix} \boldsymbol{N_R} \\ \boldsymbol{N_0} \end{pmatrix} \boldsymbol{v}, \tag{2.8}$$

in which $\boldsymbol{N}$ is decomposed into blocks of $\boldsymbol{N_R}$ (the reduced stoichiometry matrix) and $\boldsymbol{N_0}$ (Kholodenko et al., 1994; Reder, 1988). Together, the relationships shown in Eqs. (2.5) and (2.7) give rise to

$$\boldsymbol{L_0 N_R} = \boldsymbol{N_0} \rightarrow (-\boldsymbol{L_0 I}) \begin{pmatrix} \boldsymbol{N_R} \\ \boldsymbol{N_0} \end{pmatrix} = \boldsymbol{0} \rightarrow \begin{pmatrix} \boldsymbol{N_R} \\ \boldsymbol{N_0} \end{pmatrix}^T (-\boldsymbol{L_0 I})^T = \boldsymbol{0}, \tag{2.9}$$

and indicate that the moiety conservation matrix in Eq. (2.10),

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{L_0} \end{pmatrix}, \tag{2.10}$$

can be derived from the left null-space of the stoichiometry matrix (Famili and Palsson, 2003; Sauro and Ingalls, 2004). Typically, $\boldsymbol{N_R}$ is identified in $\boldsymbol{N}$ after the null-space is calculated. The number of independent metabolites, m_0, is denoted by the rank of $\boldsymbol{N}$, thus the reduced stoichiometry matrix N_R is $m_0 \times r$ in size. This indicates that the stoichiometry matrix $\boldsymbol{N}$ has m_0 independent rows and $m - m_0$ moiety-conservation relationships.

In GSSMs, a biomass reaction is typically used to describe cell growth. This biomass reaction is used as a sink for biomass precursors (e.g. DNA, RNA, proteins, and lipids) that together define the biomass composition of the cell. These biomass precursors contain moieties, such as adenosine, that require the continuous synthesis of adenosine. Therefore, a non-zero flux through this biomass reaction results in a drain of the moieties. Hence, in such a GSSM, a strict application of moiety conservation detection along the lines detailed above, results in fewer moieties. Yet, to understand the dynamics of metabolic pathways, they are relevant because the turnover of ATP is much greater than the rate at which the adenosine moiety is synthesized.

2.3 Steady-state flux modes

By definition, at a steady state of the metabolic network the following relationship holds:

$$\boldsymbol{N_R J} = \boldsymbol{0} \tag{2.11}$$

where we used the convention that the reaction rate vector, $\boldsymbol{v}$, at steady state is denoted by $\boldsymbol{J}$, which is the flux vector. Eq. (2.11) gives rise to m_0 flux relationships, each of which represent a linear combination between fluxes. As a consequence, $r - m_0$ fluxes are minimally required to determine all fluxes. Hence, Eq. (2.12) must exist because it describes all linear combinations of independent fluxes ($\boldsymbol{J^I}$) that give rise to the dependent fluxes in $\boldsymbol{J^D}$:

$$\boldsymbol{J^D} = \boldsymbol{K_0 J^I} \rightarrow \boldsymbol{J} = \begin{pmatrix} \boldsymbol{J^I} \\ \boldsymbol{J^D} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{K_0} \end{pmatrix} \boldsymbol{J^I} \rightarrow \boldsymbol{N_R} \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{K_0} \end{pmatrix} = \boldsymbol{0}. \tag{2.12}$$

This means that the right null-space of the (reduced) stoichiometry matrix equals the kernel matrix, given by

$$\boldsymbol{K} = \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{K_0} \end{pmatrix}. \tag{2.13}$$

The columns of $\boldsymbol{N_R}$ may have to be reordered to write the null-space in this form. In addition, each column of $\boldsymbol{K}$ can be divided by any number; all resulting vectors continue to lie in the null-space of $\boldsymbol{N_R}$. The rows of $\boldsymbol{K}$ represent the flux values of a specific reaction.

If we denote the j-th column of $\boldsymbol{K}$ by $\boldsymbol{k}_j$, any flux vector $\boldsymbol{J}$ can be written as a linear combination of the columns of $\boldsymbol{K}$, as given by

$$\boldsymbol{J} = \sum_{j=1}^{r-m_0} \alpha_j \boldsymbol{k}_j \tag{2.14}$$

in which the weighting coefficients, α_j, can take any value. The set of all flux vectors of the metabolic network is contained within the null-space of $\boldsymbol{N_R}$. However, this is a huge space and below we discuss definitions that reduce this space by incorporating additional thermodynamic information or postulating optimal metabolic functioning. Eq. (2.14) illustrates that the definition of $\boldsymbol{K}$ cannot be unique because multiplication of the multipliers α_j by any factor λ can be compensated for by the division of every element in $\boldsymbol{k}_j$ by λ. Because no restrictions apply to the values of the multipliers, $\boldsymbol{K}$ cannot be uniquely chosen.

The vectors $\boldsymbol{k}_j$ have a network topological interpretation. They represent routes through the network along which every metabolite is at steady state, if the fluxes carry the values dictated by $\boldsymbol{k}_j$. This is why the $\boldsymbol{k}_j$ terms are often called flux modes. In Fig. 2.1A, a toy metabolic network is shown which contains 26 reactions and 23 metabolites. The external metabolites T, U, X, and Y are considered to be fixed; the other 19 metabolites are considered to be variable. The stoichiometry matrix has full rank and, therefore, no conserved chemical moieties occur. The number of independent fluxes equals seven (=26-19). Or in other words, seven flux modes exist and they are displayed in Fig. 2.1B. The color codes of the reactions indicate reaction rate values. Because all of these metabolites along a flux mode are required to operate a steady state, the flux modes have to be either cycles or routes from source to sink metabolites.

The flux modes do not necessarily agree with thermodynamics, since several irreversible reactions are forced to have a negative flux. In addition, the flux mode in the upper-right part of Fig. 2.1B is complex and could be swapped for a simpler flux mode if desired. This indicates the problems associated with analysis of the null-space and explains why alternative definitions for steady-state flux routes in metabolic networks have been developed; these are introduced and discussed in the upcoming sections of this chapter (2.4–2.9). These alternative definitions are unique representations of the null-space and agree with the thermodynamic preference of the reactions.

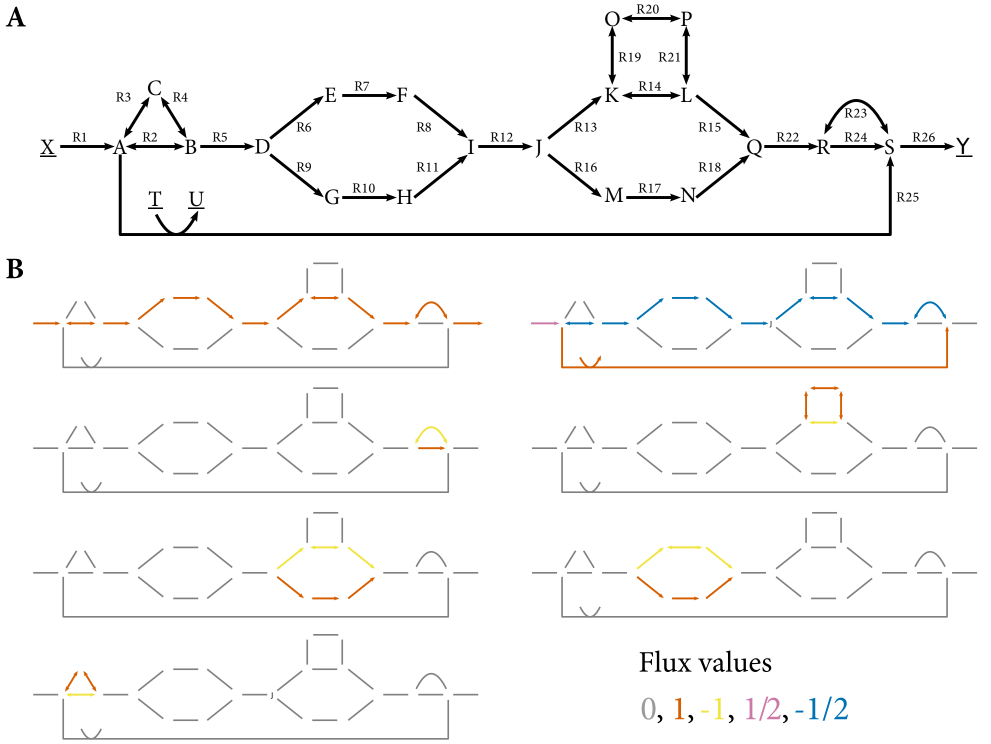


Figure 2.1. A simplified metabolic pathway to illustrate the concept of flux modes. *(A) a network diagram of a simplified metabolic network. Arrows indicate reactions and are labeled as R1, R2, …, R26. Double-headed arrows indicate reversible reactions. Irreversible reactions are indicated by single-headed arrows, which point in the thermodynamically preferred direction. Underlined metabolites are considered to be fixed in concentration to allow for a steady state. All reactions are uni-uni reactions, except R25, which has stoichiometry of* $A + T \rightarrow U + 0.5\ S$. *We can rewrite this stoichiometry as* $A_2 + P \rightarrow AP + A$ *to illustrate that there is no stoichiometric inconsistency with the isomerization reactions. To deal with thermodynamic inconsistencies, imagine adding fixed metabolites V and W to R24 to drive this reaction forward. A description of this model in the SBML level 3 format can be found in the Supplemental Dataset S1. (B) an overview of the seven flux modes (colors correspond to flux values).*

2.4 Flux Balance Analysis

In Section 2.3, we discussed the steady-state relationship $\boldsymbol{N_R}\,\boldsymbol{J} = \boldsymbol{0}$. We illustrated that this system of equations was underdetermined; more unknown fluxes occurred than the number of linear relationships ($r > m_0$). Therefore, the null-space of the stoichiometric matrix $\boldsymbol{N}$ did not lead to a unique flux vector, but a whole solution space. To realistically narrow down the solution space, Flux Balance Analysis (FBA) is a popular method.

Flux Balance Analysis

A linear programming approach to optimize (maximize or minimize) an objective function subject to the steady-state constraint, thermodynamic constraints, and capacity constraints

FBA selects only those flux values that together can optimize some biologically relevant objective, such as maximum biomass rate or maximum ATP production rate (Schuetz et al., 2007). This optimization is achieved by a linear programming approach (Fell and Small, 1986; Savinell and Palsson, 1992) and FBA can be mathematically represented by

$$\begin{aligned} \text{Maximize or Minimize } Z_{obj} &= \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{J} \\ \text{subject to } \boldsymbol{N}\boldsymbol{J} &= \boldsymbol{0} \\ \boldsymbol{J}^{\boldsymbol{min}} \leq \boldsymbol{J} &\leq \boldsymbol{J}^{\boldsymbol{max}}, \end{aligned} \tag{2.15}$$

in which vector $\boldsymbol{c}$ dictates a linear relationship between fluxes of $\boldsymbol{J}$ that forms the objective function Z. The $\boldsymbol{J}^{\boldsymbol{min}}$ and $\boldsymbol{J}^{\boldsymbol{max}}$ are vectors of minimum and maximum values respectively, that any flux of vector $\boldsymbol{J}$ can attain during this optimization. These flux bounds can represent experimental measurements by bounding all known flux values within experimental errors or they can derive from thermodynamic considerations that force fluxes to be either strictly negative or positive. This linear program narrows down the feasible steady-state flux space of the stoichiometric matrix by applying stoichiometric, thermodynamic, and environmental constraints and by optimizing an objective function. Hence, this linear program results in an optimal solution space,

Optimal solution space

The set of flux distributions that, in combination, give a unique and maximum value for the objective function Z.

Mathematically, the optimal solution space can be written as

$$F_{opt} = \{\boldsymbol{J} : \boldsymbol{N}\boldsymbol{J} = \boldsymbol{0}, \boldsymbol{J}^{\boldsymbol{min}} \leq \boldsymbol{J} \leq \boldsymbol{J}^{\boldsymbol{max}}, \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{J} = opt\}, \tag{2.16}$$

which means that each flux distribution $\boldsymbol{J}$ must be in steady state ($\boldsymbol{NJ} = \boldsymbol{0}$), satisfy the flux bounds ($\boldsymbol{J^{min}} \leq \boldsymbol{J} \leq \boldsymbol{J^{max}}$), and satisfy optimality ($\boldsymbol{c^T J} = opt$). This optimal solution space is considerably smaller than the space dictated by $\boldsymbol{K}$.

Performing FBA on the toy metabolic model shown in Fig. 2.1A with the input flux R1 $\leq$ 1, and maximization of the flux through R26 as the objective function yields a flux distribution, one of which is presented in the upper-left part of Fig. 2.1B. This example indicates that the flux distribution that results from FBA can be (a linear combination of the) flux modes of the stoichiometric matrix. We also performed FBA on the iTM686 model of *Synechocystis*, further described in Chapter 3. Input fluxes represent a light limited growth condition (photon and HCO_3^- maximum uptake fluxes of 3.7 and 30 mmol gDW^{-1} h^{-1} respectively) and free exchange of ammonia, water, oxygen, protons, phosphate, sulfate, and other metal ions. As the metabolic objective, we considered maximization of the rate of biomass synthesis. This optimization predicts a growth rate of 0.052 h^{-1} and a flux distribution in which 40.0% reactions are inactive; about 60.0% reactions of the whole metabolic network are used.

2.5 Flux Variability Analysis

A useful tool to gain an insight into network flexibility is Flux Variability Analysis (FVA):

Flux Variability Analysis

A linear programming approach to maximize and minimize each flux of the metabolic network, while satisfying all given constraints at the optimal objective function value (Mahadevan and Schilling, 2003).

FVA gives the span of the fluxes that exist within the optimal solution space defined by the linear program given in Eq. (2.15) . The linear program for FVA is given by

$$\begin{aligned} \text{Maximize or Minimize } J_j \qquad & \text{for } j{=}(1, 2, \ldots, r) \\ \text{subject to } \boldsymbol{NJ} &= \boldsymbol{0} \\ \boldsymbol{c^T J} &= Z_{obj} \\ \boldsymbol{J^{min}} \leq \boldsymbol{J} &\leq \boldsymbol{J^{max}}, \end{aligned} \tag{2.17}$$

in which Z_{obj} is the objective function value of the previous linear program (Eq. (2.15)). By fixing the objective function value at the value obtained from the FBA optimization, FVA determines the range for each flux within which all numerical values are valid FBA solutions. The results of FVA optimization can be used to determine the spans ($= |J_j^{FVA\ max} - J_j^{FVA^{min}}|$) as an absolute difference between the FVA maximum ($J_j^{FVA\ max}$) and FVA minimum ($J_j^{FVA\ min}$) values of each flux J_j within the optima. On the basis of these spans,

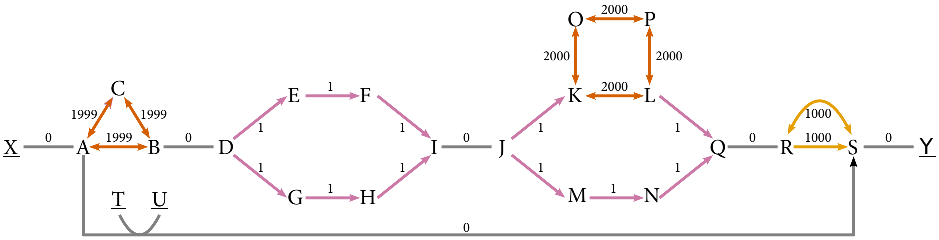


Figure 2.2. FVA performed on the toy metabolic model. *Resulting spans are shown for all reactions in which zero span is for fixed (in-)active reactions (gray, R25 is inactive), a span of one is for active but variable reactions (purple), and large spans are for reactions (red, orange) in cycles.*

we can determine the fixed and flexible parts of the metabolic network while it achieves a particular metabolic objective. These spans can hit infinity because, in an optimal flux distribution of the metabolic network, some reaction rates may not be constrained at all. The FVA span gives an indication of the range of values that a reaction may attain. However, the actual value a reaction can take in a particular flux distribution depends on the entire reaction network: Fluxes cannot be changed independently because this would violate the steady-state constraint (Burgard et al., 2004).

Fig. 2.2 shows the absolute spans resulting from FVA for all fluxes in the toy metabolic model. The constraints for this FVA are identical to the FBA calculations for this model discussed above and infinity flux bounds are represented by a large value of 1000. This analysis depicts the reactions that have a fixed flux and a span of zero (gray arrows) and reactions that have variable fluxes with a span of one (purple arrows). Reactions with a span of zero are either active (essential) reactions because alternative optimal paths are not present or inactive (non-essential) reactions because they yield a sub-optimal FBA solution (R25). Some reactions (red arrows) have large spans because they are part of metabolic cycles (R2–R4, R14 and R19–R21, and R23–R24). In any optimal FBA solution with a maximal flux of one through R1, a net flux one is required from R1 to R5. Therefore, the allowed flux values of R2–R4 are between -999 and 1000 (span = 1999); a flux of -999 through R2 results in a flux of 1000 through R3 and R4 and vice versa. In contrast, to obtain an optimal FBA solution no net flux is required through the second metabolic cycle (R14 and R19–R21). For instance, a flux of one through R16–R18 results in no flux through R13–R15 in any optimal FBA solution. The reactions of this metabolic cycle can, therefore, operate at their maximum bounds in both directions without violating optimal metabolic functioning (span = 2000). Also, a net flux of one is required from R22 to R26. Because R24 is irreversible, the maximal flux R23 can obtain in any optimal solution is one (with $J_{R24} = 0$). Since R23 is reversible, an optimal FBA solution can be obtained if the fluxes through R23 and R24 are -999 and 1000 respectively. The spans of R23 and R24 are thus 1000 (R23: -999 to 1 and R24: 0–1000).

We also performed FVA on the *Synechocystis* model iTM686 with the same constraints and objective function value as the FBA optimization described above. Analyzing the spans of all fluxes revealed that only 9.1% reactions could vary in the optimal solution space. Of the 90.9% fixed reactions, 59.7% carry a non-zero flux and the remaining 40.3% are inactive. This means that 63.3% (9.1% + 0.909 × 59.7%) reactions can have a non-zero flux. The percentage of non-zero fluxes in an FBA outcome is equal or less, because some variable fluxes can also be zero in an FBA solution.

2.6 Interpretation of the sensitivity parameters associated with an FBA solution

Two sensitivity parameters—reduced costs and shadow prices—are associated with an FBA solution. A reduced cost (r_j) can be interpreted as:

Reduced cost

The sensitivity of the objective function with respect to the change in the j-th flux value.

In biological terms, this can be interpreted in the following manner: If a flux J_j has a reduced cost of r_j in a particular FBA solution and this flux value is increased by δJ_j, then the objective function value is changed to $Z + r_j \delta J_j$. Reduced costs assigned to nutrient uptake fluxes give us an indication of the growth-limiting compounds in the medium. The reduced costs assigned to the uptake fluxes of substrates, that are not allowed to be consumed, identify which nutrients could be added to the medium to achieve a higher growth rate (Goffin et al., 2010). Sometimes, inactive substrate fluxes are of no interest and scaled reduced costs (sr_j) (Teusink et al., 2006) are used to identify the limiting substrates. Scaled reduced costs can be represented by

$$sr_j = \frac{r_j J_j}{Z}. \tag{2.18}$$

A shadow price (γ_i) is defined as:

Shadow price

The sensitivity of the objective function with respect to the change in a constraint (Palsson, 2006).

Consequently, if metabolite i is added, then objective function value Z changes according to the shadow price. Shadow prices have been used to analyze the effects of substrate availability on the growth in phenotypic phase plane analysis (Price et al., 2004).

2.7 Elementary Flux Modes

Elementary flux modes (EFMs) and extreme pathways (ExPas) were developed to uniquely characterize the right null-space $\boldsymbol{K}$ of a stoichiometry matrix. These EFMs are defined as:

Elementary flux modes

The non-decomposable steady-state pathways in metabolic models.

In contrast to FBA-related techniques, EFM (and ExPa) analyses are only based on model stoichiometry and therefore allow an unbiased analysis without imposing an optimization principle. These definitions rely on a convex set of flux vectors (Schuster et al., 1999; Schilling et al., 2000). By taking a convex combination of these flux vectors ($\boldsymbol{e}_n$), any possible steady-state flux distribution ($\boldsymbol{J}$) can be generated. Assuming that we have K EFMs, we can write this as

$$\boldsymbol{J} = \sum_{k=1}^{K} \alpha_k \boldsymbol{e}_k \tag{2.19}$$

where α_k are non-negative weighting coefficients that total one,

$$\sum_{k=1}^{K} \alpha_k = 1 \text{ with } (\alpha_k \geq 0). \tag{2.20}$$

EMFs (and ExPas) can be exploited to evaluate, for instance, pathway redundancy, to find (sub-)optimal pathways for the investigation of pathway properties, such as cost and length, and to study the effect of gene deletions (Price et al., 2002; Wiback and Palsson, 2002; Papin et al., 2002). Unfortunately, both approaches suffer from excessive running times, that is, characterizing the right null-space of the stoichiometric matrix is a non-deterministic polynomial-time (NP)-hard computational problem.

EFMs fulfill the following three conditions: (pseudo-)steady state, thermodynamic feasibility, and non-decomposability (Schuster et al., 1999). These conditions have three implications. First, internal metabolites of an EFM are neither net consumed or produced due to the steady-state condition. Second, all flux rates of an EFM are thermodynamically feasible in contrast to the flux modes, $\boldsymbol{k}_j$. And third, no subset of an EFM exists that fulfills the first two conditions without violating the third condition. The complete set of EFMs can be partitioned into three types: (i) all optimal yield pathways converting one or more substrates into a product, (ii) all sub-optimal yield pathways converting one or more substrates to a product, and (iii) internal loops in the metabolic model.

The toy metabolic network shown in Fig. 2.1A contains 28 EFMs, as shown in Fig. 2.3. In this toy model, we can identify all three types of EFMs earlier defined. To begin with, there are 24 type I EFMs that characterize all optimal pathways, which are shown in Fig. 2.3A–X. Any route from metabolite X to Y, ignoring reaction R25, is a type I EFM. In this network,

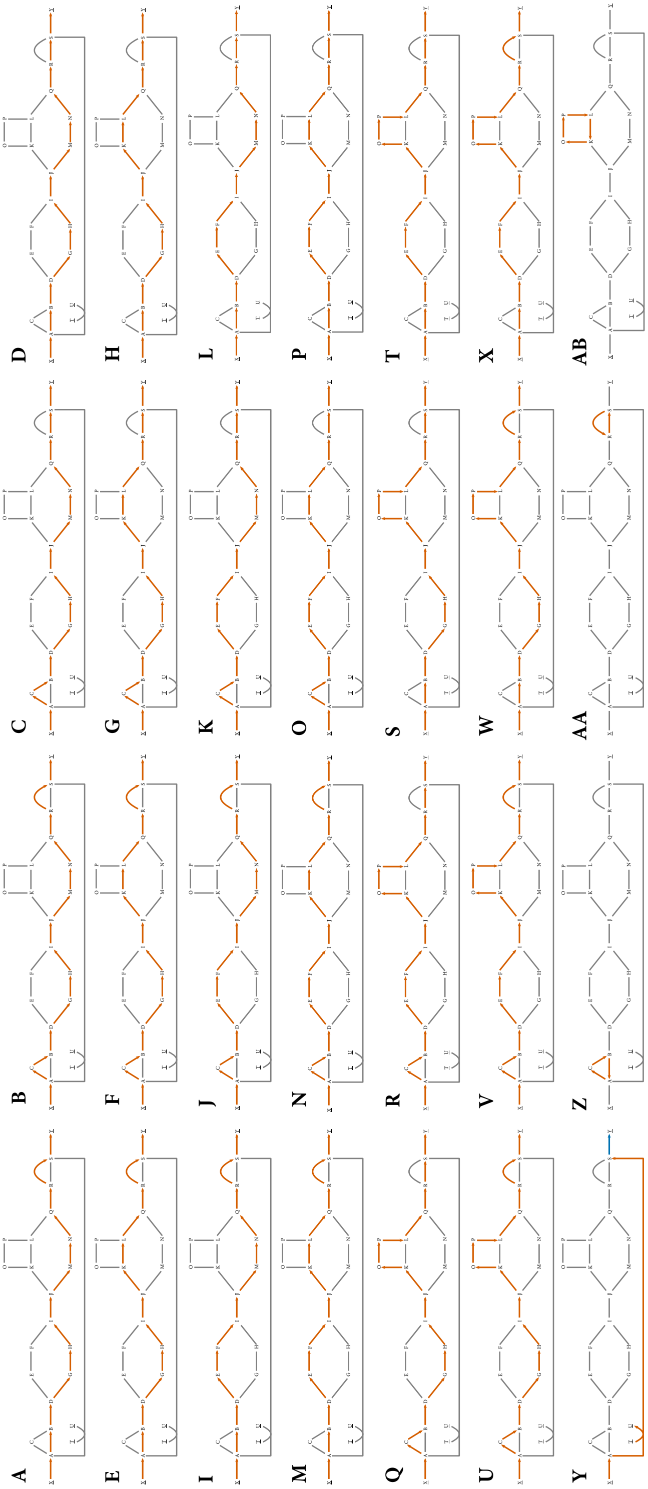


Figure 2.3. Topological characterization of all EFMs. *(A)–(X) EFM type I. (Y) EFM type II. (Z)–(AB) EFM type III. Visualizing ExPas requires decoupling of all reversible reactions into two irreversible reactions. Because all exchange reactions are irreversible, the set of relevant ExPas match this set of EFMs. Colors correspond to reaction values (red = 1, blue = 1/2).*

there is only one type II EFM that gives a sub-optimal yield (Fig. 2.3Y). Thus, any route from metabolite X to Y, involving reaction R25, results in a sub-optimal yield because R25 has stoichiometry A + T → U + S . Finally, there are three type III EFMs that characterize the internal loops of this metabolic network, as shown in Fig. 2.3Z–AB. Generally, these cycles are responsible for the large number of EFMs. To illustrate this point, without these three cycles the toy metabolic network has only five EFMs and there would be four EFMs to characterize all optimal pathways, one EFM to characterize the sub-optimal pathway, and zero EFMs to characterize internal loops.

2.8 Extreme Pathways

The alternative approach, ExPas, determines the edges of the cone that describe the steady-state solution space and the thermodynamic preference of reactions (Schilling et al., 2000). These ExPas are defined as:

Extreme pathways

A unique and minimal set of flux vectors that completely characterizes the steady-state capabilities of GSSMs (Papin et al., 2002).

The set of ExPas does, therefore, not have to contain all pathways with an optimal and sub-optimal yield, in contrast to the EFMs. Convex combinations of ExPas that satisfy the three EFM conditions can be used to obtain all optimal and sub-optimal pathways. In addition to the three conditions of EFMs, ExPas require two additional conditions: (iv) network reconfiguration and (v) systematic independence.

Network reconfiguration results in a classification of each reaction as an internal or exchange reaction. Moreover, each internal reversible reaction is split into two irreversible reactions: a reaction describing the forward reaction and a reaction describing the backward reaction. Systematic independence guaranties that an ExPa cannot be represented by a non-negative linear combination of other ExPas. Because of the systemic independence condition, ExPas are always a subset of the EFMs; each ExPa is also an EFM, but not necessarily vice versa. This can result in fewer ExPas than EFMs for the same metabolic model. For a metabolic model of the human red blood cell, the average number of EFMs used for a given ExPa was about four (Papin et al., 2004). However, if all exchange reactions in a metabolic model are irreversible, the sets of relevant EFMs and ExPas are identical. This is a general property of EFMs and ExPas (Papin et al., 2004; Klamt and Stelling, 2003). Each originally reversible internal reaction, split into two irreversible reactions, fulfills all ExPa and EFM conditions, resulting in additional ExPas and EFMs. These ExPas and EFMs can be considered irrelevant (Papin et al., 2004; Klamt and Stelling, 2003) because they only re-define reversibility. The exchange reactions (R1 and R26) in this toy metabolic model are irreversible, so the sets of relevant EFMs and ExPas are identical.

2.9 Characterizing the optimal flux space

FBA can be exploited to calculate the maximum yield of a product on a certain substrate. FBA simulation provides a steady-state flux distribution, which corresponds to a point in the (optimal) solution space. A unique optimal steady-state flux distribution through the metabolic network cannot be guaranteed because the constraints defined by the stoichiometric network are insufficient. Accordingly, a solution space of optimal steady-state flux distributions exists that each give rise to the maximal yield. This solution space represents a polyhedron (Grötschel et al., 1988) and this space is considerably smaller than the entire steady-state solution space characterized by flux modes, EFMs, or ExPas. This reduction in solution space is achieved in FBA by the consideration of additional constraints, a particular nutrient environment, and the demand for flux distributions that optimize a metabolic objective. Characterizing the optimal solution space of FBA remains a NP-hard computational problem. Above, we characterized the variability of the flux values within the optimal solution space. Next, we characterize this solution space in network topological terms.

In contrast to both the EFMs and ExPas approaches, CoPE-FBA (Comprehensive Polyhedra Enumeration Flux Balance Analysis) characterizes only the optimal solution space, which is done in terms of a compact set of subnetworks (Kelk et al., 2012). These subnetworks account for all alternative flux distributions in the optimal steady state predicted by FBA. CoPE-FBA therefore provides the topological structure underneath flux variability, at least in the optimal solution. The solution space of optimal flux distributions contains three topological features: vertices, rays, and linealities.

Vertices are optimal paths of the metabolic network, including reactions with fixed and variable fluxes. Rays are irreversible, thermodynamically infeasible cycles and linealities are reversible cycles in the metabolic network (both can also be input-output pathways). No net conversion occurs in either the rays or linealities of the FBA polyhedron. The optimal solution space of the toy metabolic network can be described by four vertices (Fig. 2.4B–E), one ray, and two linealities (Fig. 2.4A). The EFMs and ExPas also consist of these three topological features. Therefore, rays and linealities are responsible for the increase in the number of type I and II EFMs and ExPas. Alternatively, rays and linealities do not influence the number of CoPE-FBA subnetworks.

Any optimal flux distribution that satisfies the metabolic optimum obtained in the FBA calculation can be written in terms of the vertices, rays, and linealities using the Minkowski sum,

$$\boldsymbol{J^{opt}} = \sum_{k=1}^{s} \alpha_k \boldsymbol{\varphi_k} + \sum_{k=1}^{t} \beta_k \boldsymbol{\phi_k} + \sum_{k=1}^{u} \gamma_k \boldsymbol{\psi_k}, \tag{2.21}$$

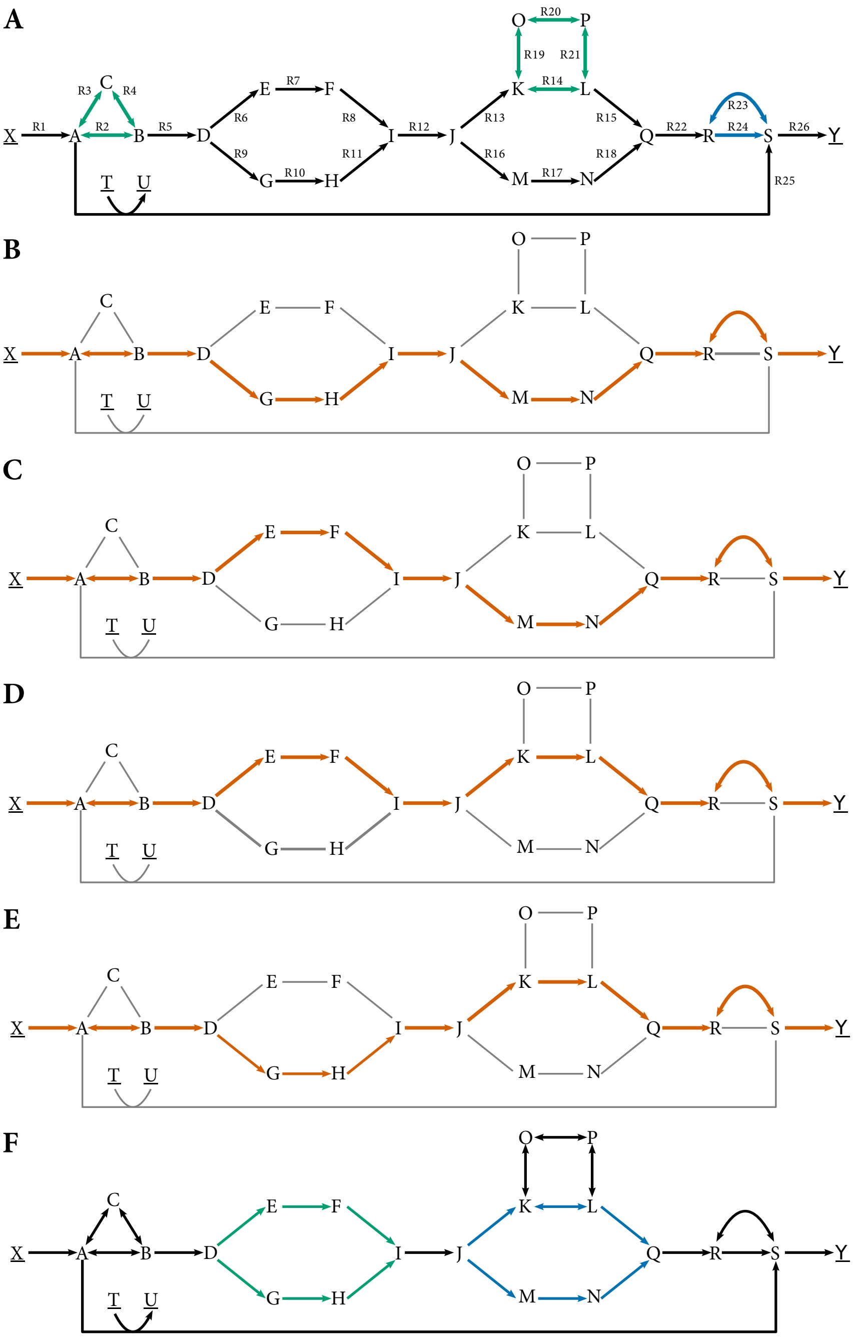


Figure 2.4. Topological characterization of the optimal FBA solution space. *The optimal solution space of our toy model contains one ray (panel A, blue, R23 and R24), two linealities (panel A, green, R2–R4 and R14, R19–R21), and four vertices (panels B-E) which originate from two CoPE-FBA subnetworks (panel F). In Chapter 4, we show that this is not a unique characterization of the optimal solution space.*

in which the vectors $\boldsymbol{\varphi_k}$, $\boldsymbol{\phi_k}$, and $\boldsymbol{\psi_k}$ represent the vertices, rays, and linealities, respectively (Grötschel et al., 1988; Kelk et al., 2012). The weighting coefficients obey the following restrictions: $\sum_{k=1}^{s} \alpha_k = 1$, $\alpha_k \geq 0$, $\beta_k \geq 0$, and γ_k can take any value. These definitions indicate that the vertices can be summed in a convex manner, the rays as a conical sum, and a linear combination can be taken over the linealities.

The subnetworks that can be identified with CoPE-FBA (and explain the numbers of vertices for a given FBA problem) satisfy three conditions: (i) only reactions belonging to a specific subnetwork display correlation in flux values across the optimal solution space, (ii) fixed net input-output stoichiometry of reactants, and (iii) thermodynamic feasibility. This means that these subnetworks contain reactions that vary independently across all vertices of the optimal solution space. Therefore, without violating the optimality condition, subnetworks with alternative internal flux distributions can be independently chosen. For this reason, the number of vertices can be determined by multiplying the number of alternative internal flux distributions for each subnetwork (Kelk et al., 2012). This illustrates the likely combinatorial explosion for the number of vertices of the optimal solution space for larger metabolic networks.

The toy metabolic network contains two CoPE-FBA subnetworks given in Fig. 2.4F. Each subnetwork has two alternative internal fluxes distributions: the top and bottom branch. Multiplying the number of alternative flux distributions for each subnetwork, $2 \times 2 = 4$, gives the number of vertices. In Chapter 4, we show that this analysis is not unique (i.e. different sets of vertices are possible) but becomes unique (i.e. all non-decomposable flux vectors are enumerated as vertices) when we perform reversible-reaction splitting. Larger metabolic models tend to contain more vertices, while the number of subnetworks stays small. For instance, the GSSM iTM686, consisting of 904 reactions and 816 species (see Chapter 3.4), has about 368640 vertices when studied under light limited conditions (Chapter 5). Nine subnetworks which contain about 8.1% of the total number of reactions in this model, are enough to characterize the optimal solution space. Comparing the number of EFMs, ExPas, vertices, and CoPE-FBA subnetworks gives an indication of the level of compactness of these approaches. Typically, for larger models the number of EFMs, ExPas, and vertices explodes, which gives # EFMs ≥ # ExPas > # vertices > # subnetworks.

2.10 Biological implications of stoichiometric network analysis

The predictions made by any mathematical model depend heavily on the underlying assumptions. The definitions of ExPas, EMFs, and those related to FBA have the steady-state assumption in common. In general, the steady-state assumption is assumed to be valid because of the timescale separation between (fast) intracellular metabolic conversions and (slow) genetic regulation (Segrè et al., 2002; Lee et al., 2006).

In addition to the steady-state assumption, FBA assumes optimization of an objective function, which is, in some cases, debatable from a biological perspective. Typical objective functions are the yield of the biomass reaction or ATP production. Optimization of these objectives is always bounded by capacity constraints of other reactions that ultimately bound the steady-state solution space. In other words, FBA optimizes an objective function relative to a limiting input flux. Thus, optimization of any reaction rate in FBA is always the optimization of a yield defined as the objective reaction rate divided by the limiting input. Optimization of growth rate rather than growth yield is a completely different strategy; this can be easily understood because the yield does not fix the rates of the metabolic processes. Selection for yield only occurs in the absence of competition for nutrients, which is an unlikely scenario in biology. The assumption of one objective may actually not always reflect reality: The occurrence of trade-offs between two metabolic objectives may cause cells to optimize both of them simultaneously (possibly, with different weights), leading to Pareto optimization problems (Schuetz et al., 2012).

2.11 Concluding remarks

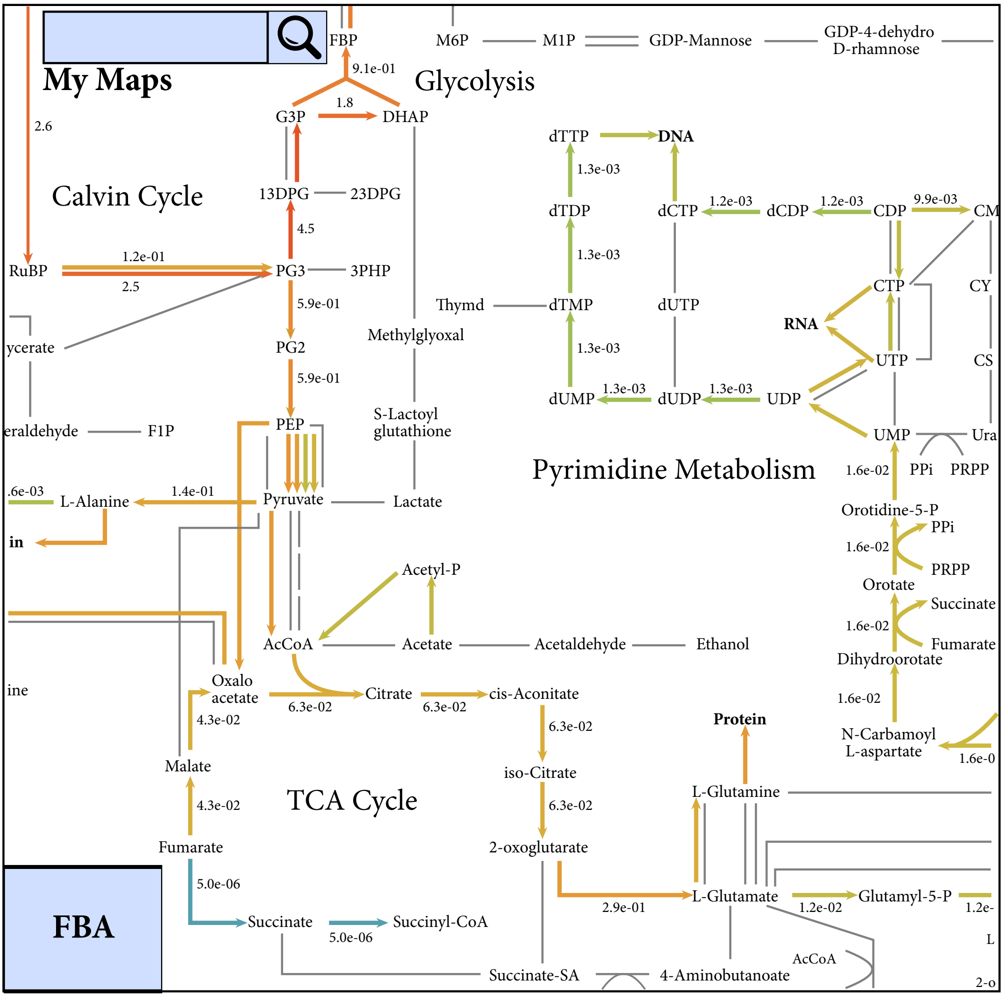
We provided an overview of the most common mathematical techniques used in the stoichiometric analysis of metabolic networks. We have not described in any detail the application of these techniques to biological problems, which is found elsewhere (Feist and Palsson, 2008; Raman and Chandra, 2009; Oberhardt et al., 2009; O'Brien et al., 2015). These applications to biology are the reasons for the existence of pathway analysis, and there are a number of success stories (Ibarra et al., 2002; Lewis et al., 2010). Yet, the simplifications and subsequent limitations of the described techniques are also clear and extensions to pathway analysis methods include the incorporation of dynamics (such as in dynamic FBA (Mahadevan et al., 2002)), additional constraints (such as space or resource limitations (Beg et al., 2007; van Hoek and Merks, 2012; Branco dos Santos et al., 2013), multidimensional optimality (Schuetz et al., 2012), and extensions to multi-species FBA (Stolyar et al., 2007; Zomorrodi and Maranas, 2012). We therefore expect that such analysis will penetrate biology in increasingly many ways to provide rigorous and quantitative hypotheses and fundamental understanding.

Supplemental Material

Dataset S1. A description of our stoichiometric toy model in the SBML level 3 format is available at http://persistent-identifier.org/?identifier=urn:nbn:nl:ui:18-23537 and in the online version of this article on which this chapter is based on.

This chapter is based on the publication:

Maarleveld T.R., Boele J., Bruggeman F.J., and Teusink B. (2014) *"A Data Integration and Visualization Resource for the Metabolic Network of Synechocystis sp. PCC 6803"*, **Plant Physiology**, 164, 3:1111-21.



"Google Maps" for metabolic highways: An interactive map of *Synechocystis* metabolism and a corresponding genome-scale stoichiometric model. This interactive map allows for visualization of both computational and experimental data.

Developing Google Maps for metabolic highways

3

Abstract

Data integration is a central activity in systems biology. The integration of genomic, transcript, protein, metabolite, flux, and computational data yields unprecedented information about the system level functioning of organisms. Often, data integration is done purely computationally, leaving the user with little insight besides statistical information. In this chapter, we present a visualization tool for the metabolic network of *Synechocystis* PCC6803, an important model cyanobacterium for sustainable biofuel production. We illustrate how this metabolic map can be used to integrate experimental and computational data for *Synechocystis* systems biology and metabolic engineering studies. Additionally, we discuss how this map, and the software infrastructure that we supply with it, can be used in the development of other organism-specific metabolic network visualizations. Besides a Python console package VoNDA (http://vonda.sf.net), we provide a working demonstration of the interactive metabolic map and the associated *Synechocystis* genome-scale stoichiometric model, as well as various ready-to-visualize microarray data sets, at http://f-a-m-e.org/synechocystis/.

3.1 Introduction

The recent advances in high-throughput experimental technologies and whole genome-sequencing resulted in an explosion of biological data availability. These genome-scale data sets are generally described as "omics" data sets of which transcriptomics, metabolics, and fluxomics data are examples (Gstaiger and Aebersold, 2009; Wang et al., 2009; Kahn, 2011; Grabherr et al., 2011). Analyses and integration of these data types can reveal important information about the system level functioning of organisms (Berger et al., 2013) and these methods are having a large impact on biology. To gain additional insight into cellular behavior, data sets are also used to develop computational models at a genome-scale level. Both the central role of metabolism for maintenance of cellular integrity and growth, and the availability of genome-scale data sets led to an increased interest in genome-scale reconstructions of metabolism. These reconstructions are based on the metabolic reactions that they can perform, given the content of their genome (Oberhardt et al., 2009; Thiele and Palsson, 2010). Essentially all metabolic reactions are catalyzed by enzymes, but for most reactions, the enzyme kinetics are unknown. Consequently, genome-scale reconstructions are often solely based on the reaction stoichiometry, and they are therefore generally referred to as genome-scale stoichiometric models (GSSMs) of metabolism.

To leverage the biological information that is included in GSSMs and to compute physiological properties, various stoichiometric network analysis tools are available (Price et al., 2004). Many of these tools are discussed in detail in the previous chapter, and they can be used to computationally study the growth characteristics of micro-organisms in detail. For instance, Flux Balance Analysis (FBA) and Flux Variability Analysis (FVA) have been used to predict the internal flux distribution and its variability while optimizing for cellular growth yield (Mahadevan and Schilling, 2003; Orth et al., 2010; McCloskey et al., 2013). Stoichiometric computational methods can also directly be of practical value. For example, OptKnock (Burgard et al., 2003) can be used to predict which gene knockout strategies would result in increased production of metabolites of interest, such as biofuels.

The analysis of GSSMs results in big data sets that often contain thousands of reactions. These results come in the form of long lists of (reaction) identifiers and (flux) values. This makes the interpretation of these data—which often come in files comprising thousands of lines—a rather tedious process. Yet to be of any biological value, interpretation of, for example, FBA results in terms of biological functions, modules, and pathways is required. To achieve this, adequate visualization is paramount, and attempts to visualize genome-scale data have indeed been made. These approaches can be classified into supervised (human-driven) and non-supervised (fully automatic) ones. The latter approach, fully automated mapping of GSSMs, results in (mathematical) graphs that are even more difficult to interpret than the raw data one set out to visualize on them, so we do not discuss them in detail here.

Human-driven visualization methods, on the other hand, are considerably more popular, as hand-drawn maps are markedly easier to interpret. Tools like KEGG (Kanehisa and Goto, 2000), Cytoscape (Smoot et al., 2011), CellDesigner (Funahashi et al., 2003), and the COBRA Toolbox (Schellenberger et al., 2011) each offer their own particular take on modeling, simulation, and visualization. Nevertheless, since none of them was originally designed for visualizing genome-scale models or genome-scale data, they are of limited use for this purpose. This becomes clear when one attempts to share, extend, or adapt an existing map—commonly, users who attempt this quickly find their progress obstructed by the need for special software, incompatible identifiers, or simply by not being able to access an editable version of the desired image.

Understandably, these issues have thus far prevented the wide-spread communal use of visualization aids, as well as their use in combination with computational results. However, the increased availability of high-throughput data makes the lack of adequate graphical interpretation tools more dearly felt than ever. Here, we address this problem by presenting the first comprehensive data visualization tool, namely, a graphical map of metabolism that is capable of displaying any type of data that is associated with reaction identifiers, gene identifiers, or metabolite identifiers. Based on the increased interest in photosynthetic micro-organisms and its importance as a model-organism for metabolic engineering studies, we decided to base our graphical metabolic map on the metabolic reconstruction of *Synechocystis* PCC 6803 (hereafter *Synechocystis*) (Shastri and Morgan, 2005; Knoop et al., 2010; Nogales et al., 2012; Saha et al., 2012), the best characterized and most extensively studied cyanobacterium. The direct conversion of carbon dioxide (CO_2) into carbon compounds makes these micro-organisms of great interest for the development of production techniques for third-generation biofuels (Chisti, 2007; Ducat et al., 2011; Angermayr et al., 2012). We illustrate the use of the map of this organism's metabolism and discuss how similar maps can be made for other organisms using the *Synechocystis* map as a template.

Our map is also directly useful for other (micro-)organisms, since the metabolic core (e.g. the carbohydrate, energy, amino acid, and nucleotide metabolism) is highly conserved between different micro-organisms (Peregrin-Alvarez et al., 2009). Therefore, the map can easily be extended to cover additional pathways or organisms. This process becomes progressively easier as more reactions are added to the map, as any added reactions can in turn be used in further mapping efforts. To demonstrate the usefulness of our graphical map, we integrated our graphical map into FAME, the Flux Analysis and Modeling Environment (Boele et al., 2012) at http://f-a-m-e.org/synechocystis/. Here, our map can be readily used for the analysis and visualization of the *Synechocystis* GSSM, which also comes pre-loaded. The map and a Python console application of our visualization software VoNDA, Visualization of Network DAta, can also be downloaded separately from http://vonda.sf.net/. The console application can be used to visualize output generated by other software packages such as the popular COBRA Toolbox. We prefer using VoNDA in combination with CBMPy (Olivier, 2011) (or COBRApy (Ebrahim et al., 2013)) because this allows for (interactive) modeling, simulation, and visualization in Python.

3.2 Results and Discussion

Synechocystis' metabolic map

We developed a comprehensive metabolic map of *Synechocystis* (Fig. 3.1A). Our comprehensive map is based on the metabolic reconstruction of Nogales et al. (2012), which we extended with reactions from Knoop et al. (2010) and other sources (documented in Supplemental Data S3.1). The extended model contains 904 reactions and 816 species. The associated map provides an overview of the reactions in the model and the resulting metabolic pathways. Together, the model and the map act as a readily extensible framework for human-friendly visualization of simulation results or experimental data (Fig. 3.1). Throughout this project, we aimed at offering an environment that can easily be extended with additional data and analysis types. To ensure its applicability to a maximum number of data and analysis types, as well as to maximize its potential for re-use and adaptation by the cyanobacterium community or other communities, a number of considerations were made during the development of the graphical map of *Synechocystis*' metabolism.

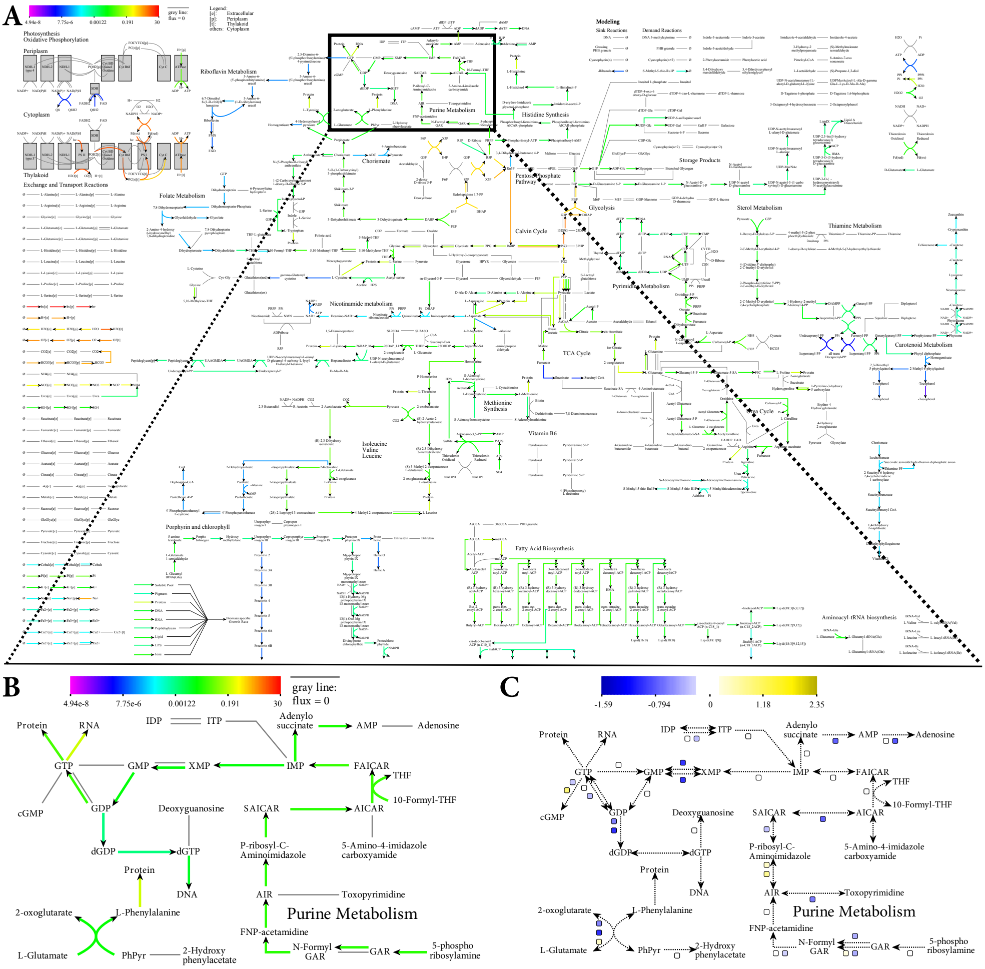
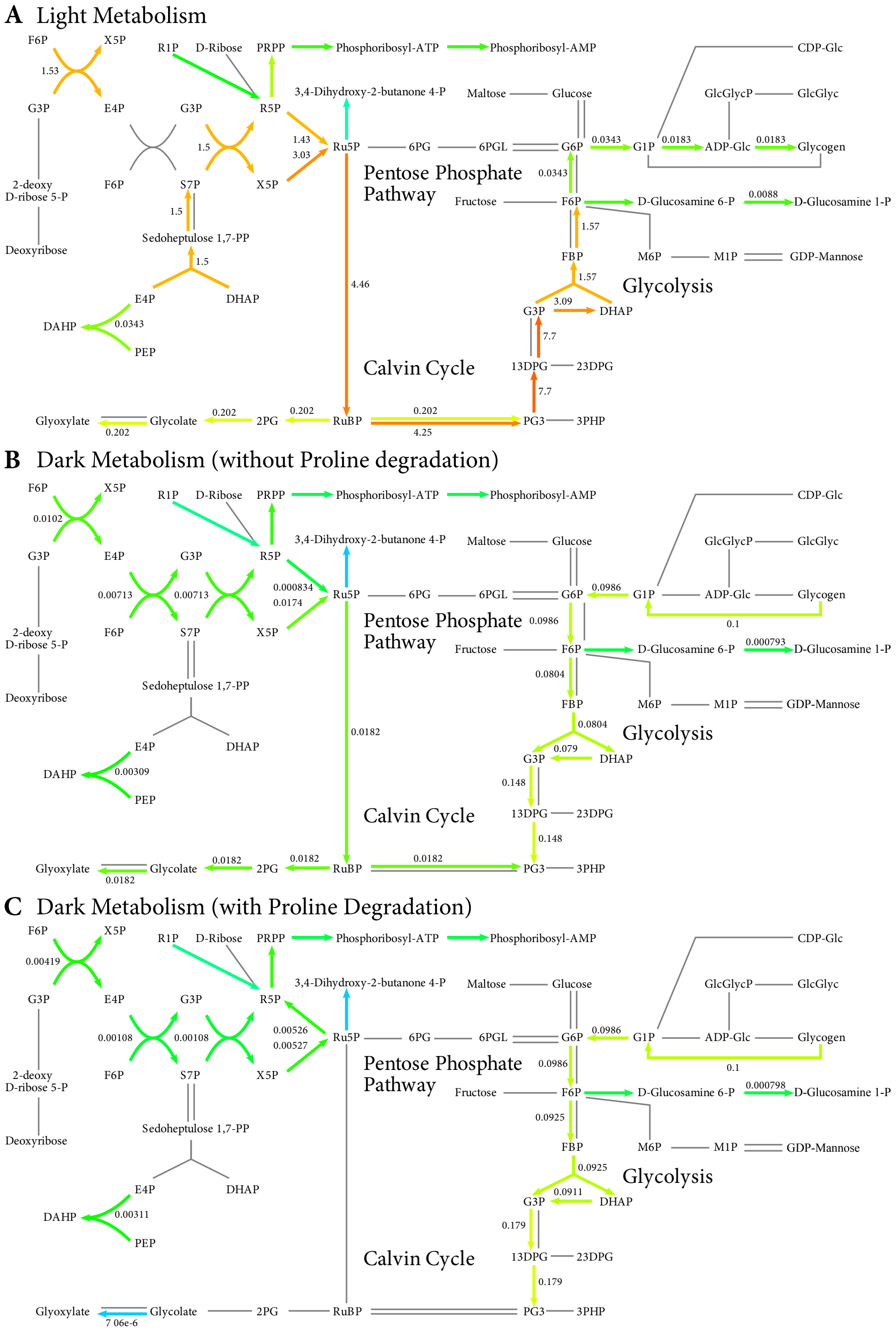


Figure 3.1. An overview of the graphical map of *Synechocystis*' metabolism and its capabilities. *Various elements (reactions, metabolites, squares) are clickable and direct the user to the KEGG database, or to run result details. (A) when displaying FBA results, the arrows denoting reactions are colored according to the color scale in the upper left corner. Gray reactions carry no flux. (B) detailed view of a section of the map when it is used to display FBA results. (C) detailed view of the same section of the map as in panel B, but now it is being used to display gene expression results. Here, the reaction lines are only displayed to indicate the general structure of the metabolic map, whereas the gene expression information is conveyed by colored boxes next to each reaction for which gene association information is available.*

An essential feature is that simulation results, such as Flux Balance Analysis (FBA) or Flux Variability Analysis (FVA), can be displayed within the tool that generated them (Fig. 3.1A-B). Moreover, experimental data, such as various flavors of omics' data sets, can be displayed (Fig. 3.1C). This was achieved by ensuring that the identifiers of the reactions, metabolites, and genes in the underlying model match those on the map. Such an annotation link is essential, and as long as experimental data sets use the same identifier classification, they can be visualized. In our map, the underlying identifiers can be easily detected by a mouseover of the element of interest in a web-browser.

We used the Scalable Vector Graphics (SVG) format to represent the map, as this allowed us to make our map interactive. Specifically, we included hyperlinks that can direct the user towards sources of additional information (KEGG) or, when FAME is used to generate simulation results and display them on the map, to details about the run that generated the displayed results. Had we utilized one of the budding visualization standards like the Systems Biology Graphical Notation (SBGN) (Le Novère et al., 2009) or SBML layout (Gauges et al., 2006), this functionality would require the user to install additional third-party software simply to view the image. Using SVG directly allows the user full graphical control over the image and allows direct embedding of dynamic, interactive elements that point to metabolite and reaction resources. The SVG format is commonly used on the web and popular image manipulation software (such as InkScape or Adobe Illustrator) can import SVG files. Therefore, these files can be used for figures in scientific publications. Using SVG also makes sure our map is readily extensible with new reactions or visualization applications. Moreover, the format is programmatically accessible, which allows for extraction and visualization of subnetworks of the entire metabolic map, or development of a new metabolic map for another organism, using this map as a blueprint.

To demonstrate that our tool greatly simplifies the analysis of genome-scale data sets, we provide two example use cases. One example concerns the physiological adaptation of cyanobacteria to changing environmental conditions. The other illustrates the expansion of the map with new reactions for biofuel production. Given GSSM constraints such as light availability, FBA is generally an accurate method to predict the optimal product or biomass yield. However, since these GSSMs are typically large, the associated metabolic flux distributions are large, which makes them difficult to analyze in detail. To exemplify this, FVA done on our GSSM predicts that, in an optimized GSSM, about 600-700 reactions can carry a flux during light and dark conditions. Our metabolic map allows for visualization of these FBA and FVA predictions, which allows for direct insight into, for instance, unexpected and interesting behavior.



A Light Metabolism

B Dark Metabolism (without Proline degradation)

C Dark Metabolism (with Proline Degradation)

Figure 3.2 ***(previous page).*** **Example use cases of the map.** *Visualization of the difference in flux distribution when growing in the presence (panel A) or absence (panels B and C) of light. When light is present (panel A), the FBA solution harbors no surprises. However, when light is absent (panel), FBA predicts that photorespiration is active, which is unlikely. Adding a reaction that converts L-Pro to Hyp returns the model to its expected state under dark conditions (panel C). Some glyoxylate is still produced from other precursors than glycolate (not shown here), to enable the production of amino acids.*

Visualizing the physiological adaptation to light and dark conditions

In this section we provide an example of unexpected behavior that was found by simulating cyanobacteria in different environmental conditions. Cyanobacteria such as *Synechocystis* are subject to a diurnal light-dark cycle and adjust their metabolism via circadian control. The GSSM can be used to simulate these conditions by changing the availability of light (photon influx) and adjusting the model constraints to indicate whether glycogen is to be produced as part of biomass (see Supplemental Data S3.2 for details).

In Fig. 3.2, we illustrate FBA predictions of the metabolic activity under light and dark conditions. We limit ourselves to the visualization of the metabolic surroundings of ribulose-1,5-bisphosphate carboxylase oxygenase (RuBisCO). RuBisCO is a well-known key enzyme for carbon fixation during photo-autotrophic growth, which carboxylates or oxygenates D-ribulose 1,5-bisphosphate. The carboxylation process (carbon fixation) yields two molecules of 3-phosphoglycerate (3PG), while the oxygenation process (photorespiration) results in one molecule each of 3PG and 2-phosphoglycolate (2PG). 2PG is toxic to *Synechocystis* and inhibits proper functioning of the Calvin-Benson cycle (Bauwe et al., 2012) and, as a result, reduces the conversion of solar energy into chemical energy. Nevertheless, both carbon fixation and photorespiration are essential for cyanobacteria in photo-autotrophic growth conditions (Eisenhut et al., 2008).

We predicted steady-state flux values through this segment of metabolism by using FBA to optimize the biomass synthesis rate. This rate optimization is carried out relative to other fixed nutrient uptake reaction rates and that we are effectively optimizing flux ratios (yields) rather than rates. Now, instead of working one's way through a list of about 1000 predicted flux values for unexpected behavior, our map directly shows that both FBA and FVA predict that, besides carbon fixation, photorespiration is essential for obtaining an optimal growth yield in light and dark conditions (Fig. 3.2A–B). The reason is that, in the metabolic model, photorespiration is the only way to produce glyoxylate, which is necessary for the production of the amino acids cysteine, glycine, and serine.

As "dark photorespiration" is not generally considered plausible, an alternative was proposed by Knoop and coworkers (2010). They added the conversion of L-proline to hydroxyproline (identified by a BLAST search (Altschul et al., 1997)) to provide a second, perhaps more plausible mechanism for glyoxylate synthesis in dark conditions. After inclusion of this reaction, our computational analysis showed that in light conditions no activity was predicted for the conversion of L-proline to hydroxyproline due to a lower carbon efficiency. According to FBA simulations, glyoxylate synthesis via L-proline breakdown would be more efficient in dark conditions (Fig. 3.2C).

Besides directing fixated carbon towards the production of precursors for (immediate) growth (e.g. nucleic acids or amino acids), *Synechocystis* also synthesizes the storage compound glycogen (Fig. 3.2A). During photo-autotrophic growth, this results in a sub-optimal growth strategy as a lower growth rate is predicted. In the absence of light, *Synechocystis* consumes the stored glycogen via either fermentation or respiration, depending on the availability of O_2. In the absence of light and presence of O_2, FBA predicts the breakdown of previously stored glycogen and energy generation via respiration. This enables the organism to maintain its integrity and grow under dark conditions. If the *Synechocystis*' maintenance requirements in the absence of light were known, modeling and simulation could help determine the minimal glycogen amount that should be stored during the light phase. This could represent an optimal strategy for this organism—optimize glycogen storage during the day to grow as fast as it can and then survive the night on the stored glycogen.

Exploring the properties of *Synechocystis* photosynthesis

Besides photorespiration, photosynthesis is another key part of *Synechocystis*' metabolism that is essential for growth. We studied the flexibility of this metabolic subnetwork by maximizing the specific growth rate as a function of the bicarbonate (HCO_3^-) and photon uptake rates (Fig. 3.3A). The specific growth rates we predict are in agreement with experimentally measured rates (Shastri and Morgan, 2005). We subsequently selected three combinations of bicarbonate and photon uptake rates for further exploration (Fig. 3.3A). Briefly, these are: (i) a state in which carbon and light are both limiting for growth (carbon and light limiting state, CLLS), (ii) a state in which light, but not carbon limits growth (light limiting state, LLS), and (iii) a state in which carbon, but not light limits growth (carbon limiting state, CLS). We give a detailed overview of the characteristics of these conditions in Supplemental Data S3.2, and of the FBA predictions under each condition in Table 3.1. In this section, we discuss some analysis results that illustrate the practical application of our tool.

Visualization of FBA predictions on our metabolic map immediately shows that the model predicts different photosynthesis strategies under different conditions, indicating the flexibility of this system. The predictions in the CLLS (Fig. 3.3B) and LLS (Fig. 3.3C) are conceptually identical, although the predicted absolute fluxes are different. However, the two conditions can be distinguished on the basis of carbon secretion: The CO_2 that cannot be fixed in the LLS is released by the cell.

By comparing the visualization of photosynthetic fluxes, it is readily apparent that a different strategy is exploited in the CLS (Fig. 3.3D). Excess photons are used to transfer

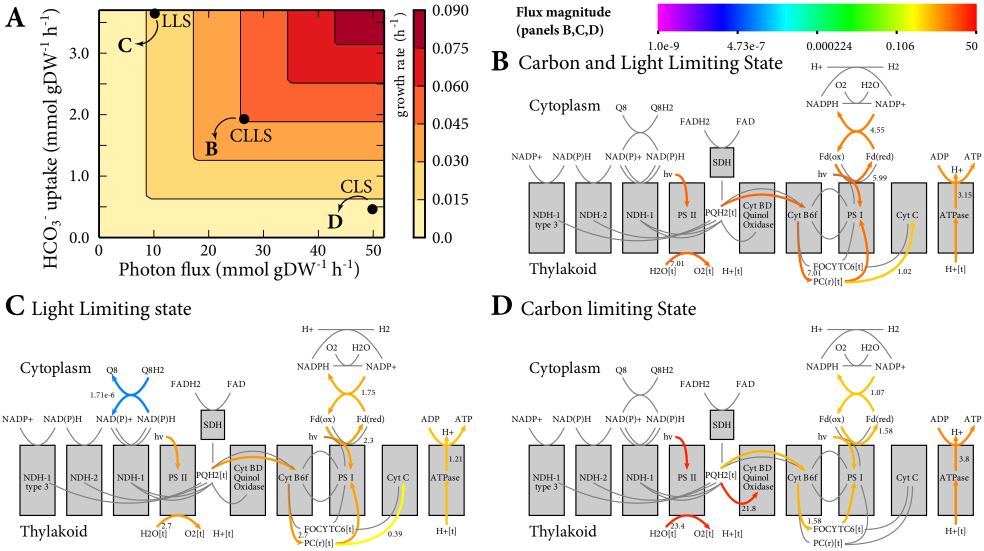


Figure 3.3. We used FBA to explore the properties of the *Synechocystis* model and map under various conditions (i.e. CLLS, LLS, and CLS). *(A) predicted growth rate as a function of carbon and light uptake rates. The black dots in the phase plane indicate the conditions (in terms of photon flux and* HCO_3^- *uptake) that are shown in panels B to D. (B-D) visualization of predicted FBA fluxes for the three scenarios (CLLS, LLS, and CLS). Arrows indicate flux direction; the color bar indicates flux magnitude.*

protons from the cytosol to the thylakoid, a process which is carried out by quinol oxidase (Cytochrome BD). These protons are then used to generate ATP. This explains the relatively low flux through photosystem I compared to photosystem II (J_{PSI}/J_{PSII}) and relatively high ATP formation flux compared to the NADPH formation flux (J_{ATP}/J_{NADPH}); more details can be found in Table 3.1. The resulting ATP is subsequently used in metabolism. FVA results (not shown) suggest that quinol oxidase is not essential for obtaining a maximal biomass yield. For example, the Mehler reaction can be used to oxidize the excess NADPH to NADP+ (Fig. 3.4A); in that case, $J_{PSI}/J_{PSII} = 1$ and $J_{ATP}/J_{NADPH} = 1.28$ indicating linear electron flow.

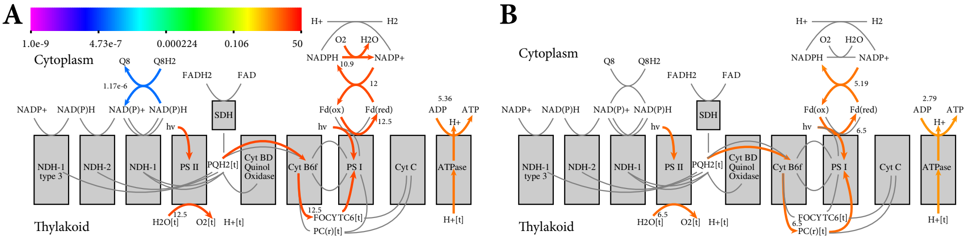


Figure 3.4. Results of FBA knock-out simulations. *(A) optimal strategy in the CLS for a quinol oxidase knock-out. The Mehler reaction is used to convert the excess NADPH into NADP$^+$. (B) optimal strategy in the CLLS after shutting down the flux through all AEF pathways.*

	J_{PSI}/J_{PSII}	J_{ATP}/J_{NADPH}	$J_{RuBisCo,CO2}/J_{h\nu}$	μ (h^{-1})	J_{23BD}
CLLS	0.99	1.6	0.085	0.045	0
CLS	0.058	7.2	0.011	0.012	0
LLS	0.99	1.6	0.085	0.017	0
KO AEF	1.0	1.28	0.075	0.040	0
Biofuel	0.99	1.3	0.13	0.0045	0.427
Biofuel CLS	0.058	8.1	0.014	0.0012	0.113
Biofuel LLS	0.99	1.3	0.13	0.0017	0.203
Biofuel KO AEF	1.0	1.28	0.11	0.0045	0.37

Table 3.1. A summary of the results of various (flux balance) analyses on the *Synechocystis* model. *When optimizing for biofuel production, we fixed the growth rate to 10% of its maximum. The constraints that were used for each of these simulations can be found in Supplemental Data S3.2.* $J_{h\nu}$*, Photon flux; KO AEF, Knock-out of AEF pathways;* $J_{RuBisCo,CO2}/J_{h\nu}$*, RuBisCo carboxylase flux divided by the photon flux;* J_{23BD} *2,3-butanediol flux;* μ (h^{-1})*, growth rate.*

In addition to the photosynthesis linear electron flow, different alternative electron flow (AEF) pathways exist that allow for different ATP/NADPH ratios (Kramer and Evans, 2011). From Figs. 3.3B-D we can immediately conclude that these AEF pathways are active during (these instances of) conditions of optimal growth yield. Indeed, when the AEF pathways are removed from the model, the predicted growth yield is lower, indicating that AEF pathways are essential if maximum yield is to be attained under these conditions (Fig. 3.4B).

Apart from plotting model predictions on the map, we can use the tool for visualization of many more data types. In Fig. 3.5 we use microarray data under conditions of carbon limitation (similar to the conditions modeled above). Visualizing this type of data allows for convenient identification of genes that are over- or underexpressed compared to wild-type, in a pathway-specific manner. For instance, compared to wild-type, our visualization shows that many genes associated to photosynthesis and respiration are initially underexpressed (Fig. 3.5A) upon CO_2 limitation. The transcriptomics data are from a time-series, whereas predictions from FBA assume steady state; therefore, these data sets are not directly comparable.

After 12 hours of CO_2 limitation, the situation most likely represents a steady-state condition and it can therefore be compared with FBA predictions. When this comparison is made, 73% of the photosynthetic gene expression data agree with predicted fluxes, while 27% do not match the flux predictions. For instance, genes associated with linear electron flow were underexpressed (Fig. 3.5B). Indeed, FBA results visualized in Figs. 3.3B and 3.3D show that lower fluxes are predicted in most reactions associated to the linear electron flow. In contrast, a decrease in ATPase gene expression was found under CLS conditions, whereas FBA/FVA predicted a higher flux through this enzyme. The gene expression data

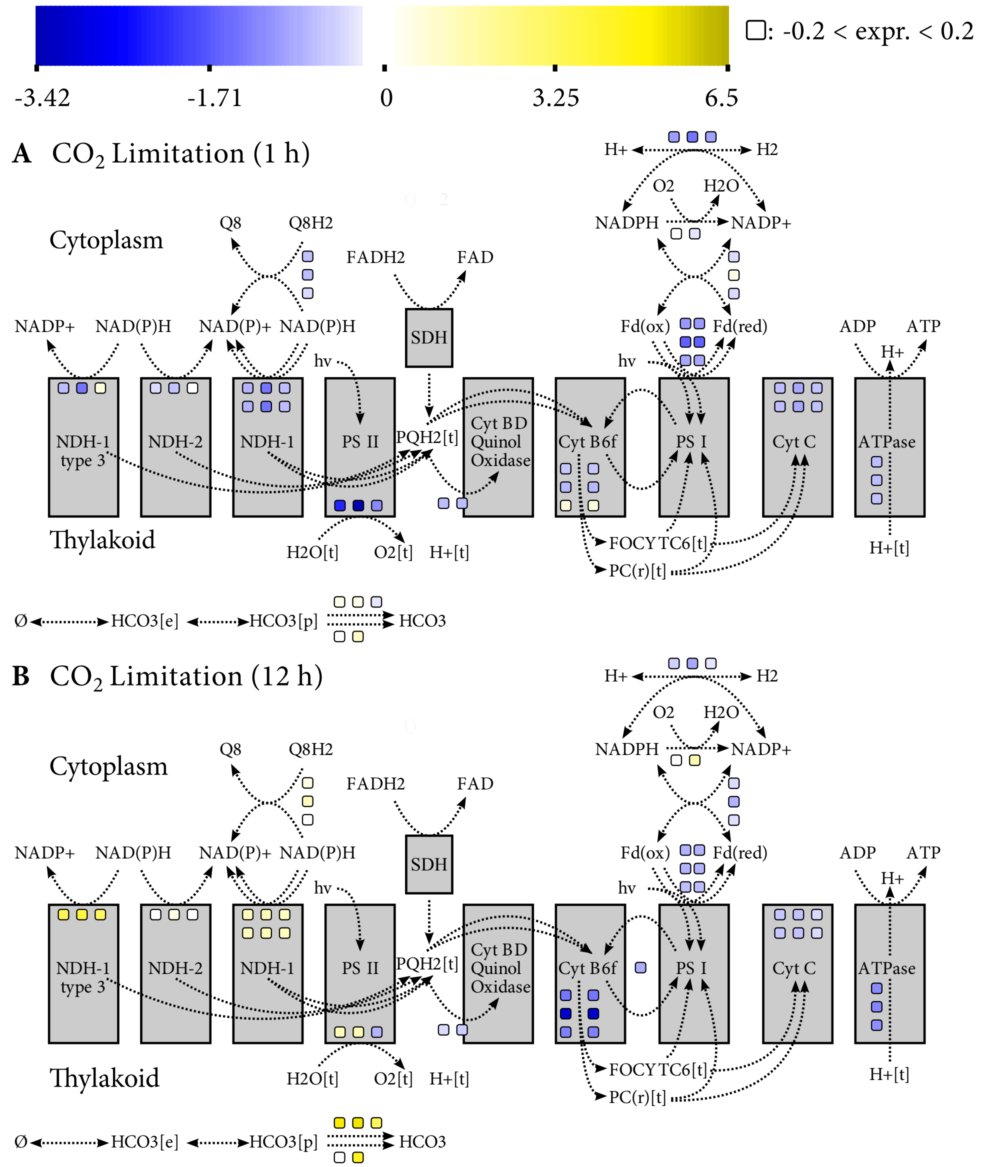


Figure 3.5. To demonstrate the applicability of our map to gene-associated ("omics"-size) data, we used it to visualize microarray data for various CO$_2$ limiting growth conditions. *The colored squares indicate differential expression relative to wild-type; yellow corresponds to overexpression and blue corresponds to underexpression. The panels depict differential gene expression after 1 h (panel A), 12 h (panel B) of CO$_2$ limitation, compared to wild-type Synechocystis.*

also indicate an increased expression of NDH-1 and NDH-1 type 3. While FBA does not predict a flux through these reactions in CLS conditions, FVA results indicate that a flux through NDH-1 related pathways would decrease the growth yield in LLS, but not in CLS conditions. Thus, the NDH-1 gene expression data, too, are consistent with the predicted solution space for *Synechocystis*' photosynthesis fluxes.

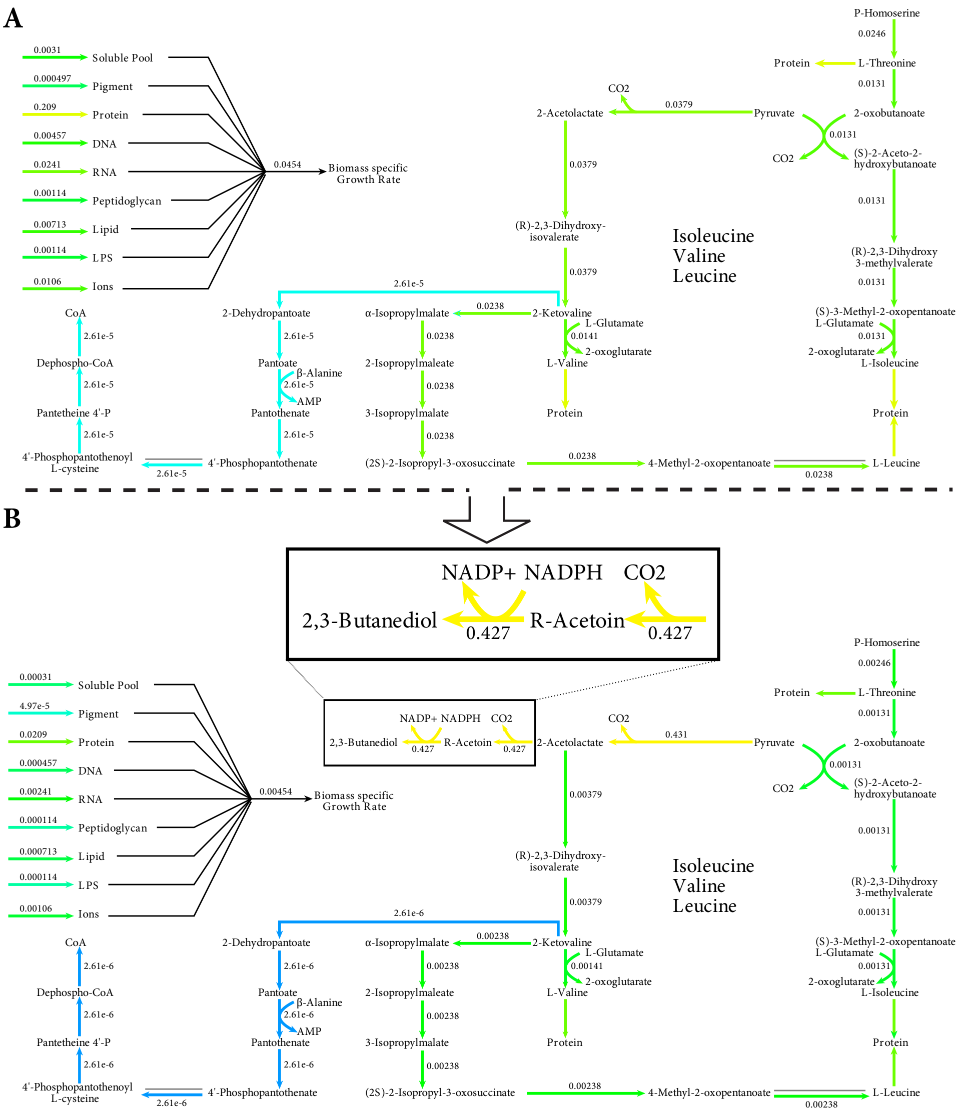


Figure 3.6. The addition of a biofuel pathway to the *Synechocystis* map. *(A) results of an FBA, optimizing for biomass production. This figure also illustrates that the total flux towards biomass is split into nine different components. (B) after the additional reactions were inserted in the map, we performed another FBA in which biomass production was fixed at 10% of the original optimum. Then, we optimized for 23BD production.*

Expansion of the map with a biofuel production pathway

To illustrate how the metabolic map can be extended or modified to other micro-organisms, we added a biofuel production pathway to the map and model. Recently, the development of a *Synechococcus elongatus* PCC 7942 strain that produces the bulk chemical 2,3-butanediol (23BD) was reported (Oliver et al., 2013). The production of 23BD using cyanobacteria offers several benefits compared to other higher alcohols. For instance, 23BD has a low host toxicity, which allows for a higher biofuel product concentration. Moreover, the common metabolite pyruvate is used as a substrate for 23BD production, and NADPH can be utilized as a reducing agent; no conversion of NADH to NADPH is necessary. Finally, catabolic enzymes from O_2-tolerant organisms can be used for the production of 23BD, which is important because O_2 is produced by *Synechocystis*' photosynthetic machinery. We therefore extended our map with a pathway that produces 23BD (see Materials and Methods and Fig. 3.6).

We performed FBA on the model after adding the 23BD synthesis pathway, under the same conditions as above (i.e. CLLS, LLS, and CLS) (Fig. 3.7). Ideally, cyanobacteria can be used as cell factories (Angermayr et al., 2012); a non-growing population that catalyzes the conversion of CO_2 into products of interest. To account for biomass replenishment and for the fact that a little growth always occurs, we fixed the growth rate to a basal growth level of 10% of its maximum specific growth rate. We then optimized for 23BD production (Fig. 3.7A). Given these conditions, our GSSM predicted 23BD production rates up to 0.90 mmol gDW^{-1} h^{-1}. Interestingly, the ratio of RuBisCo carboxylase activity over the photon uptake flux was about 1.5 times higher when we optimized the flux to 23BD compared to optimizing the specific growth rate (Table 3.1). Our metabolic map allowed for a straightforward identification of different factors (Fig. 3.8) that contributed to this difference. First, HCO_3^- is not entirely converted into CO_2 when biomass is synthesized. Instead, a substantial part (7%) of HCO_3^- is used for the synthesis of malonyl-CoA, an essential precursor for fatty acid biosynthesis. Second, the two reactions converting pyruvate to R-acetoin (the precursor of 23BD) release CO_2, which is subsequently fixed in the Calvin cycle. And third, it takes fewer photons to optimize the flux to 23BD (compared to optimizing for biomass production).

The production pathway of 23BD we added to the model requires NADPH, but not ATP (see Fig. 3.6B). Therefore, we expected a lower J_{ATP}/J_{NADPH} ratio compared to the optimization of specific growth rate. Our simulations show that, when optimizing for 23BD production, the percentage of J_{PSI} devoted to AEF pathways is much lower for both the CLLS and LLS (Figs. 3.3B-C vs. Figs. 3.7B-C). As a result, indeed, optimization of 23BD production required a lower J_{ATP}/J_{NADPH} ratio. This ratio is relatively close to the ratio that is obtained via a linear electron flow. Therefore, one may expect that the AEF knock-out yields a similar flux towards 23BD.

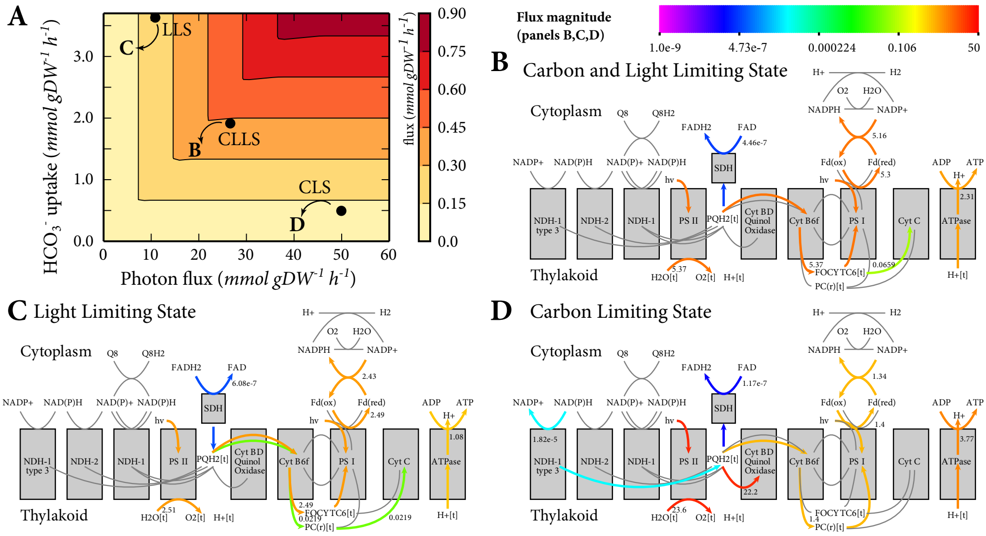


Figure 3.7. We used FBA to explore the properties of the *Synechocystis* model and map under various conditions (i.e. CLLS, LLS, and CLS), while optimizing for 23BD production. *(A) predicted flux to 23BD as a function of carbon and light uptake rates. (B-D) visualization of predicted FBA fluxes for the three scenarios (CLLS, LLS, and CLS respectively) indicated in panel A. Arrows indicate flux direction; the color bar indicates flux magnitude.*

However, surprisingly, an AEF knock-out in the CLLS and LLS resulted in a decrease of 12% in 23BD production production rate. As it turns out, the relatively small fraction of flux through the AEFs is eventually used for HCO_3^- uptake. Without the AEF pathways, less HCO_3^- can be imported into the cell, which results in a lower production of 23BD. In contrast, in the CLS, where light is available in excess, many metabolic routes lead to an optimal FBA solution, even though the objective that is optimized for may differ—in the case of our simulations, the objective was either growth (Fig. 3.3D) or biofuel production (Fig. 3.7D). Consequently, the characteristics of the system in a carbon limiting state are quite similar between these two objectives (cf. Figs. 3.3D and 3.7D).

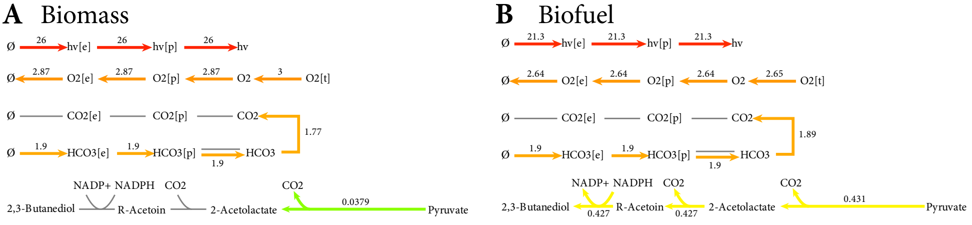


Figure 3.8. Our tool allows for simple identification of factors that contributed to the $J_{RuBisCo}/J_{hv}$ difference between optimization of biomass (panel A) and 23BD production (panel B). *These factors are: (i) HCO_3^- is not entirely converted to CO_2 when biomass is synthesized, (ii) fewer photons are necessary to optimize the biofuel pathway, and (iii) additional CO_2 is released, which is subsequently fixed in the Calvin cycle. In addition, a different oxygen evolution is predicted.*

Implementation

With the development of the first interactive map of cyanobacterial metabolism, we provide the community with a graphical means of interpreting the large amounts of data generated by high-throughput experimental or computational methods. Given the absence of commonly available visualization options for results of analyses of GSSMs, our main aim was the creation of an immediately useful map of *Synechocystis* metabolism. Visualization of heterogeneous data on this map is enabled by the standalone VoNDA package or the integration into FAME.

Besides providing a pathway-oriented, human-friendly overview of the contents of the GSSM of *Synechocystis* metabolism, we included a number of features in our graphical map that facilitate interpretation despite not being common practice in current-day visualization efforts. An important one is the choice to include multiple instances of common metabolites, such as ATP, ADP, NAD(P)(H), and other co-factors. These metabolites are included on the map in various locations, which greatly reduces the clutter that would arise if only one instance of each were drawn (graph based visualization solutions tend to do this).

Other helpful features are hidden when they are not in use, but deserve a brief explanation here. Space has been reserved on the map to display flux values for reactions and the direction in which the reaction takes place. An example are reaction-associated squares, which can be used for the visualization of gene expression data (Fig. 3.1C and Fig. 3.5). Each metabolic reaction has three such squares associated with it; if a reaction has more than three genes associated with it, we visualize the three genes with the highest differential expression value. Furthermore, when displaying run results generated in FAME, metabolite names link to a page with a list of fluxes that produce and consume the respective metabolite. Finally, when displaying run results, the lines that represent reactions are color-coded to represent the magnitude of the flux through them (or the variability of this flux, in the case of FVA results), and an explanatory color bar is displayed at the top of the image.

Exploiting *Synechocystis*' metabolic map for other micro-organisms

To alleviate the effects of our design decisions on users that would have favored another approach, we created the entire map in SVG, an open source image format. Our GSSM uses the BiGG nomenclature (Schellenberger et al., 2010), and therefore, our map is of direct value for all GSSMs that utilize this commonly used nomenclature. In addition, all identifiers on the map can be easily migrated to cognate identifiers in another nomenclature system (e.g. (Lang et al., 2011; Bernard et al., 2012; Kumar et al., 2012)). More importantly, the map can be modified to suit different species than *Synechocystis*. The addition of reactions and metabolites to the map can be done using either text or image manipulation software; when using an image editor, one must make sure that each new element is somehow assigned the correct identifier. We provide an online SVG-editor in FAME, which allows for easy manipulation of the metabolic map, including any results superimposed on it. While the quality of any customized map strongly depends on the knowledge and skill of whoever modifies it, and of the quality of the model it describes, this work presents the community with a solid basis from which to start this process. We provide an example of how one can go about this in Materials and Methods and in Supplemental Fig. S3.2. Our demonstration of the expansion of the map and model with a biofuel production pathway not only illustrates the extensibility of our work, but also how adequate visualization can lead to novel biological insights.

Regardless of any discussion about its implementation, the presentation of the first map of metabolism that is both computer-friendly and human-friendly will surely help advance our understanding of *Synechocystis*, but also of other organisms to which the map can be adapted. By offering this resource not only as a flat file, but also as an integrated part of an interactive online analysis environment, we expect that our map will be of even greater use to the community.

3.3 Conclusions

The increased pace at which experimental data can be generated has fueled genome-scale research efforts, which presently call for genome-scale interpretation efforts. For the case of *Synechocystis*, an important model organism for biotechnological applications, we now enable the interpretation of these data from a biological (pathway) perspective by presenting the first comprehensive graphical map of its metabolism. This is the first results interpretation tool that is graphical, interactive, machine-friendly, and, importantly, extensible. We expect that it will add great value to the community of cyanobacterium researchers, but also to the scientific community as a whole.

3.4 Methods

Adaptation of the *Synechocystis* model and creation of the corresponding map

We present a metabolic map and a corresponding stoichiometric model of *Synechocystis* metabolism consisting of 904 reactions, 816 species, and 686 genes. The model we present here is an extension of the iJN678 model described in (Nogales et al., 2012). Pathways that were amended include the tricarboxylic acid cycle (Zhang and Bryant, 2011), arginine metabolism (Schriek et al., 2007, 2008, 2009), and proline metabolism (Knoop et al., 2010). A full list of the modifications we made can be found in Supplemental Data S3.1. For example, we modified the succinate dehydrogenase reactions in iJN678 to ensure that $FADH_2$ can be synthesized and degraded (originally, iJN678 was unable to degrade $FADH_2$). This modification also allowed us to extend arginine metabolism, where $FADH_2$ is synthesized. To facilitate the interpretation of results deriving from the model, we decomposed the biomass function into reactions towards nine major components (e.g. DNA, RNA, and protein). This allows users to track the uptake and destination of each of these components individually (Supplemental Fig. S3.2).

In spite of its usefulness to biotechnology, the biological relevance of 23BD to organisms is relatively low, which makes it a rare find on metabolic maps, and our initial version of the *Synechocystis* map was no exception. The first reaction, the conversion of two pyruvate molecules into 2-acetolactate and CO_2 is an essential metabolic reaction and was therefore already on our map. We added two new reactions to our map (R_ACLDC and R_23BDD), as well as the metabolites they connect (R-acetoin and 23BD). We also added exchange reactions to the map and model.

In total, we added 44 reactions and 22 species, removed 7 reactions, and we revised incorrect mass and charge balancing for 19 reactions. We refer to our new version of the model as iTM686 (Supplemental Data S3.3 and S3.4); the model can be accessed or downloaded in the SBML Level 3 format at http://f-a-m-e.org/synechocystis. The model's constraints and objectives are represented using features from the recently launched Flux Balance Constraints (Olivier and Bergmann, 2013).

Analyses performed on the model

We analyzed all instances of the *Synechocystis* model using Flux Balance Analysis (FBA) and Flux Variability Analysis (FVA). As these analyses hinge upon the quasi-steady-state assumption, they can only predict steady-state flux distributions. More details can be found in Chapter 2. In-depth descriptions of these computational methods are abundant in literature. We refer to Price et al. (2004), Orth et al. (2010), and Santos et al. (2011) for an overview of the technical and methodological details. Throughout this chapter, wherever we mention performing an FBA, this FBA was performed with the additional objective of minimizing the absolute sum of fluxes to obtain a more biologically sensible solution. The details of all simulations, including the constraints and objectives that were used are described in Supplemental Data S3.2.

Integration with FAME

To demonstrate the functionality of the map as well as to facilitate its use, we added a *Synechocystis*-specific section to FAME, the Flux Analysis and Modeling Environment (Boele et al., 2012). Visiting http://f-a-m-e.org/synechocystis causes the *Synechocystis* GSSM (iTM686) and our map to be loaded. The user is then able to view, edit, and run the model, and to have analysis results visualized on the map we present in this chapter. Though the ability to visualize results on custom-made maps was already present in FAME, the URL described above pre-loads the model and map and eliminates the need for loading several files separately.

In addition, we expanded FAME with an in-browser image editor, and with the ability to visualize gene expression data from microarray profiling experiments on maps that support this kind of data. We pre-loaded the results of 98 experiments from the CyanoExpress microarray dataset (Hernandez-Prieto and Futschik, 2012) into the *Synechocystis* section of FAME. These data sets can be selected and superimposed on the map for analysis, as a demonstration of the ability of the *Synechocystis* map to display gene expression data as well as flux analysis results.

Acknowledgments

The authors would like to thank Brett Olivier, Pascal van Alphen, Philipp Savakis, Andreas Angermayr, and Klaas Hellingwerf for helpful discussions and tips for developing the data integration and visualization tool.

Supplemental Material

The following materials are available at http://persistent-identifier.org/?identifier=urn:nbn:nl:ui:18-23538 and in the online version of this article on which this chapter is based on.

Supplemental Data S3.1. A full list of modifications made to the GSSM iJN678.

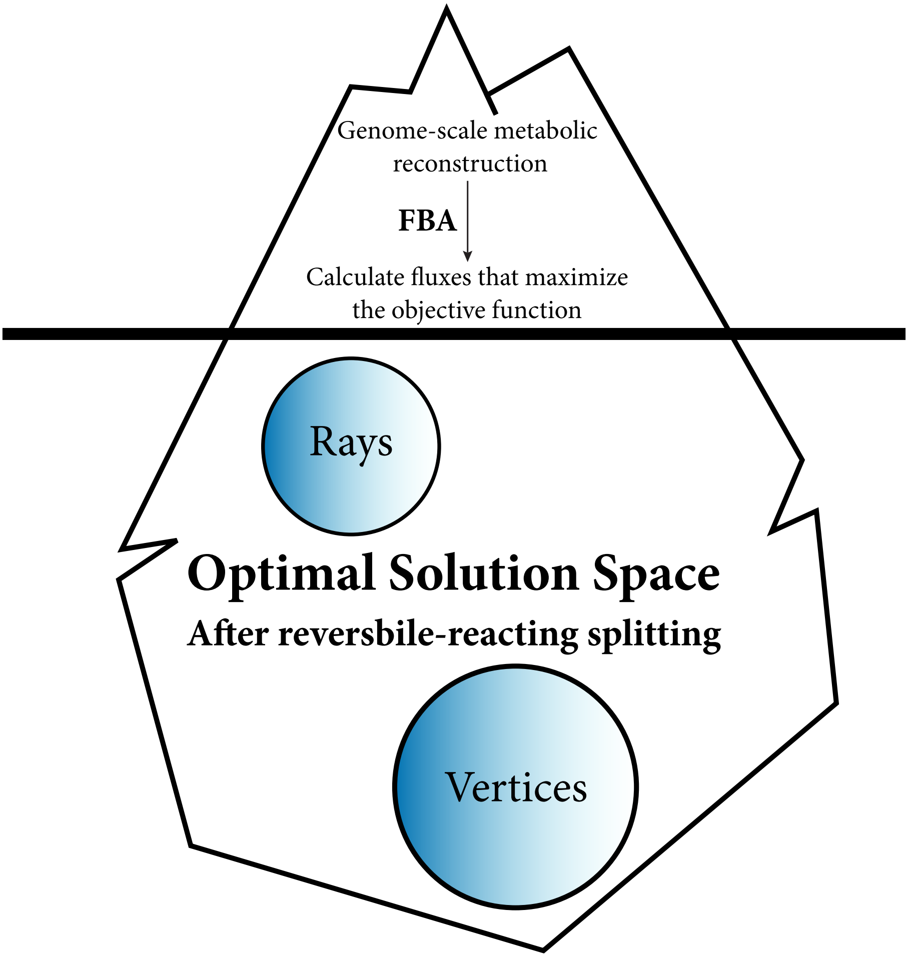
Supplemental Data S3.2. FAME batch commands to reproduce the results presented in this chapter.

Supplemental Data S3.3. *Synechocystis* metabolic network file.

Supplemental Data S3.4. *Synechocystis* metabolic network overview.

This chapter is based on the publication:

Maarleveld T.R., Wortel M.T., Olivier B.G. Teusink B., and Bruggeman F.J. (2015) *"Interplay between constraints, objectives and optimality for genome-scale stoichiometric models of metabolism"*, **PLoS Computational Biology**, 11, 4:e1004166.


Genome-scale metabolic reconstruction
FBA
Calculate fluxes that maximize the objective function
Rays
Optimal Solution Space
After reversbile-reacting splitting
Vertices

The tip of the iceberg: The predictions made by FBA are the tip of the ice berg; there typically exists a multidimensional optimal solution space of flux distributions that gives rise to optimization of the FBA objective function.

Interplay between constraints, objectives, and optimality for metabolic highways

4

Abstract

High-throughput data generation and genome-scale stoichiometric models have greatly facilitated the comprehensive study of metabolic networks. The computation of all feasible metabolic routes with these models, given stoichiometric, thermodynamic, and steady-state constraints, provides important insights into the metabolic capacities of a cell. How the feasible metabolic routes emerge from the interplay between flux constraints, optimality objectives, and the entire metabolic network of a cell is, however, only partially understood. We show how optimal metabolic routes, resulting from flux balance analysis computations, arise out of elementary flux modes, constraints, and optimization objectives. We illustrate our findings with a genome-scale stoichiometric model of *Escherichia coli* metabolism. In the case of one flux constraint, all feasible optimal flux routes can be derived from elementary flux modes alone. We found up to 120 million of such optimal elementary flux modes. Optimal flux routes no longer depend exclusively on elementary flux modes when we impose additional constraints; new optimal metabolic routes arise out of combinations of elementary flux modes. The solution space of feasible metabolic routes shrinks enormously when additional objectives—e.g. those related to pathway expression costs or pathway length—are introduced. In many cases, only a single metabolic route remains that is both feasible and optimal. This chapter contributes to reaching a complete topological understanding of the metabolic capacity of organisms in terms of metabolic flux routes, one that is most natural to biochemists and biotechnologists studying and engineering metabolism.

4.1 Introduction

Research in biotechnology and medicine benefits from understanding the metabolic capacity of organisms, including their sensitivities to genetic and environmental changes. Genome-scale stoichiometric models (GSSMs) of metabolism (Edwards and Palsson, 1999, 2000) and the availability of annotated genome sequences have greatly accelerated metabolic research. The combined use of high-throughput metabolomics data, comprehensive protocols (Thiele and Palsson, 2010), and (automated) reconstruction tools (Henry et al., 2010) has resulted in an explosion in the number and size of GSSMs of metabolism (Price et al., 2004; Oberhardt et al., 2009). Stoichiometric simulation has become an indispensable tool to deal with these large models, used in biotechnology (Teusink and Smid, 2006; Nogales et al., 2013) and medicine (Raman et al., 2005; Shlomi et al., 2011).

The most common stoichiometric mathematical method is Flux Balance Analysis (FBA) (Lee et al., 2006; Orth et al., 2010), which—given certain capacity constraints on fluxes—optimizes an objective function, for example, the biomass production flux (Feist and Palsson, 2010). The accuracy of FBA predictions depends on the availability of realistic flux constraints, which can be derived from experimental data. Generally, there are insufficient flux constraints to obtain a single unique solution and a large space of optimal flux distributions results (Mahadevan and Schilling, 2003; Reed and Palsson, 2004; Kelk et al., 2012). These alternative flux distributions give an impression of the robustness of a metabolic network (Stelling et al., 2002), but not every alternative is equally favorable for the organism. In some environments organisms are strongly selected for yield, almost regardless of the protein burden, while in other environments the protein burden has a significant impact. The solution space can be analyzed further with secondary objectives (Blank et al., 2005; Snitkin et al., 2008; Grafahrend-Belau et al., 2009; Mazumdar et al., 2009; Collins et al., 2012), e.g. minimization of the number of active fluxes (Schuetz et al., 2007) or the sum of absolute fluxes (Holzhütter, 2004), which have been used as proxies for maximization of the protein expression efficiency and minimization of the protein burden, respectively.

Analyzing the solution space and optimizing secondary objectives requires adequate mathematical and computations methods. Several approaches were proposed to give insight into the geometry of the optimal solution space (Mahadevan and Schilling, 2003; Burgard et al., 2004; Reed and Palsson, 2004; Wiback et al., 2004; Bilu et al., 2006; Bordel et al., 2010), which is mathematically represented by a polyhedron (Grötschel et al., 1988). Flux Variability Analysis (FVA) (Mahadevan and Schilling, 2003) and Flux Coupling Analysis (FCA) (Burgard et al., 2004) provide valuable information on the boundaries of the solution space, but do not give understanding in terms of metabolic routes. Such an understanding would be extremely helpful, as most biologists intuitively think in terms of metabolic routes.

Characterization of the optimal solution space provides valuable insight into how our limited knowledge of constraints affects the prediction of a metabolic state of an organism. The recently developed method, CoPE-FBA (Comprehensive Polyhedron Enumeration FBA) (Kelk et al., 2012), enumerates the vertices, the corner points of the optimal solution space. The number of vertices originates from the feasible, alternative metabolic routes through a small number of subnetworks, consisting only of reactions with correlated flux variability (Fig. 4.1). This method provides the structural insights that FVA and FCA lack, and explains the typical combinatorial explosion of the vertices; the optimal solution space can easily have millions of vertices that arise from independent combinations of alternative flux routes through only a few, small segments of the metabolic network. However, CoPE-

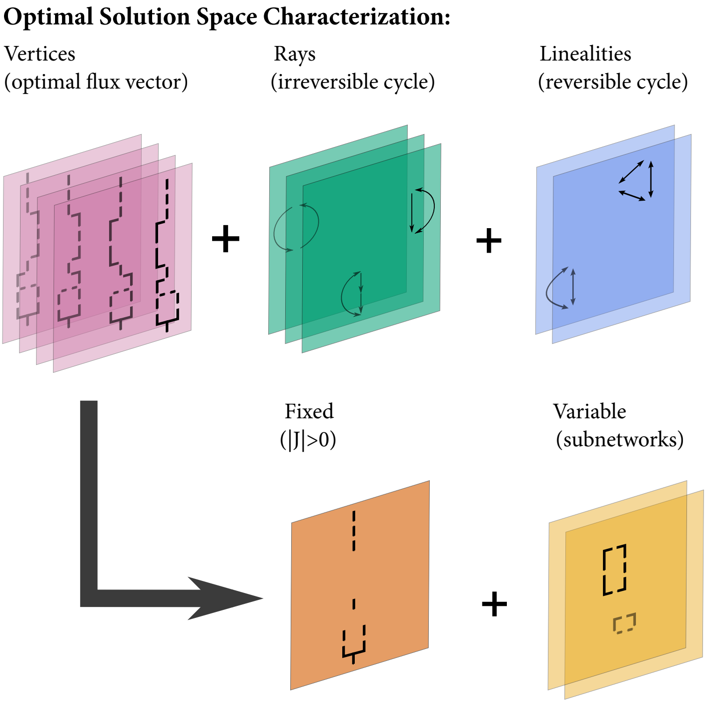


Figure 4.1. Characterization of the optimal solution space of metabolic models. *The optimal solution space can be characterized by three topological features: vertices, rays, and linealities. Typically, optimal solution spaces of microbial GSSMs are characterized by many vertices and only a few linealities and rays. Linealities do not exist when reversible reaction are split. Vertices can be described by a fixed active part which is identical for each vertex and a variable part, a few CoPE-FBA subnetworks (Kelk et al., 2012). We refer to Fig. S4.1 for examples of rays we found in the E.coli iAF1260 GSSM of metabolism.*

FBA suffers from computational difficulties, it is slow, and—perhaps more important—the provided solution does not yield all non-decomposable flux routes in the optimum, limiting the use of CoPE-FBA. For instance, it cannot be used to assess the influence of secondary objectives on the solution space.

We aim to obtain a better understanding of the interplay between constraints, objectives, and optimality for GSSMs. We uniquely characterize the optimal solution space by adjusting CoPE-FBA to split each reversible reaction into two irreversible reactions; this yields all non-decomposable flux routes in the optimum. Developing a computational method to efficiently enumerate all non-decomposable flux routes in the optimum is the topic of the next chapter. In this chapter, we start by illustrating the differences between the CoPE-FBA outcomes of metabolic models with and without reversible-reaction splitting. Next, we explain the relationship between these vertices and elementary flux modes (EFMs) with an optimal substrate-product yield. Finally, we show that secondary objectives typically collapse the optimal solution space to a unique solution (a vertex) or to a small set of vertices, using the iAF1260 GSSM of *Escherichia coli* metabolism.

4.2 Results

Characterization of the optimal solution space: Illustration with a toy model

We developed the toy network shown in Fig. 4.2A to illustrate: (i) the characterization of the optimal solution space of an FBA in terms of metabolic flux routes, (ii) that reversible-reaction splitting guarantees finding all non-decomposable metabolic flux routes in the optimum, (iii) the relationship between vertices and optimal-yield EFMs, and (iv) the optimization of secondary objectives over the optimal solution space. Our toy network consists of 18 metabolites and reactions where the source metabolite X and sink metabolite Y are considered boundary metabolites. All reactions, besides the reactions where ATP and ADP act as cofactors, are isomerization (uni-uni) reactions. Reversible reactions are illustrated by two headed arrows.

For our FBA model, we selected maximization of the flux through reaction R18 as our objective function, Z_{obj}. To constrain the solution space we used one inequality constraint, $J_1 \leq 2$. Throughout this chapter, we call this type of (inequality) constraint a restricting non-zero flux constraint. The resulting FBA is formulated as the following linear program:

$$\begin{aligned}
&\text{Maximize } Z_{obj} = J_{18} \\
&\text{subject to } \boldsymbol{NJ} = \boldsymbol{0}, \\
&\qquad -\infty \leq J_{rev} \leq \infty, \qquad J_{rev} \in \text{reversible reactions}, \\
&\qquad 0 \leq J_{irrev} \leq \infty, \qquad J_{irrev} \in \text{irreversible reactions}, \\
&\qquad 0 \leq J_1 \leq 2.
\end{aligned} \tag{4.1}$$

Here $\boldsymbol{NJ} = \boldsymbol{0}$ is the steady-state constraint with $\boldsymbol{N}$ as stoichiometric matrix and $\boldsymbol{J}$ as flux vector (or flux pathway). Simple metabolic models can be optimized by hand, but linear programming is required for the solution of any realistic GSSM. FBA optimization confirmed that, for this set of capacity constraints, maximization of our objective function gives $J_{18} = 1$.

Characterization of the optimal solution space for a metabolic model with reversible reactions

Several flux pathways maximize the objective $J_{18} = 1$; our FBA model is underdetermined. We can describe the optimal solution space with the Minkowski sum (see Eq. (2.21)) in terms of three mathematical objects: linealities, rays, and vertices (Fig. 4.1) (Grötschel et al., 1988). Each of these mathematical objects relate to a topological motif in a metabolic network.

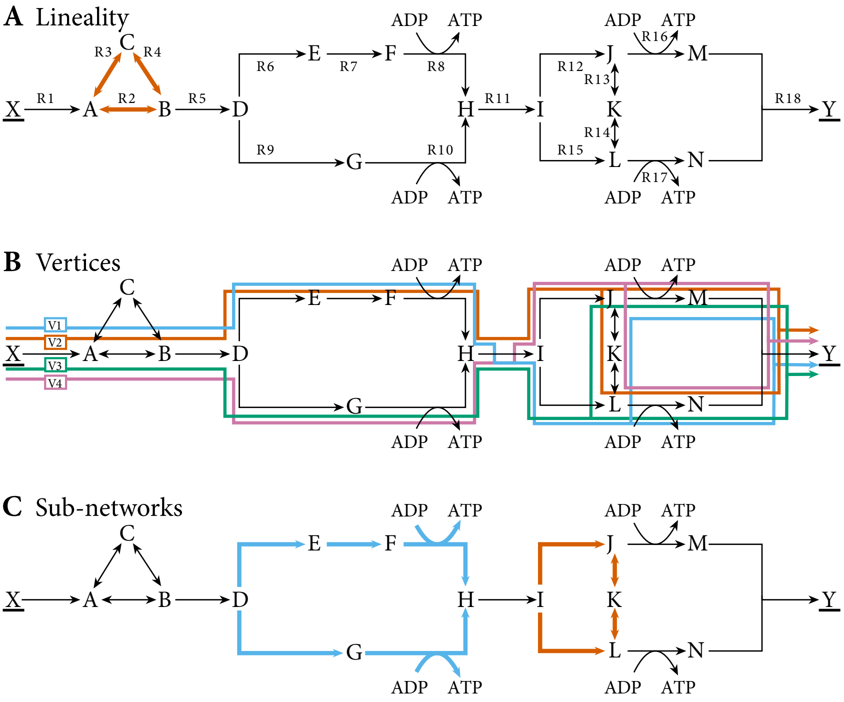


Figure 4.2. The optimal solution space of a toy model with reversible reactions. *Metabolites (capital letters) are converted by reversible (two headed arrows) and irreversible (single headed arrows) reactions to achieve the conversion of X to Y (underlined metabolites are boundary species). The forward direction of reversible reactions is defined from left to right or from top to bottom, and a backwards flux is denoted by a minor sign (-R13 indicates conversion from K to J). We maximized the flux through R18 with FBA, subject to steady-state constraints and* $J_1 \leq 2$*, where* J_1 *is the flux through reaction R1. The optimal solution space is characterized by one lineality (panel A) of reactions* {*R2–R4*} *(red) and (panel B) four vertices that arise from two branches at intersections D and I: V1 (blue), V2 (red), V3 (green) and V4 (purple). (C) two CoPE-FBA subnetworks illustrate the alternatives that create the four vertices (panel B); in subnetwork one (blue) these are* {*R6–R8*} *(V1 and V2) and* {*R9–R10*} *(V3 and V4), and in subnetwork two (red)* {*R12–R14*} *(V2 and V4) and* {*R15, -R13, -R14*} *(V1 and V3).*

Linealities are reversible cycles or input-output pathways (boundary to boundary metabolite(s), see Fig. S4.1 for an example) that indicate in which flux directions the optimal solution space is unbounded. Linealities cease to exist when we split each reversible reaction into two irreversible reactions later. The set of reactions R2, R3, R4 (i.e. {R2–R4}) form a lineality (Fig. 4.2A). The reactions can take any value, but there must be a net flux of two through {R2–R4} that converts A into B.

Rays are irreversible (thermodynamically infeasible) cycles or input-output pathways. Our toy model does not contain a ray. If at least one of the reactions R2, R3 or R4 would have been irreversible, {R2–R4} would have been a ray (Fig. 4.2A).

Vertices are the corner points of the FBA polyhedron—the solution space of optimal flux distributions—and therefore they cannot be represented as a convex combination of other optimal flux pathways (they are non-decomposable). Convex combinations of neighboring vertices form the facets—“edges”—of this polyhedron. Our toy model contains four vertices termed V1–V4 (Fig. 4.2B). A combination of these vertices and cyclic networks can be used to represent every optimal flux vector.

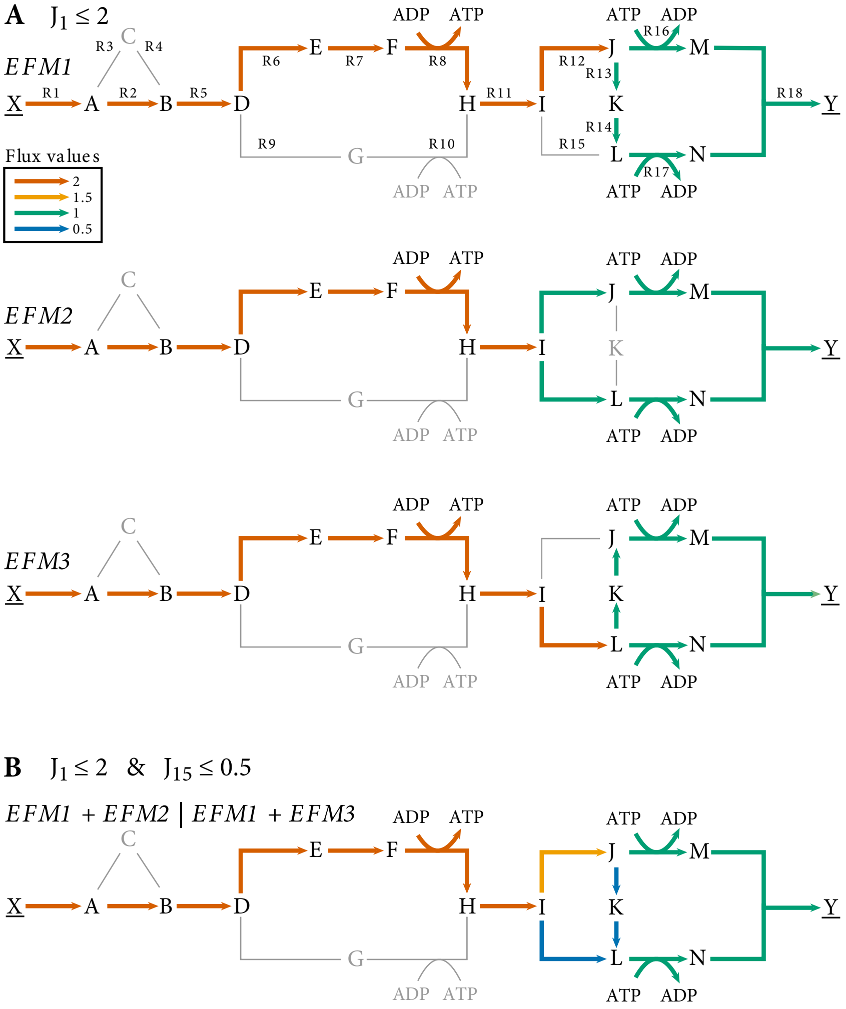

A $J_1 \leq 2$
EFM1
X A B C D E F G H I J K L M N Y
R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15 R16 R17 R18
ADP ATP
Flux values
2
1.5
1
0.5
EFM2
EFM3
B $J_1 \leq 2$ & $J_{15} \leq 0.5$
EFM1 + EFM2 | EFM1 + EFM3

Figure 4.3. Vertices correspond to optimal-yield EFMs or convex combinations of those EFMs. *Vertices correspond to optimal-yield EFMs if they are restricted by one flux constraint (panel A) and to a convex combination of EFMs if they are restricted by more than one flux constraint (panel B). Colors represent different flux values (red = 2, orange = 1.5, green =1, and blue = 0.5). (A) visualization of EFM1, EFM2, and EFM3 (out of the twelve optimal-yield EFMs normalized to* $J_{18} = 1$*). Both EFM1 and EFM3 have a corresponding vertex with and without splitting, whereas EFM2 has only with splitting a corresponding vertex. (B) taking a convex combination of EFM1 and EFM2 or EFM1 and EFM3 (panel A) corresponds to a vertex when the constraints are* $J_1 \leq 2$ *and* $J_{15} \leq 0.5$.

Vertices originate from combinations of alternative flux distributions in CoPE-FBA subnetworks, quickly leading to a combinatorial explosion (Kelk et al., 2012). In the optimum, these subnetworks consist of reactions with correlated flux variability and have a fixed net input-output stoichiometry of reactants and products. The toy model contains two subnetworks (Fig. 4.2C). In the subnetwork consisting of {R6–R10}, the alternative flux distribution {R6–R8} is anti-correlated with alternative flux distribution {R9–R10} and the reactions within these sets are positively correlated with each other (Fig. S4.2). Both sets of reactions have an identical input-output relationship: D + ADP → H + ATP. Multiplying the number of alternative flux distributions through each subnetwork gives the total number of vertices; both subnetworks of the toy model have two alternative flux distributions—the lower and the upper branch—which gives a total of four vertices (Fig. 4.2B).

The disadvantages of metabolic models with reversible reactions

The representation of the network in its current form—where reversible reactions are not split into two irreversible reactions—complicates the characterization of the optimal solution space; not all non-decomposable optimal pathways are vertices and the set of vertices is not unique. This set of vertices corresponds to the minimal generating set of the optimal solution space, a well-known concept in EFM analysis (Wagner and Urbanczik, 2005). For our toy model, we enumerated all (optimal-yield) EFMs to illustrate the difference between the set of vertices we enumerated with CoPE-FBA and the set of non-decomposable flux pathways in the optimum.

If there is only one upper bound (as in the toy model), all non-decomposable optimal pathways are instances of the optimal-yield EFMs. EFM analysis showed that our toy model has thirteen EFMs of which twelve are operational modes that produce Y with an optimal yield (see also Fig. S4.3 for a more detailed analysis). Thus, for only one third of the optimal-yield EFMs, there exists a corresponding vertex (Fig. S4.3B). The remaining eight EFMs do not have corresponding vertices, because: (i) two non-decomposable optimal pathways are a convex combination of different vertices; 1/2 V1 (EFM1) + 1/2 V2 (EFM3) = EFM2 (Fig. 4.3A) and (ii) six more optimal pathways are a linear combination of vertices (or of the two convex combinations) and the lineality (with flux through {R3, R4} rather than {R2}). Because of these additional non-decomposable optimal pathways, finding all the possible pathways that optimize a (secondary) objective directly from the vertex representation is not possible—we shall return to this later.

Metabolic models usually contain many sub-optimal modes—modes with a lower yield—or non-operational modes—modes that cannot produce any objective flux. Our toy network was designed to not contain suboptimal operational modes. The non-operational mode is the lineality shown in Fig. 4.2A.

The EFM analysis further showed that the optimal solution space can be described with different sets of vertices. For instance, CoPE-FBA gave that {R2} was part of each vertex and {R3, R4} was only part of a lineality. Reversing this situation by making {R2} part of the lineality and {R3, R4} part of each vertex is also a valid description of the optimal solution space. This means that this decomposition is not unique.

Reversible-reaction splitting yields all non-decomposable flux pathways in the optimum

To solve the issues with uniqueness and completeness, we exploited an existing technique in the field of EFM analysis: Splitting reversible reactions into separate forward and backward reactions (Klamt and Stelling, 2003; Wagner and Urbanczik, 2005). In the split model, the vertices are instances of exactly the non-decomposable pathways in the optimal state. Since all reactions are irreversible after the splitting procedure, linealities do not exist anymore. The split toy model contains seven rays, because each split reversible reaction forms an additional ray and two more rays from {R2–R4} in forward and backward direction (see Fig. 4.4A). After splitting, rays no longer signify thermodynamically-infeasible irreversible cycles. Each forward and backward reaction together forms a new EFM, but the set of optimal-yield EFMs is identical before and after splitting (Fig. S4.3A and S4.3D), which was also proven by Gagneur and Klamt (2004).

For our toy model, each optimal-yield EFM now has a corresponding vertex (Fig. S4.3E); the vertices lie on their corresponding EFMs. This means that our toy model contains now twelve rather than four vertices. To explain the difference between the set of vertices with and without splitting, we first focus on {R2–R4}. With splitting, the model contains vertices with both {R2} and {R3, R4}, while without splitting vertices only contain either {R2} or {R3, R4}. The variability in reactions {R2–R4} causes the number of vertices to double (2 × 2 × 2 vs. 2 × 2), because {R2–R4} is also a CoPE-FBA subnetwork (Fig. 4.4B) in the split model. The second difference originates from the CoPE-FBA subnetwork described by {R12–R15}. The flux distribution through {R12, R15} (see also EFM2 Fig. 4.3A) cannot be obtained via a convex combination of alternative flux distributions and is, therefore, now also a vertex. This difference causes the number of vertices to further increase from eight to twelve (2 × 2 × 3 vs. 2 × 2 × 2).

Additional non-zero flux constraints cause a dissimilarity between optimal-yield EFMs and vertices

After reversible-reaction splitting, the set of optimal-yield EFMs corresponds to the set of vertices if there is only a *single* non-zero restricting flux constraint (for proof see Box 4.1). With more constraints this is not necessarily the case, because EFMs are based on stoichiometry and thermodynamics alone, while vertices also depend on flux constraints. We illustrate this by discussing examples of different types of flux constraints on the sets optimal-yield

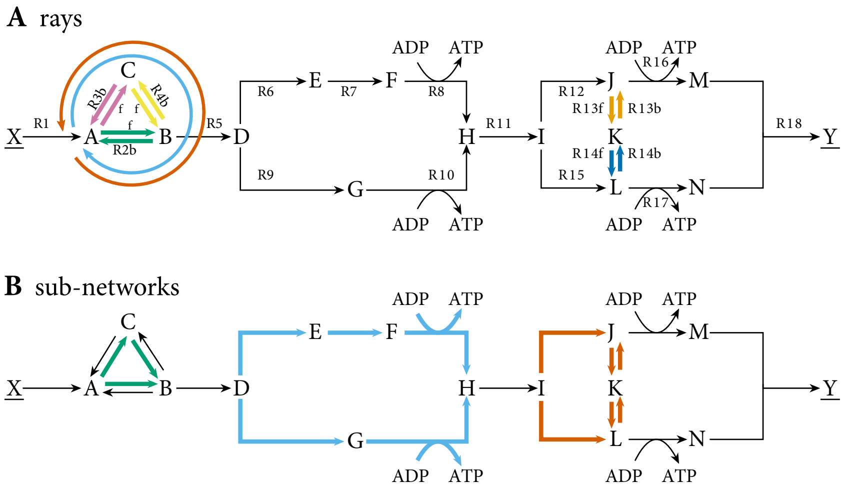


Figure 4.4. Reversible-reaction splitting guarantees finding all non-decomposable flux pathways in the optimum. *Metabolites (capital letters) are converted by irreversible reactions to achieve the conversion of X to Y (underlined metabolites are boundary species). Split reversible reactions are denoted as R3f and R3b. We maximized the flux through R18 with FBA, subject to the steady-state constraint and* $J_1 \leq 2$*. The optimal solution space is now characterized by seven rays (panel A) and twelve vertices which originate from three CoPE-FBA subnetworks (panel B). (A) the five split reversible reactions and* {*R2–R4*} *in forward and backward direction form together seven rays. (B) three subnetworks give rise to twelve vertices (*$2 \times 2 \times 3$*). The third subnetwork (red) now has a third alternative flux distribution* {*R12, R15*} *which was without reversible-reaction splitting a convex combination of the other two flux distributions,* {*R12, R13f, R14f*} *and* {*R15, R13b, R14b*}*.*

EFMs and vertices: setting a flux to zero, adding a restricting constraint, and adding a demanding constraint.

Setting a flux to zero (e.g. an anaerobic growth condition) effectively removes a reaction from the system. The set of optimal-yield EFMs still corresponds to the set of vertices (of course all pathways using the removed flux, e.g. oxygen uptake, are absent). As an example, removing R15 from our toy network would result in the same set of four optimal-yield EFMs and vertices.

We illustrate the effect of adding an additional restricting non-zero flux constraint (e.g. an upper bound on oxygen uptake) in our toy model with: $J_{15} \leq 0.5$. With this constraint, $J_{18} = 1$ cannot be achieved with EFMs that include reaction R15 (e.g. EFM2 and EFM3 shown in Fig. 4.3A). The corresponding vertices are infeasible, because vertices are only defined in the optimal space (see also Fig. S4.3C and S4.3F for a more detailed analysis). The corner points of the new optimal solution space are now described by a different set of vertices: still feasible vertices and vertices that arose after adding the second constraint. Each newly introduced vertex arose from an infeasible vertex and a neighboring feasible vertex. An example of such a vertex is given in Fig. 4.3B, which corresponds to a convex combination of EFM1 and EFM2 or EFM1 and EFM3. In this example, the number of corner points of the optimal solution space decreased after adding the second flux constraint, but this is not a general outcome.

Demanding a flux through a reaction that decreases the substrate-product yield (e.g. ATP maintenance reaction) yields different vertices. An optimal-yield EFM through the demand reaction is then added to each vertex. The number of vertices increases if multiple optimal-yield EFMs coexist through the flux demanding reaction (which then form another CoPE-FBA subnetwork).

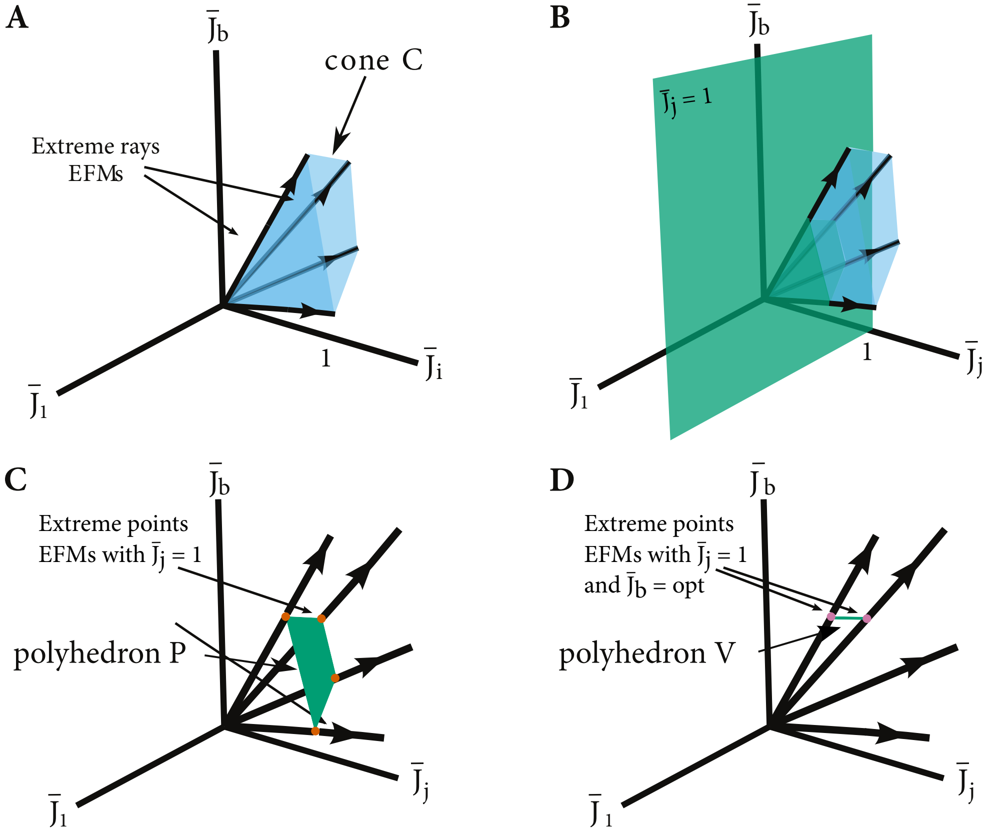


Figure 4.5. Schematic representation of the optimization of the biomass objective function (J_b). *We used reversible-reaction splitting to make all reaction rates positive. (A) the resulting solution space is a pointed cone C. Its extreme rays are the intersection of the planes with the zero vector. These extreme rays coincide with the EFMs as proven by Gagneur & Klamt (2004). (B) before we maximize the biomass objective function,* J_b*, we first fix the input reaction,* $J_j = 1$*. (C) the intersection of the cone C and the plane* J_j *defines the polyhedron P. (D) maximizing* J_b *(which is a linear function) gives the optimal solution space* V*. The corner points are vertices and optimal-yield EFMs with* $J_j = 1$*.*

Box 4.1: Proof

Here we describe the proof that after reversible-reaction splitting, each optimal-yield EFM has an instance that is a vertex if there is only a *single* non-zero restricting flux constraint. This proof is analogues to the proof that the solution to the optimization of a specific flux are always extreme rays as described by Wortel et al. (2014).

We want to maximize a linear objective function given a capacity constraint on an input reaction, $J_j \leq 1$. This is the only constraint, thus $J_j = 1$ in the optimum. This linear program (LP) can be formulated as

$$\max_{\boldsymbol{J}} \left\{ \boldsymbol{c}^T \cdot \boldsymbol{J} \middle| \boldsymbol{N}\boldsymbol{J} = \boldsymbol{0}, \boldsymbol{J}^{min} \leq \boldsymbol{J} \leq \boldsymbol{J}^{max}, \underbrace{J_j = 1}_{\text{hyperplane}} \right\}, \tag{B1}$$

where $\boldsymbol{N}$ is the stoichiometric matrix, $\boldsymbol{c}$ is a vector of coefficients that represent the contribution of each flux in vector $\boldsymbol{J}$ to the objective function. Reversible-reaction splitting simplifies this LP, defining all fluxes as positive. This introduces a new stoichiometry matrix $\bar{\boldsymbol{N}}$ and rate vector $\bar{\boldsymbol{J}}$,

$$\bar{\boldsymbol{J}}^{opt} = \arg\max_{\bar{\boldsymbol{J}}} \left\{ \boldsymbol{c}^T \cdot \bar{\boldsymbol{J}} \middle| \underbrace{\bar{\boldsymbol{N}}\bar{\boldsymbol{J}} = \boldsymbol{0}, \bar{\boldsymbol{J}} \geq 0}_{\text{C, a convex set}}, \underbrace{J_j = 1}_{\text{hyperplane}} \right\}. \tag{B2}$$

The feasible flux space is the cone C defined in Eq. (B2) (Fig. 4.5A). C is characterized by its extreme rays. The intersection of the cone C with the plane $J_j = 1$ (Fig. 4.5B) defines the solution space (Fig. 4.5C),

$$P = \left\{ \boldsymbol{J} \middle| C \cap J_j = 1 \right\}. \tag{B3}$$

The extreme points of the polyhedron P are the extreme rays of C. Gagneur & Klamt (2004) proved that these extreme rays are the EFMs of the original metabolic network. Next, we maximize the linear function $\boldsymbol{c}^T \cdot \bar{\boldsymbol{J}}$ over P which defines the optimal solution space (Fig. 4.5D),

$$V = \left\{ \boldsymbol{J} \middle| P \cap \boldsymbol{c}^T \bar{\boldsymbol{J}}^{opt} \right\}. \tag{B4}$$

The corner points of V are EFMs of C (but not every EFM is a corner point of V). Since we maximize a linear function over polyhedron P, the optimum is achieved at either a corner point (vertex), an edge or a face. The objective function is often simple and includes only a single or a few fluxes (e.g. biomass production), thus the optimal solution often coincides with a face of the (many-dimensional) polyhedron The vertices of this optimal face coincide with the vertices of the polyhedron P, and are therefore instances of EFMs.

In this proof we performed two EFM elimination steps: (i) EFMs that do not satisfy $J_j = 1$ are not part of P and (ii) EFMs that do not satisfy $\boldsymbol{c}^T \cdot \bar{\boldsymbol{J}}^{opt}$ are not part of V. By fixing $J_j = 1$ and maximizing $\boldsymbol{c}^T \cdot \bar{\boldsymbol{J}}$ we only delete corner points of C; we do not create new corner points (as would be the case when we include additional constraints). These corner points are defined as vertices of the optimal solution space, thus the set of vertices of V correspond to the set of optimal yield EFMs if there is only a *single* non-zero restricting flux constraint.

Secondary optimization collapses the optimal solution space

In this section, we demonstrate how secondary optimization simplifies after reversible-reaction splitting and that secondary optimization reduces the solution space to only one or a few vertices. As a secondary optimization objective, we used minimization of the number of active fluxes—hereafter pathway length P_L; see Eq. (4.5). Mathematically, this secondary optimization can be written as

$$\begin{aligned}
&\text{Minimize } P_L\\
&\text{subject to } \boldsymbol{NJ} = \boldsymbol{0},\\
&\qquad J_{18} = Z_{obj} = 1,\\
&\qquad -\infty \le J_{rev} \le \infty, \qquad J_{rev} \in \text{reversible reactions},\\
&\qquad 0 \le J_{irrev} \le \infty, \qquad J_{irrev} \in \text{irreversible reactions},\\
&\qquad 0 \le J_1 \le 2,
\end{aligned} \tag{4.2}$$

in which the output flux of the toy model is set to its maximal value obtained with the previous FBA, shown in Eq. (4.1). We used mixed-integer linear programming to select the flux pathway with the minimal P_L from the optimal solution space for one ($J_1 \le 2$) and two ($J_1 \le 2$, $J_{15} \le 0.5$) restricting flux constraints, which gave a minimal P_L of 11 and 12 respectively. Next, we determined the P_L for each optimal-yield EFM and vertex of the "non-split" and "split" model (see Fig. S4.4 for more details).

Without splitting, the vertices are instances of a subset of all possible non-decomposable pathways (EFMs) in the optimal state. Therefore, for the optimization of secondary objectives in the *non-split* model, we cannot focus solely on the vertices. We have to take into account the whole optimal solution space—the Minkowski sum of vertices, rays, and linealities (for details see Methods)—which is cumbersome.

Analyzing the effect of linealities and rays is counterintuitive because both linealities and rays represent cycles that catalyze no net conversion. This makes them independent from the chosen growth medium and objective function. However, without splitting, linealities and/or rays can influence secondary objectives when they share reactions with one or more vertices. In this case we can construct non-decomposable optimal flux pathways that are not vertices by taking, for instance, a linear combination of a vertex with a connected lineality. An example is the lineality described by {R2–R4} (Fig. 4.2A). Adding {R2, R3, -R4} to one of the four vertices gives rise to a new optimal flux pathway; one with more active reactions.

Taking a convex combination of vertices shortens P_L when more active reactions become inactive than vice versa. A reaction becomes inactive when it goes in different directions with the same flux in alternative flux distributions. When $J_1 \le 2$ is the only constraint, a convex combination of alternative flux distributions in {R12–R15} (Fig. 4.2C) shortens P_L

by one reaction: Specifically, R13 and R14 carry flux in both alternative flux distributions whereas in different directions. This analysis shows that the minimal P_L is a convex combination of vertices V3 and V4 (Fig. 4.2B). This optimal pathway becomes infeasible when we add the second flux constraint $J_{15} \leq 0.5$; then, vertex V4 minimizes P_L.

Reversible-reaction splitting allows us to directly enumerate all possible non-decomposable pathways in the optimal state; no convex combination turns active reactions inactive, because all fluxes are positive. The shortest optimal flux pathway is always a vertex (and corresponds to an optimal-yield EFM if it is restricted by only one non-zero flux constraint). Theoretically, multiple shortest vertices can co-exist. The fact that only a vertex or several vertices optimize the secondary objective is a specific result for pathway length minimization (see also Fig. S4.5). For instance, the optimal solution space after minimization of the sum of absolute fluxes as secondary objective (Eq. (4.7)) can consist of a line or a plane, besides (multiple) single point(s).

Concluding remarks about reversible-reaction splitting

Reversible-reaction splitting has many advantages for the characterization of the optimal solution space. We first summarize those advantages before we set out to analyze a GSSM:

(i) The vertices discovered with CoPE-FBA are all possible non-decomposable pathways in the optimal state. For the analyses of optimal flux pathways, we can, therefore, focus solely on vertices. These vertices can be compactly described by a set of subnetworks that describe all the variability in non-decomposable optimal flux pathways.

(ii) The optimal solution space is uniquely characterized.

(iii) Secondary optimization yields an optimal solution space consisting of one or multiple vertices.

(iv) Rays no longer signify thermodynamically-infeasible irreversible cycles.

Typically, splitting yields many more vertices and rays (each split reversible reaction forms an additional ray). We identified three different mechanisms that contribute to the increase in vertices. First, splitting can yield additional CoPE-FBA subnetworks that originate from rays or linealities with a input-output relationship different from zero. An example is the lineality given by {R02–R04} (Fig. 4.2A) that is a subnetwork with splitting (Fig. 4.4B). Second, optimal flux pathways that are convex combinations of vertices before splitting become vertices after splitting. We encountered such a case in the toy model where the convex combination of vertices V3 and V4 resulted in an additional vertex after splitting. And third, rays or linealities connected to CoPE-FBA subnetworks give rise to additional vertices. Imagine for the toy network, for instance, a reversible reaction that converts metabolite F into metabolite E (the reverse of reaction R7). Before splitting, R7 and this newly introduced reaction form a ray. After splitting, an additional vertex exists through this newly introduced reaction.

A real life example: *Escherichia coli* growing on glucose

We analyzed the realistic GSSM iAF1260 of *Escherichia coli* (*E. coli*) metabolism (Feist et al., 2007). By modifying the oxygen uptake constraint, we constructed three different FBA models of iAF1260 that depict aerobic, aerobic restricted, and anaerobic growth (for details see Methods). We set maximization of biomass production rate as the objective function. For these growth conditions, general CoPE-FBA results for both the model with and without reversible-reaction splitting are shown in Table 4.1. With splitting we found for each growth condition many more vertices (up to 120 million). Since we also used an ATP maintenance demand constraint of 8.39 mmol gDW^{-1} h^{-1}, our vertices are not instances of EFMs. Without this constraint, all vertices in both the aerobic and anaerobic growth condition are instances of their corresponding EFMs (not shown). We found only a few CoPE-FBA subnetworks which together completely reveal the variability in the vertices. Most reactions are inactive after optimizing the objective function (Table S4.1). In the remainder of this section, we use the results obtained from the model with splitting because this yielded all non-decomposable flux pathways in the optimum.

Gaussian and multimodal distributions of vertices after secondary optimization

We studied the distributions of objective values of a secondary optimization over the vertices obtained in the first optimization. For each vertex, we determined the pathway length P_L, pathway sum of absolute fluxes P_J, and pathway cost P_C (see Methods). Similar to work done by Shlomi et al. (2011), our protein cost definition was solely based on enzyme-synthesis cost. For instance, we did not take the protein lifetimes into account. Ignoring protein lifetimes implies that P_J and P_C are closely related; P_C is taking P_J multiplied with a protein cost for each individual reaction. In Fig. 4.6, we thus only show the results for P_L and P_C. Initially, we intuitively expected many vertices with intermediate P_L, P_J, and P_C, and few with relatively low or high P_L, P_J, and P_C. In other words, we expected a Gaussian-shaped distribution for both P_L, P_J, and P_C. As expected, the P_L was indeed Gaussian-shaped distributed for all tested growth conditions. This is illustrated by the dashed black lines in the top panel of Fig. 4.6 which correspond to a Gaussian distribution where we used the sample mean and standard deviation as input.

In contrast, P_C was clustered into distinct groups (i.e. a multimodal distribution). An accurate determination of pathway cost is a challenge and we hypothesized that a different cost function could show a different distribution. Therefore, we investigated four different definitions of protein cost: minimum, maximum, average, and equal (i.e. sum of absolute

Model	Toy model		*E. coli* iAF1260 model	
Reversible-reaction splitting	No	Yes	No	Yes
Growth condition			Anaerobic	
Total reactions	12	23	2374	3226
Rays	0	7	26	604
Linealities	1	0	1	0
Vertices	4	12	839808	120932352
Subnetworks	2	3	6	9
Model	*E. coli* iAF1260 model		*E. coli* iAF1260 model	
Reversible-reaction splitting	No	Yes	No	Yes
Growth condition	Aerobic restricted		Anaerobic	
Total reactions	2374	3226	2374	3226
Rays	25	602	25	602
Linealities	1	0	1	0
Vertices	1679616	40310784	31104	1492992
Subnetworks	4	4	6	8

Table 4.1. Characterization of the optimal solution space with and without reversible-reaction splitting. *The optimal solution space is typically characterized by many vertices and a few rays and linealities. Vertices originate from a combinatorial explosion of only a few CoPE-FBA subnetworks. Reversible-reaction splitting yields a unique characterization of this optimal solution space, but typically many more vertices (and rays) exist. With reversible-reaction splitting, there is one additional ray in the aerobic growth condition which concerns an irreversible cycle of oxygen.*

fluxes; P_J). When multiple proteins were associated to a particular reaction via an OR rule, the minimum, maximum or average was taken (for more details see Methods). In all cases, we found a multimodal distribution. Nonetheless, we did find an effect of the cost function; taking the “minimal” cost function typically resulted in the largest difference between both clusters, while taking the “equal” cost function typically resulted in the smallest difference between both clusters (Table S4.2, Fig. S4.6). These results show that for explaining the multimodal distribution of vertex cost, the effect of fluxes was much more important than the effect of protein costs.

We already explained that enumeration of a model with reversible reactions does generally not result in a unique characterization of the optimal solution space; different subsets of all non-decomposable flux pathways in the optimum exist. To illustrate the possible differences, we enumerated the *E.coli* iAF1260 model including reversible reactions for the same conditions of which results are shown in Fig. S4.7. Comparison with Fig. 4.6 shows that during aerobic growth conditions only two rather than four clusters were found.

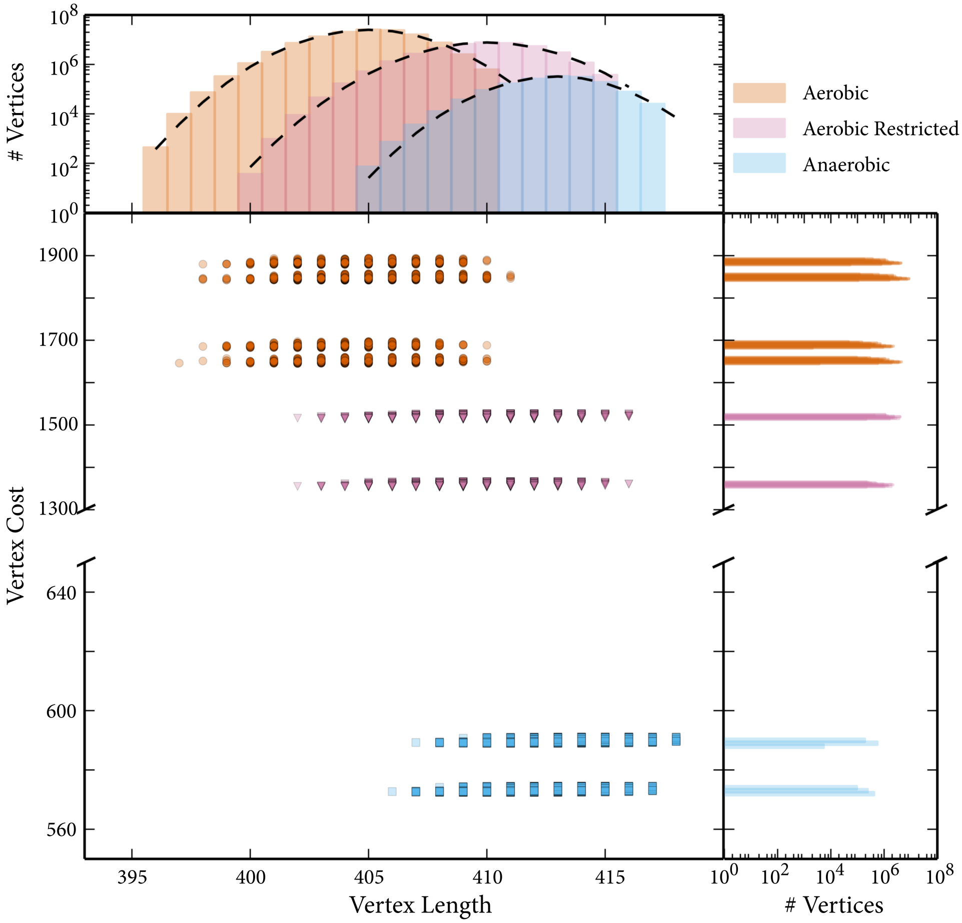


Figure 4.6. *E.coli* vertex cost follows a multimodal distribution. *For three growth conditions—aerobic (red, circle), aerobic restricted (purple, triangle), and anaerobic (blue, square)—we analyzed the vertex cost (P_C) and vertex length (P_L) of each vertex. Each dot in the main panel represents a vertex with a specific cost and length. Our results indicate that for E.coli the vertex length follows approximately a Gaussian-shaped distribution (dashed lines are Gaussian distributions with sample mean and sample standard deviation). Vertex cost follows a multimodal distribution; vertices are clustered in distinct groups with a specific cost. Due to file size limitations we only show a subset (10000) of vertices for all conditions in the scatter plot.*

CoPE-FBA subnetworks explain differences in secondary optimization

CoPE-FBA subnetwork analysis revealed the shape of the distributions of the length, sum of absolute fluxes, and cost of vertices. Several different CoPE-FBA subnetworks contributed to the total length difference and within the majority of these subnetworks we found alternative flux distributions with different lengths; the length distribution within the subnetworks were uniform or already Gaussian in shape. It is, therefore, not possible to reconstruct many

	E. coli iAF1260 growth condition		
	Aerobic	Aerobic restricted	Anaerobic
min(P_L)	432	36	72
min(P_J)	24	24	24
min(P_c)	1	1	1

Table 4.2. Secondary optimization can reduce the optimal solution space to a unique flux distribution. *For the E.coli iAF1260 GSSM, secondary optimization reduced the optimal solution space from 1–120 million vertices to about 1–432 vertices. When we minimize pathway cost (P_c) a unique optimal flux distribution exists, but many suboptimal flux distributions do exist (Fig. 4.6).*

vertices with a short or long vertex length, which explains the Gaussian-shaped distribution of vertex length.

Alternatively, one (aerobic restricted and anaerobic cases) or two subnetworks (aerobic case) explain the main differences in P_J and P_C. Within these subnetworks, we found two distinct modes—a relatively cheap and a relatively expensive mode. While some of these subnetworks were relatively large, our results show that the main cost difference in this particular subnetwork originates from only a few metabolic reactions (Table S4.3). As a consequence, we found many vertices with a relatively low P_C and many vertices with a relatively high P_C—a multimodal distribution of vertex cost. In the aerobic growth conditions, the cost difference mainly emerged from using different electron acceptors for the NADH dehydrogenase; cheap pathways used ubiquinone-8 and costly pathways used menaquinone-8 and/or demethylmenaquinone-8. In aerobic growth conditions ubiquinone-8 is the major quinone in *E. coli* (Unden and Bongaerts, 1997; Bekker et al., 2010). In anaerobic growth conditions, the cost difference mainly emerged from exploiting a different strategy for the ATP-dependent conversion from PEP and F6P to DHAP, G3P, and PYR in main carbon metabolism (Fig. S4.8).

Lastly, we studied the reduction of the solution space after secondary optimization. For the *E. coli* iAF1260 model with splitting, secondary optimization reduced the solution space to only one or a few vertices (Table 4.2). In case of P_L-minimization, only vertices can be optimal solutions, since convex combinations increase the number of active reactions. Compared to minimization of P_J and P_C, the solution space after minimization of P_L contained more vertices in all of the tested growth conditions. This was expected because P_L is solely based on the number of active reactions, specific flux values are not of interest. Taking these flux values into account typically results in more diverse outcomes. Hence, it is less likely to find as many vertices with a minimal P_J. Similarly, adding different protein costs to each reactions further diversifies these outcomes. As a result, the optimal solution space for P_C-minimization resulted in a unique flux distribution for all tested growth conditions.

4.3 Discussion

The recently developed computational method, CoPE-FBA (Comprehensive Polyhedra Enumeration Flux Balance Analysis) (Kelk et al., 2012), offers the premise of a simplified biological understanding of the optimal solution space of metabolic network models; a kind of understanding which is not possible with other popular methods such as Flux Variability Analysis (Mahadevan and Schilling, 2003) and Flux Coupling Analysis (Burgard et al., 2004). We further developed this method: Rather than enumerating the minimal generating set, we used reversible-reaction splitting (Wagner and Urbanczik, 2005; Klamt and Stelling, 2003) to enumerate all non-decomposable flux pathways in the optimum. This allows us to focus solely on the vertices for the analysis of optimal flux pathways.

The further development of CoPE-FBA facilitated in achieving a better understanding of how optimal flux pathways resulting from FBA arise out of EFMs, use of constraints, and optimality conditions. We recall that the vertices correspond to optimal-yield EFMs if there is *only* a single restricting flux constraint. Both restricting and demanding flux constraints modify the (optimal) solution space. Typically, the optimization problem remains underdetermined and an optimal solution space continues to exist. Adding additional constraints that concern all flux values in the model (e.g. protein cost constraints) can result in a unique solution. Then, the optimal state is an instance of an optimal-yield EFM if there is only a single restricting flux constraint. Alternatively, the optimal state corresponds to a convex combination of optimal-yield EFMs.

Other constraints that also concern all reactions, but not their flux values, such as minimal P_L, often leads to optimal solution spaces. While these objectives have been used frequently to find more realistic FBA outcomes (Blank et al., 2005; Snitkin et al., 2008; Grafahrend-Belau et al., 2009; Mazumdar et al., 2009; Collins et al., 2012), we showed that for both minimization of P_L and P_C, optimal solution spaces continue to exist (Table 4.2). This result shows that we should be careful drawing conclusions from predicted flux distributions after using a secondary objective. Using CoPE-FBA with only irreversible reactions allows for a straightforward identification of the origin of the remaining solution space. Specifically, these solution spaces originate from identically favorable pathways through CoPE-FBA subnetworks. Similarly, these CoPE-FBA subnetworks can be used to directly explain the differences after secondary optimization, as we showed for the multimodal distributions of vertex cost.

In this research, we further demonstrated the use of CoPE-FBA for the *E. coli* iAF1260 GSSM of metabolism by determining P_L and P_C for each enumerated vertex for different growth conditions. We found Gaussian-shaped and multimodal distributions for P_L and P_C respectively. These results can be further used to deduce a hypothesis of the selection pressure if we know the flux distribution. If the objective of *E. coli* would be to minimize P_C (or P_J), we would not expect *E. coli* to exploit the unique optimal solution since the difference between this optimal solution and many suboptimal solutions is almost negligible. We do, however, expect *E. coli* to exploit the "cheap" reactions that cause the bi- or multimodal distribution of vertex cost. Interestingly, in aerobic conditions, our analysis predicted that all cheap pathways exploit ubiquinone-8 which is also the major electron acceptor in *E. coli* under these conditions (Unden and Bongaerts, 1997; Bekker et al., 2010). If the objective of *E. coli* would be to minimize P_L, many different flux vectors give rise to an optimal or near-optimal solution. The multitude of optimal solutions hinders the construction of a hypothesis about P_L from individual reactions. In future research we see as possible application of our method, finding the minimal flux distance between alternative optimal flux vectors in different conditions, to answer questions about how species can adapt to changing conditions.

4.4 Conclusions

We present a better understanding of the principles of the optimal solution space and we show how we can enumerate all non-decomposable flux pathways in this space. This paves the way to answer biological questions about the flexibility of organisms while growing at optimal states. This work, therefore, contributes to reaching a topological understanding of metabolic functionality in the optimum in terms of metabolic flux pathways. In the future, the development of graphical maps (as demonstrated in Chapter 3) can further simplify the analysis by allowing for straightforward visualization and inspection of these metabolic flux pathways.

4.5 Methods

Minkowski sum

In Chapter 2 we described the Minkowski sum (Eq. (2.21)) which provides a description of any $\boldsymbol{J^{opt}}$ in terms of vertices, rays, and linealities. Linealities do not exist after reversible-reaction splitting, which reduces the Minkowski sum to

$$\boldsymbol{J^{opt}} = \sum_{k=1}^{s} \alpha_k \boldsymbol{\varphi_k} + \sum_{k=1}^{t} \beta_k \boldsymbol{\phi_k}, \tag{4.3}$$

where the vectors $\boldsymbol{\varphi_k}$ and $\boldsymbol{\phi_k}$ represent the vertices and rays respectively. Additionally, s and t represent the upper boundaries of the sum functions indicating the number of vertices and rays respectively, and α_k and β_k represent the weighting coefficient that satisfy the following constraints: $\sum_{k=1}^{s} \alpha_k = 1$, $\alpha_k \geq 0$, and $\beta_k \geq 0$. In words, vertices can be summed by a convex combination and rays can be summed as a conical combination.

Each split reversible reaction fulfills all conditions for a ray, thus many more rays exist when we split each reversible reaction. Each of these additional rays is also an EFM and an extreme pathway which are considered irrelevant because they only reformulate reversibility (Papin et al., 2002; Klamt and Stelling, 2003).

Rank test

We successfully used the rank test (Klamt et al., 2005) to show that each enumerated vertex $\boldsymbol{v}$ is an instance of an (optimal-yield) EFM. First, we determined the zero indices of $\boldsymbol{v}$. Second, from the stoichiometric matrix $\boldsymbol{N}$ of the subnetwork we eliminated all columns with a zero index in $\boldsymbol{v}$ to create a submatrix $\boldsymbol{N_{nz}}$. Third, we used single value decomposition to determine the rank of $\boldsymbol{N_{nz}}$. And fourth, we used the rank–nullity theorem to determine its nullity,

$$\text{nullity } \boldsymbol{N_{nz}} = 1, \tag{4.4}$$

the dimension of the right nullspace which should be one if $\boldsymbol{v}$ is an EFM.

Secondary objectives

As a secondary objective, we used, in addition to pathway length (P_L) and pathway sum of absolute fluxes (P_J), also pathway cost (P_C)—a proxy for the minimization of the ATP utilization in protein synthesis—to reduce the size of the solution space,

$$P_L = |\{J_j : J_j \neq 0\}|, \tag{4.5}$$

$$P_J = \sum_{j=1}^{r} |J_j|, \tag{4.6}$$

$$P_C = \sum_{j=1}^{r} c_j |J_j|. \tag{4.7}$$

The P_L is identical to the number of flux carrying reactions, while the P_C is identical to the sum of absolute flux values multiplied with c_j, the protein cost for each individual reaction. This cost is the scaled length of the proteins that were associated to this reaction, which we used as a proxy for all costs. In other words, $c_j < 1$ when the associated protein length is less than average and vice versa. We set $c_j = 1$ when no information about associated proteins was available. Because multiple proteins can be associated to a particular reaction via AND and OR rules, different definitions of c_j were used: maximum, average, minimum, and equal. An AND rule corresponds to taking the sum of protein lengths, while an OR rule corresponds to taking the maximum, average, or minimum. Using equal cost is identical to minimizing the sum of absolute fluxes, a widely-used secondary objective. Taking the maximum, average, minimum, or equal definition of c_j did not effect the interpretation of our results.

Genome-scale stoichiometric models

The aerobic restricted version (maximum O_2 uptake was 18.5 mmol gDW^{-1} h^{-1}) of iAF1260 was obtained from the BiGG database (Schellenberger et al., 2010). Maximum glucose uptake was set to 12.77 mmol gDW^{-1} h^{-1} and we modified the bounds on the O_2 uptake reaction to create specific aerobic (no constraint on O_2 uptake) and anaerobic (exchange of O_2 set to zero) conditions. In all cases, the model required an ATP maintenance flux of 8.39 mmol gDW^{-1} h^{-1}. The model was edited and prepared for enumeration using PySCeS CBMPy (Olivier et al., 2005; Olivier, 2011). All models are provided as Supplementary Dataset S4.1. Optimization of secondary objectives (minimization of P_L, P_J, and P_C) was also done with PySCeS CBMPy. We used a mixed-integer linear program to minimize P_L and a linear program to minimize P_J and P_C.

Acknowledgments

We thank Mark Hanemaaijer (Vrije Universiteit Amsterdam), Christoph Kaleta (University of Jena), Jan van Klinken (LUMC), and Arne Reimers (CWI) for useful discussions. Additionally, we thank SURFsara (www.surfsara.nl) for the support in using the Lisa Compute Cluster.

Supplemental Material

The following materials are available at http://persistent-identifier.org/?identifier=urn:nbn:nl:ui:18-23539 and in the online version of this article on which this chapter is based on.

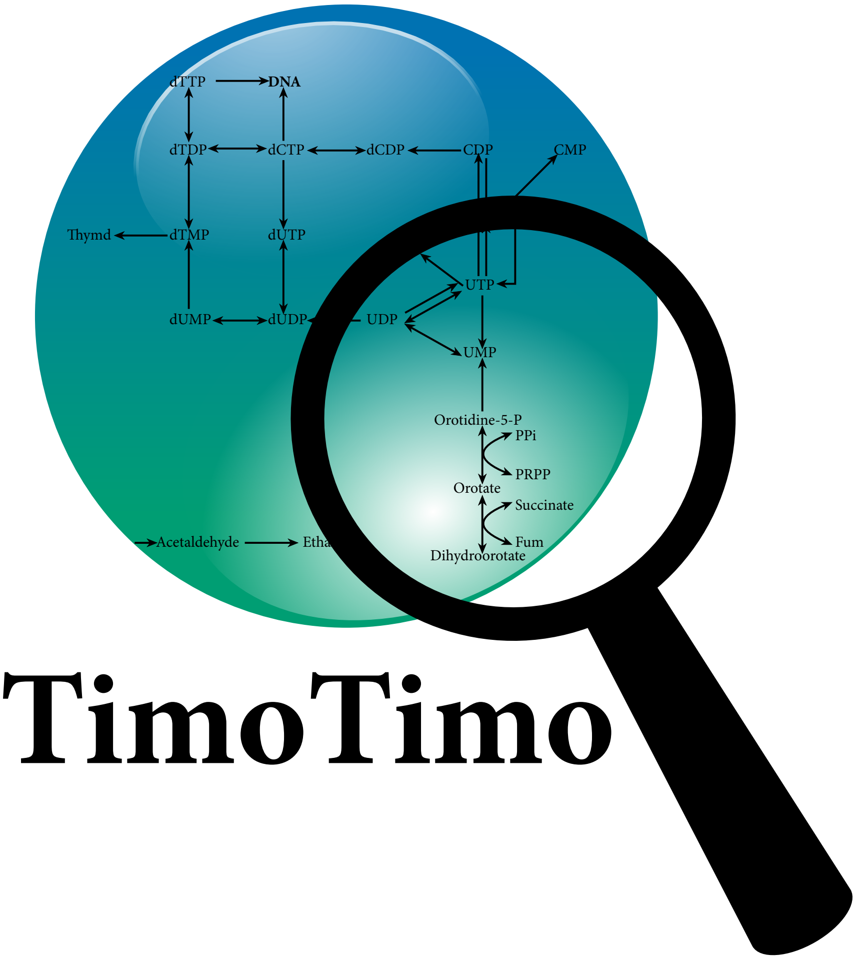
Supplemental Dataset S4.1. A zip-file containing SBML files encoding the metabolic networks used in this chapter.

Supplemental Fig. S4.1.–S4.7. Fig. S4.1 shows examples of rays in the *E. coli* iAF1260 genome-scale metabolic model. Fig. S4.2 shows subnetworks with correlated flux variability. Fig. S4.3 shows that vertices correspond to optimal-yield EFMs if they are restricted by a single limiting flux constraint. Fig. S4.4 shows that minimization of pathway length yields one or more vertices with reversible-reaction splitting. Fig. S4.5 illustrates the effect of reversible-reaction splitting. Fig. S4.6 shows that multimodal vertex cost distributions are consistent for different definitions of cost. Fig. S4.7 shows the vertex distribution without reversible-reaction splitting. Fig. S4.8 visualizes the *E. coli* iAF1260 genome-scale metabolic model cost-explaining module in anaerobic growth conditions.

Supplemental Table S4.1.–S4.3. Different additional tables.

This chapter is based on the publication:

Maarleveld T.R., Wortel M.T., Olivier B.G. Teusink B., and Bruggeman F.J. (2015) *"Interplay between constraints, objectives and optimality for genome-scale stoichiometric models of metabolism"*, **PLoS Computational Biology**, 11, 4:e1004166.



The TimoTimo (CoPE-FBA 2.0) quickly enumerates the optimal solution space in terms of vertices and rays.

The TimoTimo: a revolutionary "navigation system" for analyzing metabolic highways

5

Abstract

Organisms depend on huge networks of molecular reactions for environmental sensing, information integration, gene expression, and metabolism. The metabolic capacities of a cell can be studied with genome-scale stoichiometric models (GSSMs) of its metabolism. In the preceding chapter we presented a better understanding of the optimal solution space of these GSSMs and we showed how to enumerate all non-decomposable flux pathways in the optimal solution space. However, we did not mention that, for GSSMs, the enumeration of all non-decomposable pathways in this space was considered to be not feasible. To enumerate each non-decomposable flux pathway in the optimum, we used the TimoTimo—the tool we present in this chapter that allows for the fast enumeration of all distinct non-decomposable pathways in the optimal solution space. In this chapter, we show its description, implementation, and performance and end with an illustrative example where we used the iTM686 GSSM of *Synechocystis*' metabolism. We believe that our tool opens up new opportunities to study the flexibility of organisms while growing at optimal states.

5.1 Introduction

The metabolic capacities of a cell can be studied with genome-scale stoichiometric models (GSSMs) of metabolism. Comprehensive insight into the entire metabolic capacity of a cell is obtained by computation of all feasible metabolic pathways that agree with stoichiometric, thermodynamic, and steady-state constraints, such as elementary flux modes or extreme pathways. When additional constraints and optimality conditions are imposed, the set of feasible metabolic pathways reduces considerably. This set of optimal metabolic pathways defines the optimal solution space. Flux Balance Analysis (FBA) is a popular method to determine an optimal flux distribution (Lee et al., 2006; Orth et al., 2010), however, it does not provide any insight into the optimal solution space. Traditionally, Flux Variability Analysis (FVA) was the method to get insight into the optimal solution space by quantifying the range of flux values that a single reaction can take in the optimal solution space (Mahadevan and Schilling, 2003).

The computational method CoPE-FBA (Comprehensive Polyhedron Enumeration FBA) was recently developed to characterize this optimal solution space in terms of vertices, rays, and linealities (Kelk et al., 2012), whereas this method suffers from extensive running times: days to weeks for most GSSMs of metabolism. In the preceding chapter, we presented a better understanding of the principles of the optimal solution space of these GSSMs. We also further developed CoPE-FBA by using reversible-reacting splitting to enumerate all non-decomposable flux pathways rather than the minimal generating set in the optimum. Characterizing the optimal solution space before reversible-reaction splitting is hard, but characterizing this space after reversible-reaction splitting is very hard to impossible for the majority of GSSMs of metabolism. In fact, we were not able to generate the results presented in the preceding chapter with CoPE-FBA as it was presented by Kelk et al. This means that there was a need for a more efficient method to characterize this optimal solution space. Besides the fact that CoPE-FBA suffered from extensive running times, it is also platform dependent and requires the use of four different programming languages (Java, Gawk, Python, and Bash).

We present the TimoTimo, a user-friendly computational method developed in Python that efficiently characterizes the optimal solution space in terms of optimal pathways. We call this method the TimoTimo (hereafter CoPE-FBA 2.0), because it determines the optimal metabolic routes (metabolic highways) just as the TomTom (the global leader in navigation, traffic and map products) determines the optimal route on road maps. We show that CoPE-FBA 2.0 allows us to characterize the optimal solution space in the order of minutes for most (bacterial) GSSMs on just an ordinary computer. We further use the GSSM iTM686 of *Synechocystis* metabolism (presented in Chapter 3) as an example illustrating the use of secondary objectives to collapse the optimal solution space.

5.2 Tool Description

CoPE-FBA 2.0 characterizes the optimal solution space (see Eq. (2.16)) which allows us to, as explained in the previous chapter, characterize in terms of linealities, rays, and vertices. In short, linealities are reversible cycles or input-output pathways, rays are irreversible cycles or input-output pathways, and vertices are the corner points of the FBA polyhedron. We also showed that vertices originate from combinations of alternative flux distributions in CoPE-FBA subnetworks, quickly leading to a combinatorial explosion (Kelk et al., 2012). In the optimum, these subnetworks consist of reactions with correlated flux variability and have a fixed net input-output stoichiometry of reactants and products. Mathematically, this can be written as

$$\boldsymbol{N_A J_A} = \boldsymbol{d} \neq \mathbf{0}, \tag{5.1}$$

where $\boldsymbol{A}$ is a vector of reactions that form the subnetwork, $\boldsymbol{N_A}$ and $\boldsymbol{J_A}$ are the stoichiometric matrix and the flux vector of the subnetwork, and $\boldsymbol{d}$ is the fixed input-output relationship of the subnetwork (Müller and Bockmayr, 2014).

These CoPE-FBA subnetworks are specific F-modules that are defined as sets of reactions (i.e. subnetworks) that yield always the same net input-output relationship. The F-modules are mathematically defined as

$$\boldsymbol{N_A J_A} = \boldsymbol{d}. \tag{5.2}$$

In other words, two types of F-modules can be distinguished:

- F-modules essential for optimality (i.e. $\boldsymbol{d} \neq \boldsymbol{0}$); and
- F-modules not essential for optimality (i.e. $\boldsymbol{d} = \boldsymbol{0}$).

In the flux space without futile cycles, F-modules essential for optimality are identical to CoPE-FBA subnetworks while F-modules not essential for optimality do not exist. Alternatively, in the flux space with futile cycles, F-modules not essential for optimality are rays or linealities not connected to optimal flux pathways. Then, F-modules essential for optimality can be a CoPE-FBA subnetwork, a ray or lineality connected to optimal flux pathways, or any combination thereof.

The design of CoPE-FBA 2.0 is based on its predecessor CoPE-FBA, thus we start the description of our computational method by describing the basic principles of its predecessor, CoPE-FBA, developed by Kelk et al. (2012). CoPE-FBA first enumerates the linealities, rays, and vertices and then the CoPE-FBA subnetworks in the flux space without futile cycles (Fig. 5.1); the space where these vertices are defined. We propose to do the reverse: use a divide-and-conquer approach to first determine the subnetworks and then determine the vertices (Fig. 5.1). The divide-and-conquer approach breaks down the problem to many (simple) subproblems where each subproblem should be simple enough to solve within seconds to minutes. To determine all non-decomposable flux vectors in the optimum, we need to determine the subnetworks within the flux space with futile cycles. In Fig. 5.2 we show how enumerating the flux space of our toy model without futile cycles yields four vertices (see also Fig. 2.4) while enumerating the flux space with futile cycles yields twenty-four vertices. These twenty-four vertices originate from four F-modules (Fig. 5.2C) that represent a lineality connected to optimal flux pathways (orange), a CoPE-FBA subnetwork (red), a combination of a CoPE-FBA subnetwork and a lineality (purple), and a ray connected to optimal flux pathways (yellow), respectively.

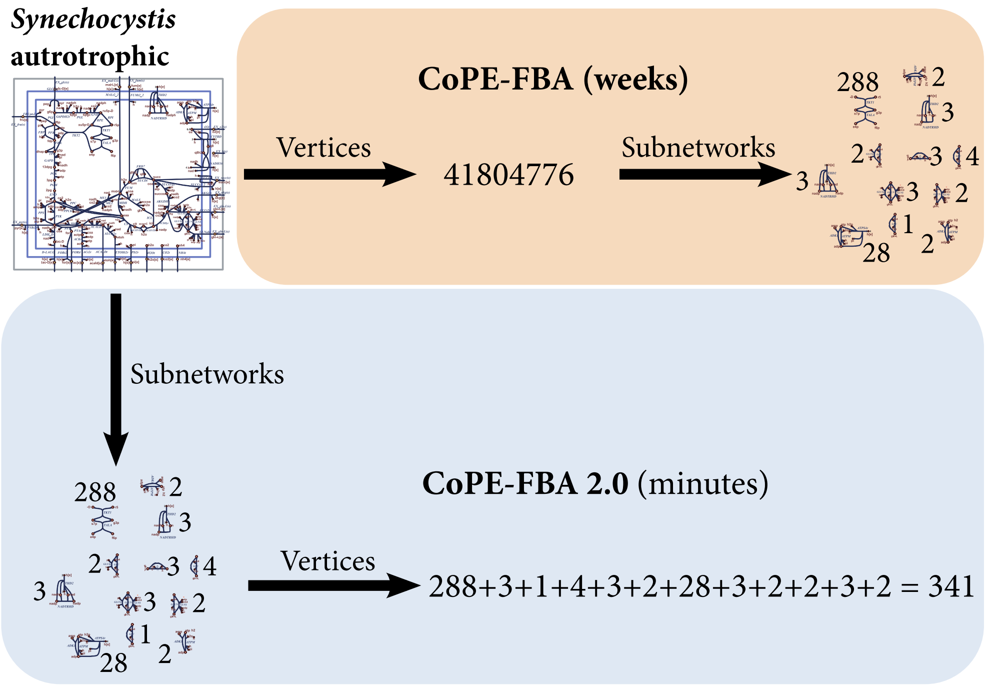


Figure 5.1. Conceptual representation of CoPE-FBA 2.0 and its predecessor. *We illustrate both approaches for the iTM686 GSSM of Synechocystis' metabolism. Determining the vertices for all subnetworks alone results in enumerating* 8.2×10^{-4} *% of the vertices at the network level. This results in a significant speed advantage (weeks vs. minutes for this particular example).*

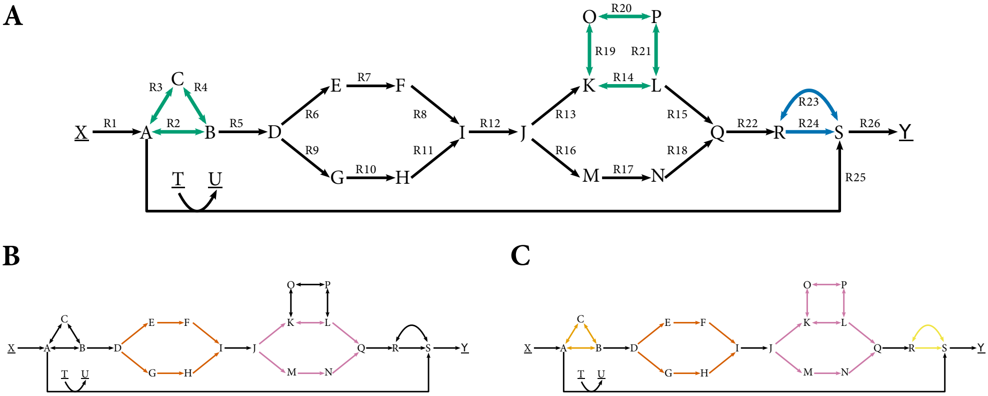


Figure 5.2. Subnetworks in the flux space with and without futile cycles. *(A) futile cycles—linealities (green) and a ray (blue)—in the toy network (before reversible reaction splitting). (B) in the flux space without futile cycles there exist two subnetworks that give rise to 4 vertices (2 × 2). (C) in the flux space with futile cycles there exist four subnetworks that give rise to 24 vertices (2 × 2 × 3 × 2).*

5.3 Implementation

The CoPE-FBA 2.0 pipeline we present here and all data files used during these study are available for download from http://memesa-tools.sf.net. Our pipeline is written in the Python language. We implemented CoPE-FBA 2.0 in Python because it is a great scripting language, allows for the use of numerous scientific libraries written in compiled languages, and can be freely installed and used on a wide range of operating systems. The CoPE-FBA 2.0 pipeline consists of three processes: model preparation, model enumeration, and model analysis (Fig. 5.3).

SBML
Model Preparation
Model Enumeration
Model Analysis
output

Figure 5.3. Schematic description of the main processes of the CoPE-FBA 2.0 pipeline.

Each of these processes is executed by different elements (Fig. 5.4):

(i) **Model Preparation**

a) **Model Setup.** This module loads the metabolic network written in the Systems Biology Markup Language (SBML) (Hucka et al., 2003) and creates the required directory structure.

b) **Scan for Reaction Duplicates.** We use CBMPy (Olivier, 2011) to scan for reactions with an exact matching stoichiometry. All redundant pairs that are not biologically relevant should be removed.

c) **Get F-modules.** We use (floating-point precision) FVA to determine the set of variable reactions and subsequently a Python implementation of FluxModules (Müller et al., 2014) to determine the F-modules.

d) **Convert SBML to H-format.** For mathematical stability we first replace "fake" infinity bounds (± 1000 or ± 999999) with $\pm\infty$. Then we use the Polyhedra format (H-format) from the CDD manual (Fukuda, 1997) to describe the convex polyhedron in exact arithmetic. These files use the extension ".ine" (**ine**qualities).

e) **Perform exact FBA.** We use exact arithmetic (QSopt_EX) to perform an FBA—this is required because we need the exact input-output relationship of every F-module. The FBA output is used in the next step to fix $\boldsymbol{d}$ of every F-module and in the final step to set the fluxes in the fixed part of the network.

f) **Model to Modules.** We combine the output of the preceding steps to construct stoichiometric models for each F-module with exact arithmetic. We added input and output reactions to fix $\boldsymbol{d}$ of each type I F-module in the optimal solution space. Dummy species were added both the input and output reaction to guarantee use of both reactions. We split reversible reactions to ensure that we characterize all non-decomposable flux pathways in the optimum. We use again the H-format to describe the modules.

  g) **Model Reduction.** We reduce each model by removing fixed reactions. We ensure steady state of each metabolite by adding the stoichiometric coefficients of the removed reactions to "$\boldsymbol{b}$" (i.e. $\boldsymbol{NJ} = \boldsymbol{b} \neq \boldsymbol{0}$).

(ii) **Model Enumeration.** Use the F-modules with split reversible reactions as input.

  a) **Enumerate vertices and rays.** Polco is used to enumerate the set of vertices and rays.
  b) **Check Scaling.** The vertices output by Polco are scaled when necessary.
  c) **Check Vertices.** As a final sanity test, the first ten vertices are checked to see if they do actually occur in the optimal solution space (the default argument is set to ten, checking a different number if also possible).

(iii) **Model Analysis**

  a) **Translate.** We translate the Polco output to a HDF5 data format that is specifically designed to store and organize big numerical data sets.
  b) **Modules to Model.** We reconstruct network vertices by reintegrating the F-modules fluxes with the fixed fluxes that were removed earlier in the pipeline.
  c) **Vertex Features.** We can easily determine vertex features such as length (number of flux carrying reactions), sum of absolute fluxes, and cost (sum of absolute fluxes multiplied with a proxy for cost).

We use CBMPy (Olivier, 2011) for various steps in the described pipeline (e.g. scan for reaction duplicates, convert SBML to H-format, and FVA). We use QSopt_EX (Applegate et al., 2007) as a rational FBA solver and Polco (Terzer, 2009) to enumerate the vertices and rays of the polyhedron respectively, which are both freely available.

5.4 Results and Discussion

Benchmarking CoPE-FBA 2.0

We selected metabolic GSSMs of different micro-organisms to benchmark CoPE-FBA 2.0 against its predecessor. The first step was to compare the output of both methods. Or in other words, do we get an identical characterization of the optimal solution space? Table 5.1 gives an overview of the characterization of the optimal solution space with reversible-reactions splitting. CoPE-FBA was not able to characterize the optimal solution space with

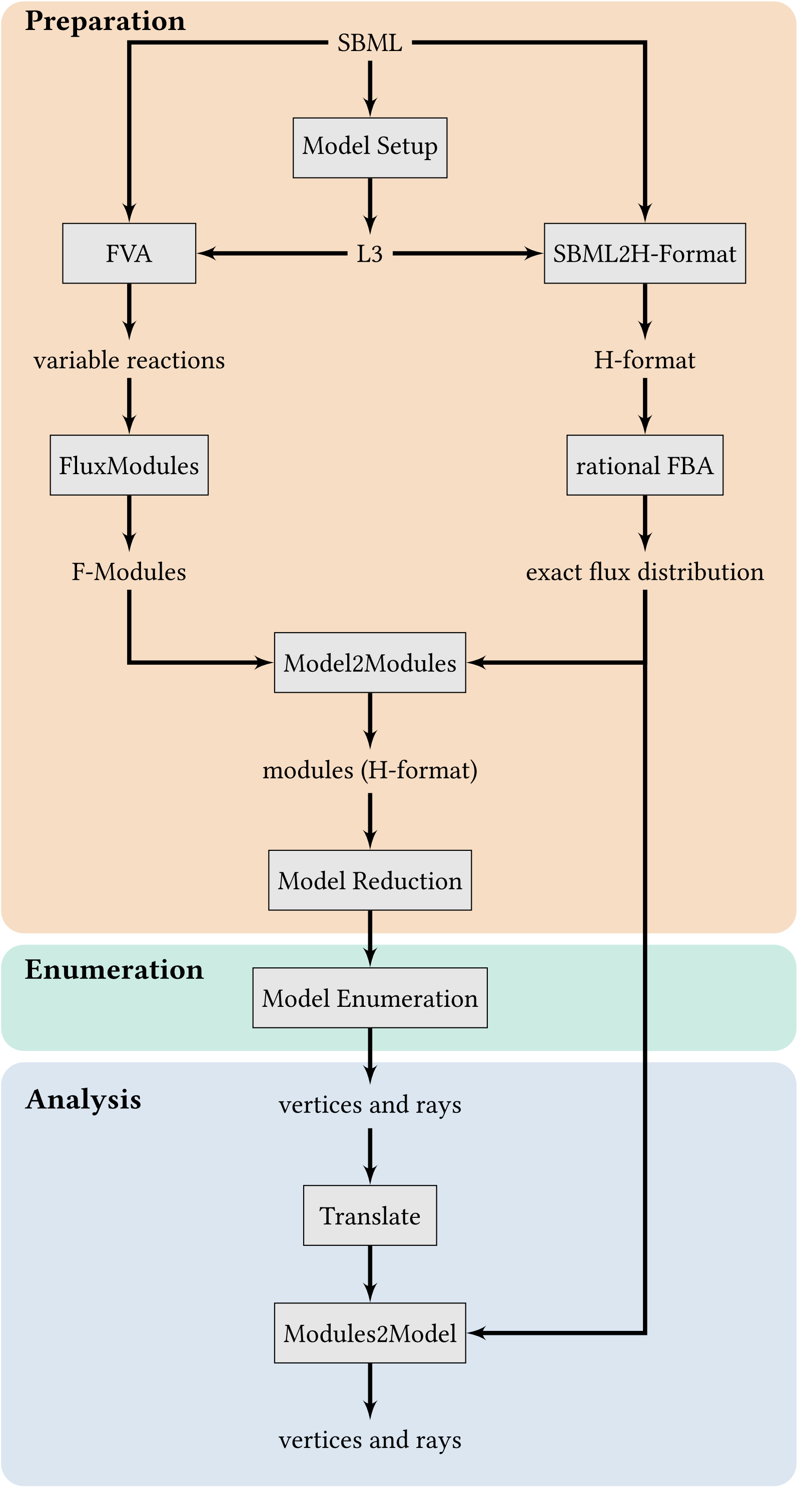


Figure 5.4. Detailed schematic description of the CoPE-FBA 2.0 pipeline. *The model preparation (red), enumeration (green), and analysis (blue) and their interactions are shown in detail. Text in grey blocks indicate scripts, and text without grey blocks indicate input or output for these scripts respectively.*

Organism (model)	Reactions	Source	Vertices	Rays	Subnetworks
S. PC6803 (iTM686)*	912	glycogen	1.4×10^9	242	12
S. PC6803 (iTM686)*	912	light	41803776	242	12
E. coli (iAF1260)*	2374	glucose	40310784	604	4 (8)
E. coli (iAF1260,ox)*	2374	glucose	120932352	604	9 (13)
E. coli (iAF1260,noox)*	2374	glucose	1492992	602	8 (12)
L. Lactis	735	glucose	1218240	511	8
S. thermophilus	555	methanol	280	186	6 (7)
M. tuberculosis (iNJ661)*	1020	glycerol	705×10^9	313	14 (16)
M. barkeri (iAF692)	688	methanol	104.832	238	6
E. coli (iJR904)	1066	malate	6912	264	9 (10)
E. coli (iJR904)	1066	glucose	138240	264	7 (8)
E. coli (iJR904)	1066	fumarate	7680	264	10 (11)
E. coli (iJR904)	1066	tryptophan	13824	264	6 (7)

Table 5.1. Overview of the optimal solution space characterization after reversible-reaction splitting for different GSSMs for growth on different sources. *The number without and within parenthesis in the subnetwork column represent the number of CoPE-FBA subnetworks and the number of F-modules respectively. Unpublished L. lactis and S. thermophilus reconstructions were provided by Prof. B. Teusink (Vrije Universiteit Amsterdam).*
** Enumeration was not done successfully with CoPE-FBA.*

reversible-reaction splitting for the GSSMs denoted with a * in Table 5.1. For these GSSMs, we observed no difference in the characterization of CoPE-FBA 2.0 and its predecessor without reversible-reaction splitting (not shown). If CoPE-FBA was able to characterize the optimal solution space with reversible-reaction splitting, the characterization was in agreement with the results obtained with CoPE-FBA 2.0.

The second step was to compare the performance of both methods. That is, how much faster can we now characterize the optimal solution space? An overview of the running time of both methods is given in Table 5.2. We distinguished between preparation and enumeration time (analysis time is ignored but takes significantly longer with CoPE-FBA due to the "translate" step). While preparation times between both methods were similar, the enumeration time of CoPE-FBA 2.0 is many orders of magnitude faster. More specifically, CoPE-FBA 2.0 typically enumerated the optimal solution space within approximately a minute while its predecessor required several days to enumerate each of these optimal solution spaces.

To illustrate this, the 41804776 vertices enumerated for the iTM686 GSSM of *Synechocystis* metabolism (presented in Chapter 3) originate from twelve subnetworks with respectively 288, 3, 1, 4, 3, 2, 28, 3, 2, 2, 3, 2 vertices (as previously shown in Fig. 5.1). This means that while we determined in total only 341 vertices (the sum), we actually enumerated 41804776 vertices (the multiplication) within one minute.

Organism (model)	Source	Running time (s) CoPE-FBA Prep.	 Enum.	Running time (s) CoPE-FBA 2.0 Prep.	 Enum.
S. PC6803 (iTM686)	glycogen	103	FAILED	112	95
S. PC6803 (iTM686)	light	102	FAILED	99	22
E. coli (iAF1260)	glucose	688	FAILED	524	147600
E. coli (iAF1260,ox)	glucose	680	FAILED	527	36
E. coli (iAF1260,noox)	glucose	682	FAILED	520	27
L. Lactis	glucose	87	475420	67	15
S. thermophilus	lactose	49	345362	45	11
M. tuberculosis (iNJ661)	glycerol	133	FAILED	126	60
M. barkeri (iAF692)	methanol	71	265760	58	11
E. coli (iJR904)	malate	150	324793	122	15
E. coli (iJR904)	glucose	149	575983	126	16
E. coli (iJR904)	fumarate	149	584690	125	16
E. coli (iJR904)	tryptophan	156	695883	132	14

Table 5.2. Comparison of running times for CoPE-FBA developed by Kelk et al. (2012) and our version, CoPE-FBA 2.0. *Running times (seconds) represent preparation and enumeration time. For a fair comparison all simulations were done on the Lisa Compute Cluster. Unpublished L. lactis and S. thermophilus reconstructions were by Prof. B. Teusink (Vrije Universiteit Amsterdam).*

We want to stress that there can be exceptions where CoPE-FBA 2.0 cannot characterize the optimal solution space in the order of minutes. We found one particular example, the iAF1260 GSSM of *E. coli* metabolism with a specific growth condition—growing on glucose with a restriction on the oxygen uptake. This particular enumeration took about 41 hours, but note that we were not able to enumerate the optimal solution space with CoPE-FBA at all (Table 5.2). Our divide-and-conquer approach failed here to break down the problem in many simple-to-solve subproblems, simply because they do not exist. In this specific example, we obtained one subproblem that was still very difficult to solve: A subnetwork that consists of many vertices (e.g. in the order of a million or more). This subproblem cannot be further simplified, which means that we found one enormous subnetwork that explains most of the variability.

A real life example: Exploring *Synechocystis'* metabolic potential in the optimum

We have now demonstrated that CoPE-FBA 2.0 outperforms its predecessor. This opens up new opportunities to explore the metabolic potential in the optimum. We illustrate this here by analyzing the optimal solution space of iTM686 GSSM of *Synechocystis* metabolism (presented in Chapter 3) with CoPE-FBA 2.0; a type of analysis no other method to date can perform. We constructed two different FBA models of iTM686 that depict autotrophic and heterotrophic growth. We set maximization of the specific growth rate as the objective function. The biomass composition of *Synechocystis* is different under these growth conditions. As discussed in Chapter 3, the iTM686 GSSM produces glycogen during autotrophic growth which can be utilized during heterotrophic growth.

With CoPE-FBA 2.0 we were able to enumerate the vertices in both growth conditions (see also Table 5.1). We used the enumerated vertices for subsequent analysis where we studied the distributions of objective values of a secondary optimization over the vertices obtained in the first optimization, optimization of specific growth rate. For each vertex, we determined the pathway length (P_L) and the pathway sum of absolute fluxes (P_J) (see Methods of Chapter 4) of which the results are shown in Fig. 5.5. We found again Gaussian-shaped and bimodal distributions for P_L and P_J respectively. The origin of these distributions was discussed in detail in the preceding chapter. In short, they are the result of flux distributions through the subnetworks. Many different subnetworks contributed to the difference in P_L and within these subnetworks we found alternative flux distributions with different lengths. In contrast, one particular subnetwork explains the main difference in P_J. Within this particular subnetwork, we found a relatively cheap and a relatively expensive flux distribution.

In autotrophic growth conditions, the sum of absolute fluxes is mainly explained by using a different strategy for bicarbonate (HCO_3^-) uptake. We analyzed a light limiting autotrophic growth condition, which means that there was an excess amount of HCO_3^- available. This means that, to maximize the specific growth rate, cheap pathways used the minimal HCO_3^- uptake required and expensive pathways used the maximal HCO_3^- uptake possible (Fig. 5.6). The difference between both strategies is 11%.

5.5 Conclusions

We presented a fast and user-friendly method, CoPE-FBA 2.0, for the characterization of the optimal solution space of GSSMs of metabolism. This method exploits a divide-and-conquer approach which allows us to characterize this space typically in the order of minutes, while its predecessor requires for similar models days to weeks to complete. We expect that our tool will aid in research about optimality principles in metabolism and that it will fulfill the premise of CoPE-FBA by providing a simplified biological understanding of the optimal solution space of metabolic network models.

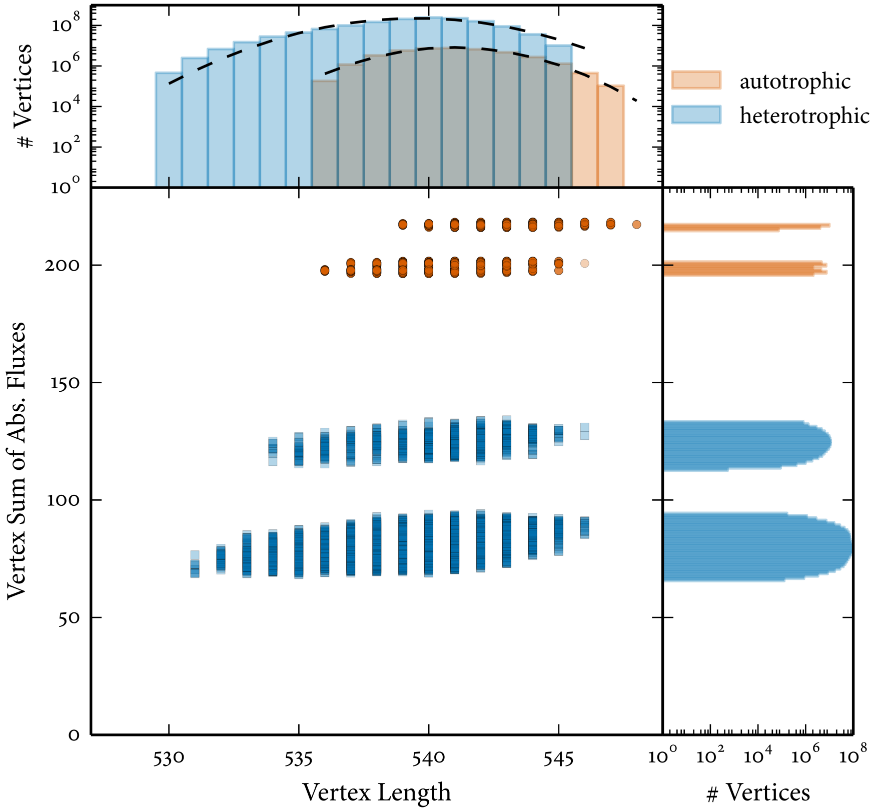


Figure 5.5. ***Synechocystis*** **vertex sum of absolute fluxes follows a bimodal distribution.** *For two growth conditions—autotrophic (red, circle) and heterotrophic (blue, square)—we analyzed the vertex sum of absolute fluxes* (P_J) *and vertex length* (P_L) *of each vertex. Each dot in the main panel represents a vertex with a specific sum of absolute fluxes and length. Our results indicate that (also) for Synechocystis the vertex length follows approximately a Gaussian-shaped distribution (dashed lines are Gaussian distributions with sample mean and sample standard deviation). Vertex cost follows a bimodal distribution; vertices are clustered in distinct groups with a specific cost. Due to file size limitations we only show a subset (10000) of vertices for all conditions in the scatter plot.*

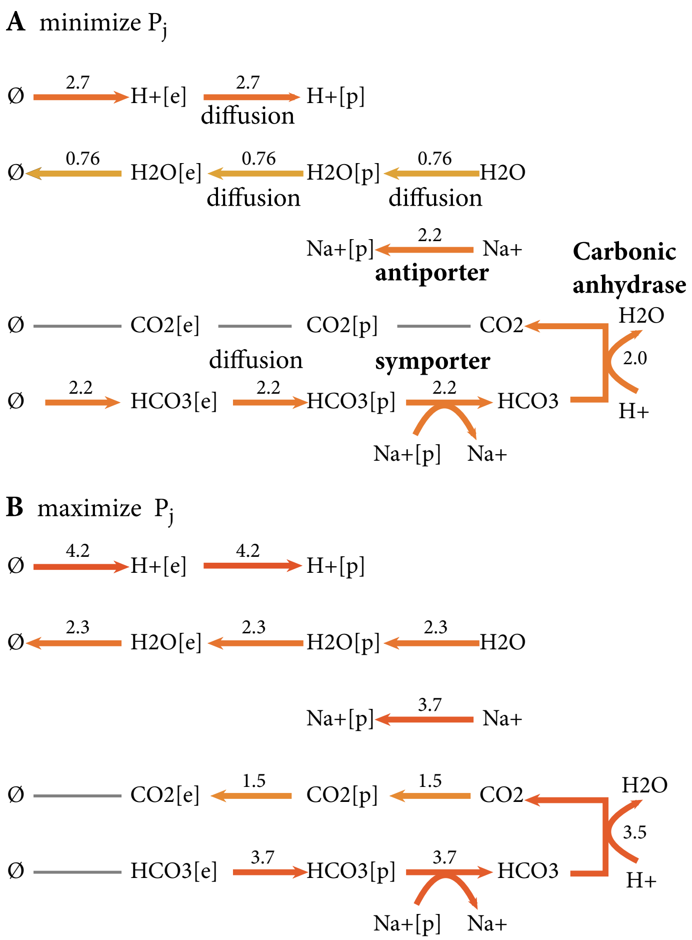


Figure 5.6. Different bicarbonate uptake strategies of *Synechocystis*. *Predicted in silico fluxes for a minimal (panel A) and maximal (panel B)* HCO_3^- *uptake. We highlighted which reactions are dependent on enzymes and which are dependent on diffusion.*

5.6 Supplemental Material

Dataset S1. A description of the stoichiometric models in SBML format can be found at http://persistent-identifier.org/?identifier=urn:nbn:nl:ui:18-23540.

Acknowledgments

Additionally, we thank Arne Reimers (CWI) for providing a Python implementation of FluxModules and SURFsara (www.surfsara.nl) for the support in using the Lisa Compute Cluster.

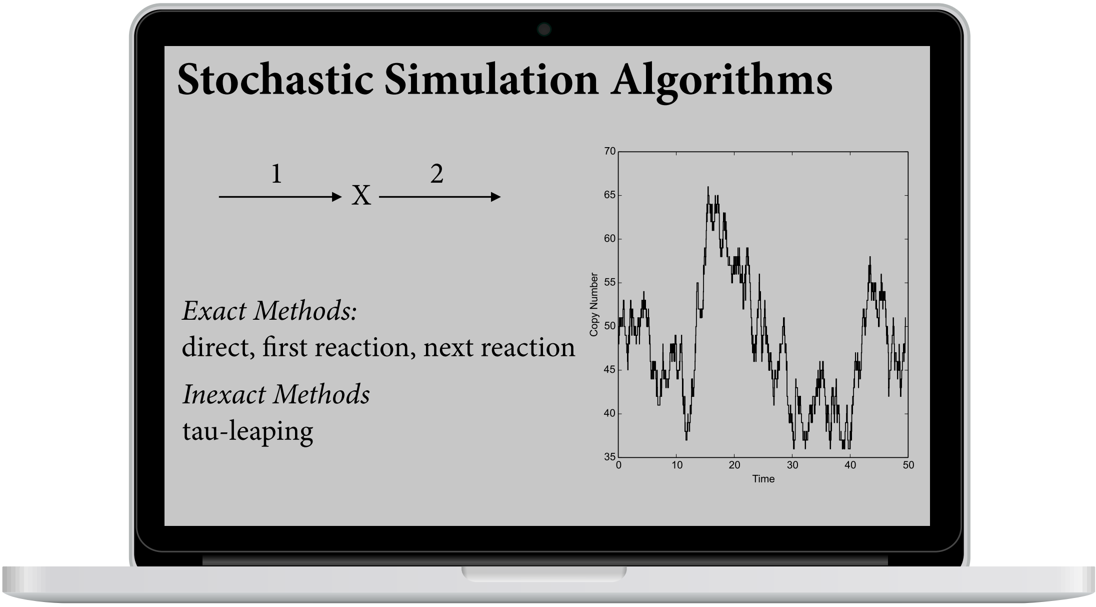
Part III

Fluctuations in Biochemical Models

Discrete Stochastic Modeling & Simulation

Section “Example 4: Stochastic simulation predicts active-state locking” is based on the publication:

Wei K., Moinat M., **Maarleveld T.R.**, Bruggeman F.J. (2014) *“Stochastic simulation of prokaryotic two-component signalling indicates stochasticity-induced active-state locking and growth-rate dependent bistability”*, **Molecular BioSystems**, 10, 9:2338-46.



Stochastic Simulation Algorithms

$$\xrightarrow{1} X \xrightarrow{2}$$

Exact Methods:
direct, first reaction, next reaction

Inexact Methods
tau-leaping

Copy Number
70
65
60
55
50
45
40
35
0
10
20
30
40
50
Time

Stochastic Simulation Algorithms: A popular discrete stochastic approach to study the (emergent) properties of biochemical models. In 1977, Gillespie published both the direct and first-reaction method which are still used today. More efficient algorithms such as the next-reaction method have been developed to improve the performance of stochastic simulation algorithms.

A toolkit for analyzing fluctuations in biochemical models: stochastic simulation algorithms

6

Abstract

Biological systems are heterogeneous by nature which can be to a large extent explained by the stochasticity of biochemical reactions that creates non-genetic cell-to-cell variability. While traditional chemical kinetics fail to capture this stochasticity, there exist stochastic modeling and simulation approaches that do capture this stochasticity. Various stochastic approaches exist, but we focus here on discrete stochastic approaches (the stochastic simulation algorithms) which are arguably the most popular these days. We start with the chemical master equation and explain that stochastic simulation algorithms generate time trajectories that are in accordance with the chemical master equation. In the remainder of this chapter, we provide an overview of both exact and inexact stochastic simulation algorithms and highlight their importance in systems biology with various simple and illustrative examples.

Introduction

DETERMINISTIC models fail to capture the discreteness and stochastic effects that occur when molecules are present at low copy numbers. Molecular dynamics is the most accurate method to describe the time evolution of a biological system. For each time point molecular dynamics determines the position and velocity of each molecular species making molecular dynamics simulations very expensive computationally. Assuming that the system is well-stirred simplifies the problem incredibly. This means that we can ignore the simulation of non-reactive molecular collisions; the collisions that require the majority of the simulation time in a molecular dynamics simulation.

The well-stirred assumption allows us to focus only on the reactive molecular collisions, thus molecular positions and diffusion can be ignored. Assuming, additionally, a constant system volume and temperature allows us to use the chemical master equation (CME)—

Chemical master equation

The equation that determines the probability that the system is in a specific state at a given (future) time.

—to describe the stochastic process which gives an exact description of the Markov process through time (McQuarrie, 1967). A Markov process is defined as:

Markov process

A random process whose future probabilities are determined by its current state.

Or in other words, in a Markov process the next state depends only on the current state, not on the past history of the states. Markov processes are extensively used for a broad set of applications. In economics and finance, these Markov processes are, for instance, used for prediction of future stock prices (e.g. with geometric Brownian motion as illustrated in Fig. 1.1C).

A single state is in this formalism represented by a particular combination of the number of molecules per cell. We consider a system of m species and r reactions. From the stochastic perspective the state is a vector of integers that each represents the (copy) number of a specific molecule per cell. The general description of the CME is given by

$$\frac{d\boldsymbol{P(t)}}{dt} = \boldsymbol{SP(t)} \tag{6.1}$$

and describes the time evolution of a stochastic system with biochemical reactions. Here, $\boldsymbol{S}$ is defined as the transition matrix and each element $P_n(t)$ of $\boldsymbol{P(t)}$ equals the probability of the system to be in state n at time t. A more specific description of the CME is given by

$$\begin{aligned} \frac{dP(\boldsymbol{x}, t|\boldsymbol{x_0}, t_0)}{dt} = &\sum_{j=1}^{r} a_j(\boldsymbol{x} - \boldsymbol{v}_j) \cdot P(\boldsymbol{x} - \boldsymbol{v}_j, t|\boldsymbol{x_0}, t_0) - \\ &\sum_{j=1}^{r} a_j(\boldsymbol{x}) \cdot P(\boldsymbol{x}, t|\boldsymbol{x_0}, t_0), \end{aligned} \tag{6.2}$$

where $\boldsymbol{x}$ is the state vector which contains the number of molecules (denoted later by n_i for species index i) of each species in time, $P(\boldsymbol{x}, t|\boldsymbol{x_0}, t_0)$ is the probability to observe the system in state $\boldsymbol{x}$ at time t given the initial state $\boldsymbol{x_0}$ at time t_0, $\boldsymbol{v_j}$ is the state-change vector of reaction index j (vector with stoichiometric coefficients), and $a_j(\boldsymbol{x})$ is the propensity function of reaction index j, that is,

$$\begin{aligned} a_j(\boldsymbol{x})dt = &\text{ the probability that reaction } j \text{ occurs once in } V \text{ in the time interval } [t, t+dt) \\ &\text{ given that the system is currently in state } \boldsymbol{x}. \end{aligned} \tag{6.3}$$

We use a simple and widely used model, the birth-death process (Fig. 6.1A) in one of its simplest forms (constant birth rate), to illustrate usage of the CME. This birth-death process is an example of a (continuous-time) Markov process. The birth-death process consists of two state transitions (Fig. 6.1B): *births*, which increase the state variable by one and *deaths*, which decrease the state variable by one.

We exploit the birth-death process to study the time evolution of the number of mRNA molecules (n_{mRNA}) in a single cell. mRNA molecules are synthesized and degraded. We therefore call these state transitions the synthesis and degradation of mRNA respectively and refer to this model as the synthesis-degradation process. We start by determining the propensity functions, the state-change vectors, and the state vector for the synthesis-degradation process. The propensity functions are given by

$$\begin{aligned} a_1(\boldsymbol{x}) &= k_{syn}, \\ a_2(\boldsymbol{x}) &= k_{deg} \cdot n_{mRNA}, \end{aligned} \tag{6.4}$$

where k_{syn} and k_{deg} denote the synthesis and degradation rate constant respectively. These rate constants are volume (V) independent, whilst rate constants of bimolecular reactions are inversely proportional to V; the search time for two reactant molecules depends on V (Berg and von Hippel, 1985).

The state-change vectors for our synthesis-death process are given by

$$\begin{aligned} \boldsymbol{v_1} &= (1), \\ \boldsymbol{v_2} &= (-1), \end{aligned} \tag{6.5}$$

of which the entries correspond to the stoichiometric coefficients of the molecules in the same order as they occur in the state vector $\boldsymbol{x}$,

$$\boldsymbol{x} = (n_{mRNA}). \tag{6.6}$$

Substituting Eqs. (6.4)–(6.6) into Eq. (6.2) gives the CME for our synthesis-degradation model:

$$\begin{aligned} \frac{dP(n_{mRNA}, t|\boldsymbol{x_0}, t_0)}{dt} =& k_{syn} \cdot P(n_{mRNA} - 1, t|\boldsymbol{x_0}, t_0) + \\ & k_{deg}(n_{mRNA} + 1) \cdot P(n_{mRNA} + 1, t|\boldsymbol{x_0}, t_0) - \\ & (k_{syn} + k_{deg} n_{mRNA}) \cdot P(n_{mRNA}, t|\boldsymbol{x_0}, t_0). \end{aligned} \tag{6.7}$$

Most CMEs cannot be solved analytically and this is where stochastic simulation algorithms (SSAs) come in.

Stochastic simulation algorithm

A Monte Carlo method to numerically generate time realizations which are in agreement with the CME.

This means that the SSA numerically simulates the Markov process which the CME describes. The CME and SSA are derived from the same principle: the propensity function. Gillespie (the developer of the first SSAs) calls the propensity function the fundamental premise of stochastic chemical kinetics, because everything else in the theory follows from it via the laws of probability (Gillespie, 2007). The important advantage of SSAs is that they are simple to implement, especially compared to procedures used to numerically solve deterministic models. Before we discuss these SSAs in detail, we use our synthesis-degradation process to illustrate the SSA and compare the SSA to a macroscopic description of the synthesis-degradation process.

If we ignore the fluctuations in biochemical models, we can write down the macroscopic description as

$$\frac{dmRNA(t)}{dt} = v_{syn} - v_{deg}, \tag{6.8}$$

which represents a continuous, deterministic process. The rate equations of the synthesis and degradation process are given by v_{syn} and v_{deg} respectively. The macroscopic description is usually done in concentrations; $mRNA(t)$ is the mRNA concentration at time t. For convenience we set the cell volume to 1 (cell volume of $E.coli$ is about 1 fl (Milo et al., 2010)) such that $n_{mRNA}(t)$ corresponds to $mRNA(t)$. Assuming that both reactions occur spontaneously allows us to use mass-action kinetics to describe v_{syn} and v_{deg}. We can now rewrite Eq. (6.8) to

$$\frac{dn_{mRNA}(t)}{dt} = k_{syn} - k_{deg} \cdot n_{mRNA}(t), \tag{6.9}$$

which represents a first-order linear non-homogeneous differential equation. Solving Eq. (6.9) analytically yields

$$n_{mRNA}(t) = n_{mRNA}(0) \cdot e^{-k_{deg} \cdot t} + \frac{k_{syn}}{k_{deg}} \cdot (1 - e^{-k_{deg} \cdot t}), \tag{6.10}$$

which simplifies to

$$n_{mRNA}(t) = \frac{k_{syn}}{k_{deg}} \cdot (1 - e^{-k_{deg} \cdot t}) \tag{6.11}$$

if we start our simulations with zero molecules of mRNA. Additionally, from Eq. (6.9) we can determine the expected mean mRNA copy number in steady state which should be equal to

$$\langle n_{mRNA,ss} \rangle = \frac{k_{syn}}{k_{deg}} \tag{6.12}$$

and since this model is described by a Poisson distribution at steady state—

Poisson process

A stochastic process where events occur continuously and independently of one another.

—the coefficient of variation depends on

$$c_v = \frac{1}{\sqrt{\langle n_{mRNA,ss} \rangle}}. \tag{6.13}$$

This means that the stochasticity is inversely proportional to the steady-state mRNA copy number.

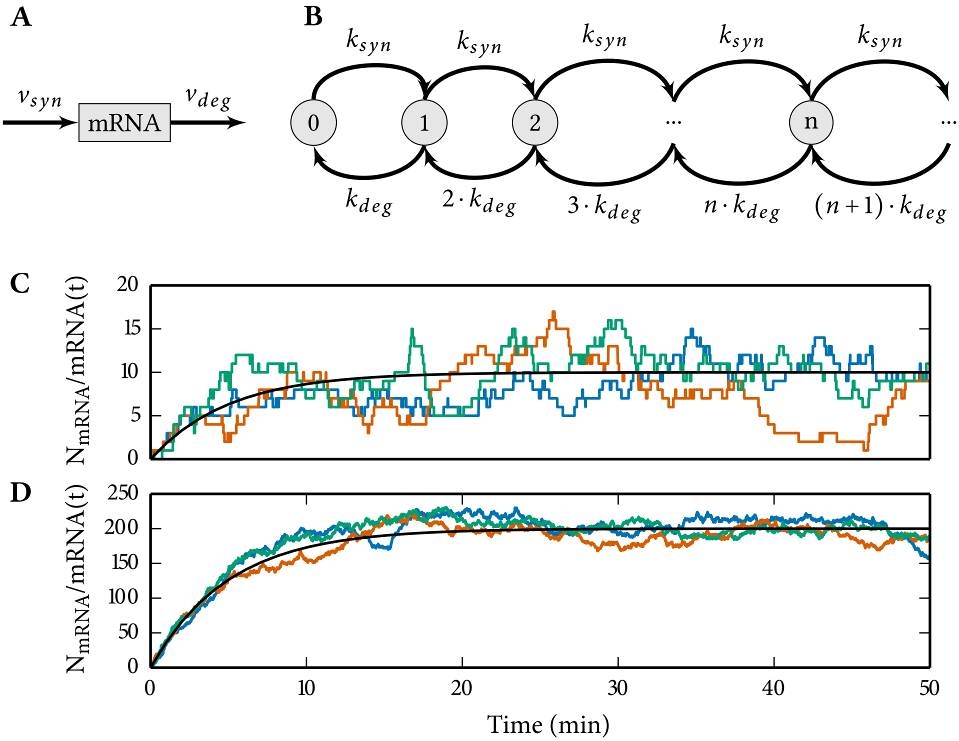


Figure 6.1. Modeling and simulation of single-cell transcription. *(A) v_{syn} and v_{deg} denote the synthesis and degradation rates of mRNA molecules respectively. (B) transition diagram of the Markovian synthesis-degradation process. Transition rates are indicated at the arrows. The synthesis rate is independent of n_{mRNA}. Panels C and D show time simulations with $k_{syn} = 2.0\ min^{-1}$, $k_{deg} = 0.2\ min^{-1}$ and $k_{syn} = 40\ min^{-1}$, $k_{deg} = 0.2\ min^{-1}$, respectively.*

In Fig. 6.1C-D, we compare the SSA output with the analytical solution for two different parameters settings. The results confirm the expectations: (i) the time realizations generated by the SSA fluctuate around the analytical solution and—in the steady-state regime—(also) around theoretical $\langle mRNA \rangle$ and (ii) c_v decreases with increasing $\langle n_{mRNA,ss} \rangle$; a deterministic approach cannot capture this stochasticity. We used an implementation of the direct method (Gillespie, 1976, 1977), the first SSA created by Gillespie, to generate these time realizations. In 1976, Gillespie introduced both the direct and the first-reaction method of which the direct method is more efficient when the number of reactions exceeds three. In the next section we discuss this direct method in more detail.

6.1 Direct Method

The direct method (Gillespie, 1976, 1977) is the classical SSA that is used to generate exact trajectories for biochemical models, which are in accordance with the CME. This means that the direct method generates a complete and detailed history of $\boldsymbol{x}$ by simulating every reaction event. To generate exact trajectories the direct method asks itself two questions:

(i) **when** does the next reaction fire?,

(ii) **which** reaction fires next?

More specifically, the first question asks only when a reactions fires, so it asks for a point in time. The second question asks only which reaction fires, not when this occurs (this is asked by the first question).

We start by providing the mathematics that are required to answer these questions. Then we show how the direct method determines when and which reaction fires next. The theoretical basis to answer these questions is given by

$$P(\tau, j|\boldsymbol{x}, t) = a_j(\boldsymbol{x}) \cdot e^{-a_0(\boldsymbol{x})\tau} \text{ with } (\tau \geq 0; j = 1, 2, \ldots, r), \tag{6.14}$$

which describes the probability that the reaction with index j fires at $t + \tau$ given that the system is in state $\boldsymbol{x}$ at time t, and where

$$a_0(\boldsymbol{x}) = \sum_{j=1}^{r} a_j(\boldsymbol{x}). \tag{6.15}$$

This means that $a_0(\boldsymbol{x})$ describes the sum of all the propensities. Eq. (6.14) is a joint probability function of τ and j given that the system is in state $\boldsymbol{x}$ at time t, that is,

$$P(\tau, j|\boldsymbol{x}, t) = P(\tau|\boldsymbol{x}, t) \cdot P(j|\tau, \boldsymbol{x}, t) \text{ with } (\tau \geq 0; j = 1, 2, \ldots, r). \tag{6.16}$$

Summing Eq. (6.14) over r reactions gives the probability that the next reaction fires at $t + \tau$,

$$P(\tau|\boldsymbol{x}, t) = \sum_{j=1}^{r} a_j(\boldsymbol{x}) \cdot e^{-a_0(\boldsymbol{x})\tau} \text{ with } (\tau \geq 0). \tag{6.17}$$

Using Eq. (6.15), we can rewrite (and simplify) this relationship to

$$P(\tau|\boldsymbol{x}, t) = a_0(\boldsymbol{x}) \cdot e^{-a_0(\boldsymbol{x})\tau} \text{ with } (\tau \geq 0), \tag{6.18}$$

which is an exponential distribution and therefore τ is an exponential random variable with mean $1/a_0$ and variance $1/a_0^2$. The probability that reaction with index j fires next is equal to its normalized propensity function,

$$P(j|\tau, \boldsymbol{x}, t) = \frac{a_j(\boldsymbol{x})}{a_0(\boldsymbol{x})} \text{ with } (j = 1, 2, \ldots, r). \tag{6.19}$$

We can obtain this relationship by integrating Eq. (6.14) over all τ. Multiplying Eqs. (6.18) and (6.19) yields as expected again Eq. (6.14).

To generate samples of both τ and j the direct method uses a Monte Carlo method:

Monte Carlo method

Any method which solves a problem by repeatedly generating random numbers

More specifically, τ is generated by

$$\tau = \frac{1}{a_0(\boldsymbol{x})} \cdot \ln(\frac{1}{rand_1}), \tag{6.20}$$

where $rand_1$ is a random number between [0,1]. By generating an array of $rand_1$ values, you can confirm that Eq. (6.20) generates an exponential distribution with mean $1/a_0$ and variance $1/a_0^2$. The direct method uses roulette wheel selection to determine the index j of the reaction that is going to fire next. For each time event, the direct method plays one game on the roulette wheel. The size of the bins are proportional to the likelihood that a reaction fires first; the reaction that fires first is the one where the roulette ball falls in the bin. The direct method does this by drawing a second random number $rand_2$ between [0,1] and selects the smallest integer j satisfying

$$\sum_{j'=1}^{j} a_{j'}(\boldsymbol{x}) > rand_2 \cdot a_0(\boldsymbol{x}). \tag{6.21}$$

We use the synthesis-degradation process with a constant synthesis rate (Fig. 6.2) to illustrate this roulette wheel selection method. The roulette wheel of this process is completely different than the typical roulette wheel you can encounter in a casino. It consists of two bins, a (black) bin for the synthesis reaction and a (red) bin for the degradation reaction (Fig. 6.2A). We start our simulations with zero molecules of mRNA (Fig. 6.2B), thus the likelihood that a synthesis reaction fires first is 100%. The roulette wheel, therefore, has at $t = 0$ only one bin; the bin for the synthesis reaction. Remember that the roulette wheel only determines which reactions fires, not when. As we generate more time steps, the system starts stochastically fluctuating around its steady state. The propensities for both reactions are then similar (Fig. 6.2B). After each time step, the roulette wheel is updated according to the current propensities (Fig. 6.2A). For instance, at $t \approx 31.54$ min the propensity of the synthesis reaction is higher than the propensity of the degradation reaction and the reverse is true at $t \approx 43.96$ min.

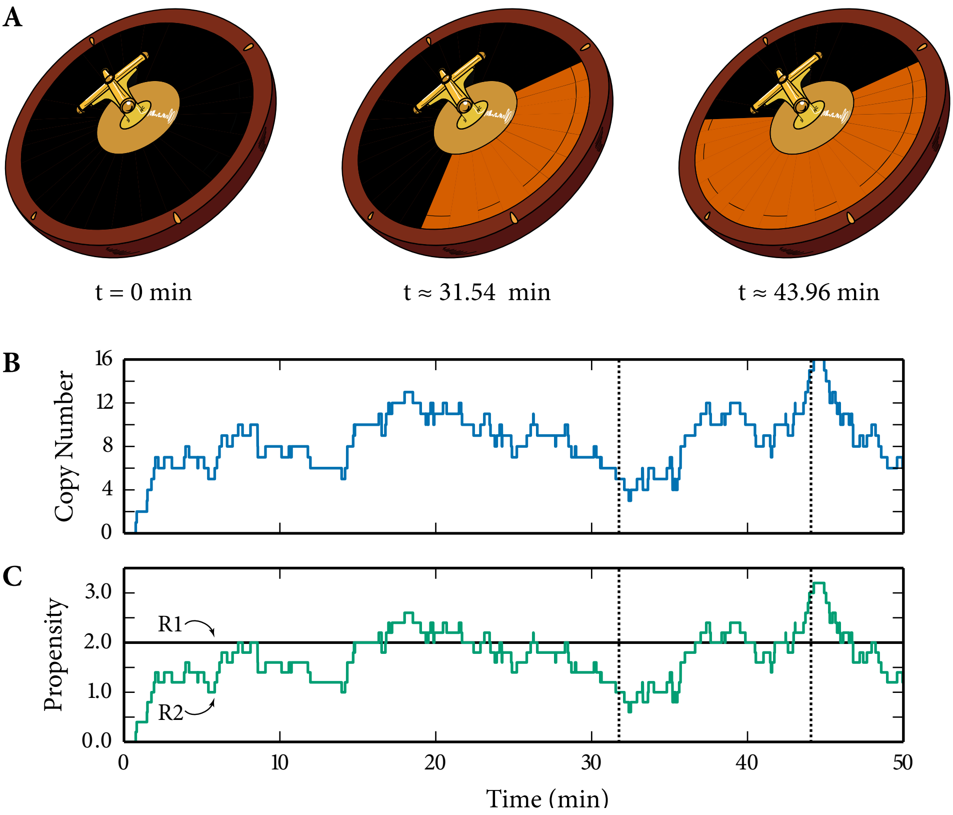


Figure 6.2. Illustrating roulette wheel selection. *(A) the roulette wheel selection procedure in action for the synthesis-degradation process with a fixed synthesis rate* (k_{syn} = 2.0 min^{-1}, k_{deg} = 0.2 min^{-1})*. Corresponding species and propensities time series data are given in panels B and C respectively.*

6.2 τ-leap method

Unfortunately, the direct method discussed in the previous section can be time-consuming. Reaction sampling has time complexity $O(r)$—

Big O notation

The time complexity of an algorithm is often expressed with the big O notation that excludes coefficients and lower order terms.

—and generating (and storing) a complete and detailed history of $\boldsymbol{x}$ can be a tedious process. Developing mathematically equivalent but more efficient implementations of the SSA has been an active research area, which led to the development of the next reaction method (Gibson and Bruck, 2000). In contrast to the direct method, this next reaction method recalculates only those propensities that alter after a specific reaction fired. The next reaction method has time complexity $O(\log(r))$ for reaction sampling, which can result in a significant speed-up compared to the direct method for models with many reactions. Despite this significant improvement (especially for sparse models), exact algorithms such as the direct and next reaction method can be too time-consuming for biochemical models with a high frequency of reaction events.

This means that the strength of exact SSAs—obtaining a complete and detailed history of $\boldsymbol{x}$ by simulating every reaction event—is also its weakness; the exact SSA can be too time-consuming. As we argue in Chapter 7, in systems biology access to a complete and detailed history of $\boldsymbol{x}$ is often required, e.g. for the calculation of event waiting times. However, a complete and detailed history of $\boldsymbol{x}$ is not always required, e.g. for generating a time series plot and the moments of random variables. To accelerate the exact SSA simulation, various software packages exploit a fixed-interval data storage approach which reduces the amount of stored data. A fixed-interval data storage approach can be useful when a complete and detailed history of $\boldsymbol{x}$ is not required.

However, these software packages do generate a complete and detailed history of $\boldsymbol{x}$. The τ-leap method developed by Gillespie (2001) goes one step further by generating a time interval τ after which all reactions fire simultaneously that occurred during τ. During the time interval $[t, t+\tau)$, the propensity functions are not updated which makes this τ-leap method an approximate SSA. Gillespie (2001), therefore, presented this method as *"an approximate procedure that in some circumstances can produce significant gains in simulation speed with acceptable losses in accuracy."*

The most important question of the τ-leap method is: What should be the size of τ? A too large τ results in non-acceptable losses in accuracy. Alternatively, a too small τ is inefficient—τ should be greater than $1/a_0$, the mean of the exponential random variable used in the exact SSAs to determine the time to the next reaction event.

We want the largest τ that satisfies the Leap Condition:

Leap Condition

τ must be small enough such that no propensity function alters substantially during the time interval $[t, t+\tau)$.

Different approaches have been developed to determine the largest τ that satisfies the Leap Condition (Gillespie, 2001; Gillespie and Petzold, 2003; Rathinam et al., 2003; Cao et al., 2006); we refer to Gillespie (2001) for the original τ-leap method and to Cao et al. (2006) for the optimized τ-leap method. For stiff systems, an implicit τ-leap method was developed by Rathinam et al. (2003).

After determining (preferably the largest) τ that satisfies the Leap Condition, the τ-leap method determines how often each reaction fires in the time interval $[t, t+\tau)$:

$$K_j(\tau; \boldsymbol{x}, t) = \text{the number of times reaction } j \text{ fires in } [t, t+\tau) \text{ for } j = (1, 2, \ldots, r). \tag{6.22}$$

If we satisfy the Leap Condition, we can assume that the propensity functions stay approximately constant during $[t, t+\tau)$. Each reaction event occurs independently of each other and we know the average rate (Eq. (6.3)), thus we can (accurately) approximate K_j by a Poisson random variable $P(a_j(\boldsymbol{x}), \tau)$ with mean and variance $a_j(\boldsymbol{x})\tau$.

We can now easily show that $\tau \leq 1/a_0$ is inefficient: $\tau = 1/a_0$ yields a mean value for K equal to $K_j = a_j(\boldsymbol{x})/a_0$ which is equivalent to the probability that reaction j fires next in the exact SSA given in Eq. (6.19). This means that, (i) if $\tau = 1/a_0$ typically one of the generated K_j's is one and all others are zero and (ii) if $\tau < 1/a_0$ typically all of the generated K_js are zero. That is also why the τ-leap method switches (temporarily) to an exact SSA if τ is not sufficiently greater than $1/a_0$. Alternatively, if $\tau >> 1/a_0$ typically multiple K_j's are nonzero and it is beneficial to use the τ-leap approximation to update $\boldsymbol{x}$ from t to $t+\tau$ as is given by

$$\boldsymbol{x}(t+\tau) = \boldsymbol{x} + \sum_{j=1}^{r} \boldsymbol{v}_j \cdot P_j(a_j(\boldsymbol{x}), \tau), \tag{6.23}$$

where $\boldsymbol{v_j}$ is the state-change vector of reaction index j.

The Poisson random variable $P(a_j(\boldsymbol{x}), \tau)$ is unbounded at the high side (bounded at zero on the low side), which means that applying Eq. (6.23) can result in negative species copy numbers. This usually occurs with species that act as reactant in (multiple) reaction(s) and that are only multiple firings away of depletion. The more recent τ-leap methods check, before updating $\boldsymbol{x}$, if species copy numbers go negative by applying Eq. (6.23).

To illustrate the τ-leap method, we use the decaying-dimerizing model because—for the right parameter settings—we can, for the majority of the simulation, determine a $\tau >> 1/a_0$ that satisfies the Leap Condition. This model consists of three species and four reactions (Fig. 6.3A). All reactions are first-order reactions, except of reaction v_2 which is a second-order reaction with identical reactants (dimerization of S1). Discrete stochastic simulation algorithms use molecular numbers, so we have to take into account that this reaction requires $n_{S1} \geq 2$ to be able to occur. This gives the following propensity functions:

$$\begin{aligned} a_1(\boldsymbol{x}) &= k_1 \cdot n_{S1}, \\ a_2(\boldsymbol{x}) &= \frac{1}{2} k_2 \cdot n_{S1} \cdot (n_{S1} - 1), \\ a_3(\boldsymbol{x}) &= k_3 \cdot n_{S2}, \\ a_4(\boldsymbol{x}) &= k_4 \cdot n_{S2}, \end{aligned} \tag{6.24}$$

and state-change vectors:

$$\begin{aligned} \boldsymbol{v_1} &= (-1,\ 0,\ 0), \\ \boldsymbol{v_2} &= (-2,\ 1,\ 0), \\ \boldsymbol{v_3} &= (2,\ -1,\ 0), \\ \boldsymbol{v_4} &= (0,\ -1,\ 1). \end{aligned} \tag{6.25}$$

For our parameter settings—$k_1 = 1.0$, $k_2 = 0.002$, $k_3 = 0.5$, and $k_4 = 0.04$—and initial conditions—$n_{S1} = 10^5$ and $n_{S2} = n_{S3} = 0$—we performed 1000 stochastic simulations for the direct and the optimized τ-leap method (Cao et al., 2006). Each simulation continued until all reactants (i.e. S1 and S2) were exhausted.

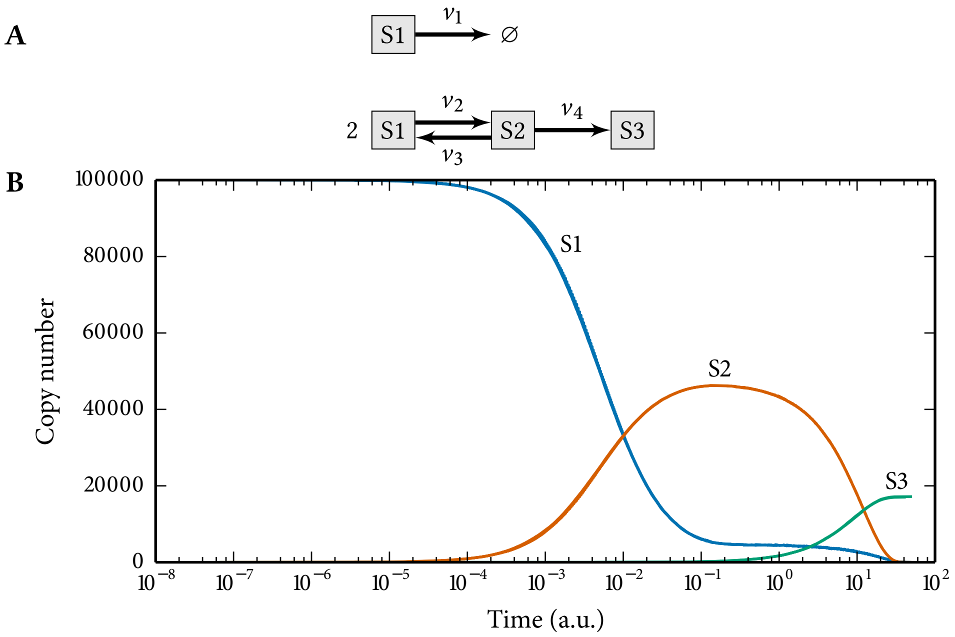


Figure 6.3. Modeling and simulation of a decaying-dimerizing process with exact and inexact stochastic algorithms. *(A) monomer S1 decays (v_1) or forms a (unstable) dimer S2 (v_2) which can subsequently be (re)converted into S1 (v_3) or the stable dimer S3 (v_4). (B) stochastic simulation for both the direct method and the optimized τ-leap method. Differences cannot be observed by the naked eye (i.e. the τ-leap method predicts the same time series behavior in only a fraction of the time of the direct method).*

In Fig. 6.3B, we overlay output generated with the direct and the optimized τ-leap method; no differences can be observed by the naked eye thus the τ-leap method accurately approximates the exact output of the direct method. The final n_{S3} count gives the number of stable dimers that were created during each simulation of which we found no significant difference between both methods (17040 ± 94 vs. 17044 ± 92). Similarly, for the simulation end time there is no significant difference between both methods (46.2 ± 2.5 vs.

46.2 ± 2.4) In contrast, there is a significant difference between the number of time steps used by both methods (525817 ± 1779 vs. 4235 ± 172). The optimized τ-leap method was, therefore, significantly faster; the exact difference depends on the implementation of both methods (and the allowed loss in accuracy of the τ-leap method).

6.3 Illustrating the importance of stochastic simulations in Systems Biology

In this section, we discuss in detail four different examples to illustrate the important of stochastic simulations in systems biology. In each example, we used the direct method implementation from StochPy (discussed in detail in Chapter 7) to generate stochastic simulations.

Example 1: Modeling and simulation of a synthesis-degradation process

In our first example, we use an elegant example which was also used by Wilkinson (2011) in his book "Stochastic modeling for systems biology". That is a molecule S which catalyzes besides its own degradation also its own synthesis (Fig. 6.4A). This means that both reactions are dependent on n_S which makes $n_S = 0$ an absorbing state, a state that once entered, cannot be left (Fig. 6.4B). We can solve this synthesis-degradation process analytically and we use a relatively large number of initial molecules. You might wonder what additional insight a stochastic simulation can provide. That is what we show in this example.

We start by writing down the mass-balance equation for the macroscopic description of the synthesis-degradation process:

$$\frac{dS(t)}{dt} = (k_{syn} - k_{deg}) \cdot S(t). \tag{6.26}$$

This mass-balance equation is a first-order differential equation, thus we can analytically solve this system which gives

$$S(t) = S(0) \cdot e^{(k_{syn} - k_{deg})t}. \tag{6.27}$$

We now know the analytical solution, so do we now understand how this model behaves under any circumstances? You might argue that we do with one exception: There could be stochastic fluctuations present if there are only a few S molecules present in the cell. Lets study this system with $S(0) = 1000$ (volume = 1) and think about the following question: Do you expect different behavior for the parameter sets $k_{syn} = 1$ min^{-1}; $k_{deg} = 2$ min^{-1}, $k_{syn} = 9$ min^{-1}; $k_{deg} = 10$ min^{-1}, and $k_{syn} = 19$ min^{-1}; $k_{deg} = 20$ min^{-1}. The analytical solution given in Eq. (6.27) shows that the behavior should be identical. This means that the simulation only depends on the difference between both parameters ($k_{syn} - k_{deg}$). This yields three types of behavior when $t \rightarrow \infty$: $k_{syn} > k_{deg}$ causes $S(t)$ to increase exponentially, $k_{syn} > k_{deg}$ causes $S(t)$ to decrease exponentially, and $k_{syn} = k_{deg}$ causes $S(t)$ to stay constant.

Now, we are going to use stochastic simulations to demonstrate that this allows us to (i) do better parameter estimation and (ii) ask more questions such as time to extinction ($n_S \rightarrow 0$) and the uncertainty of n_S at a specific point in time. We started by using stochastic simulations to generate three independent time trajectories for the three parameter sets (Fig. 6.4C). While the analytical solution is identical for the three parameter sets, the stochastic simulations show remarkably different behavior.

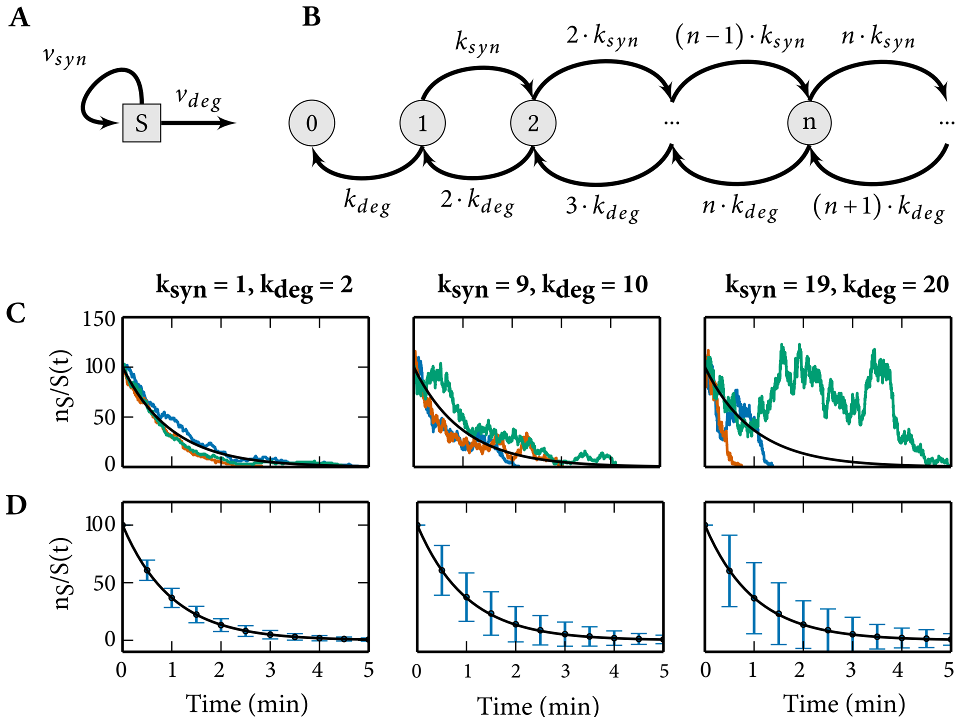


Figure 6.4. Modeling and simulation of the synthesis-degradation process. *(A) v_{deg} and v_{syn} denote the synthesis and degradation rates of S molecules respectively. (B) transition diagram of the Markovian synthesis-degradation process. Transition rates are indicated at the arrows. (C) stochastic simulations (blue, red, green) of the synthesis-death process for different parameter settings. (D) the sample means and standard deviations of n_S from the simulation runs at t = 0, 0.5, 1, ..., 5. Solid black lines denote the analytical solution.*

To explain this behavior, we performed another stochastic simulation where we generated 1000 independent time trajectories for the three parameter sets from which we calculated the sample mean and standard deviations at t = 0, 0.5, 1, ..., 5 (Fig. 6.4D). These results show that for all parameter sets, the sample mean corresponds to the analytical mean at every point in time. Additionally, these results demonstrate that the sample standard deviation (i.e. the noise) increases with increasing parameter values. This means that our stochastic simulations show us that:

- $k_{syn} - k_{deg}$ determines the mean, and
- $k_{syn} + k_{deg}$ determines the noise.

Assuming that we have adequate experimental data available, stochastic simulations can determine (i) both k_{syn} and k_{deg}, (ii) the uncertainty at a specific point in time for a given set of parameters, and (iii) the time until extinction (with statistical variance); in the deterministic approach, the number (or concentration) of S molecules never reaches zero.

Example 2: Modeling and simulation of bursty single-cell transcription

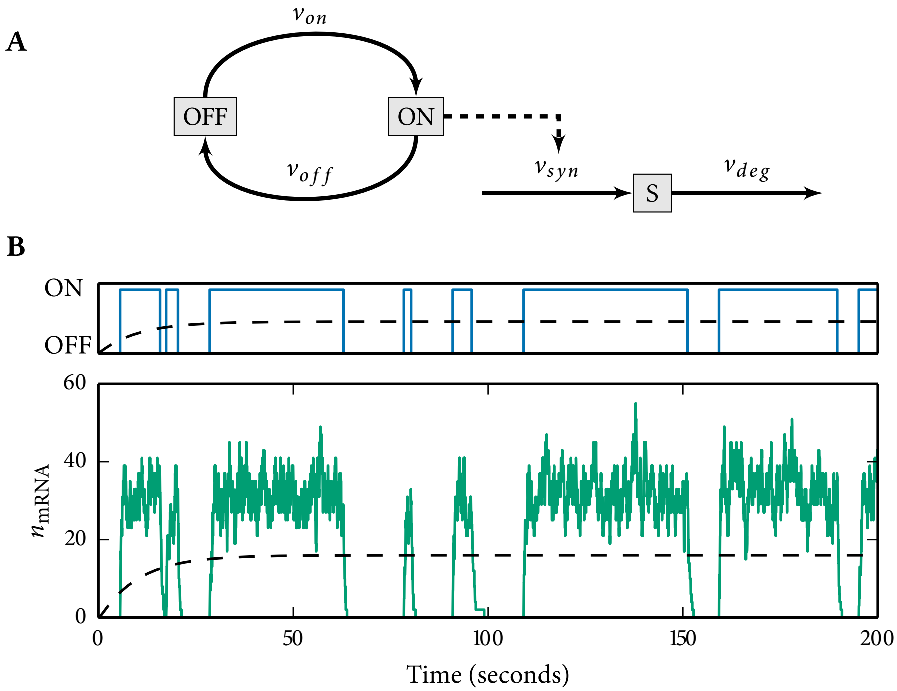


Figure 6.5. Modeling and simulation of bursty single-cell transcription. *(A) v_{on}, v_{off}, v_{syn}, and v_{deg} denote the ON switching, OFF switching, synthesis, and degradation rates, respectively. (B) simulations using deterministic (dashed lines) and stochastic (solid lines) simulations.*

In our second example we added two possible states (the ON state and the OFF state) to the synthesis-degradation model of mRNA shown in Fig. 6.1. The mRNA synthesis occurs only in the ON state, whilst mRNA degradation occurs in both states (Fig. 6.5A; we discuss this model in detail in Chapter 7). Depending on the choice of kinetic parameters, this system can display transcription bursts that can cause significant cell-to-cell variability in mRNA copy numbers (Shahrezaei and Swain, 2008; Dobrzynski and Bruggeman, 2009). The bursty nature of this model is explained by a gene that switches spontaneously between an inactive (OFF) state and an active (ON) state. Synthesis of mRNA molecules occurs only during the ON state whilst degradation of mRNA molecules occurs continuously.

In contrast to our previous example, we cannot analytically solve the time evolution of this system thus we focus here on numerical simulations. In Fig. 6.5B we compare deterministic and stochastic simulation output for a bursty transcription mode. The results show that the stochastic simulation (solid lines) successfully predicts the bursty behavior, whereas the deterministic simulation (dashed lines) fail to predict the bursty behavior. The reason is that the stochastic simulation predicts the behavior of a single cell and the deterministic simulation predicts the behavior of a population of cells. In Chapter 7 we show that transcription bursts lead to a bimodal mRNA copy number distribution and two time scales in the distribution of event waiting times of mRNA synthesis times.

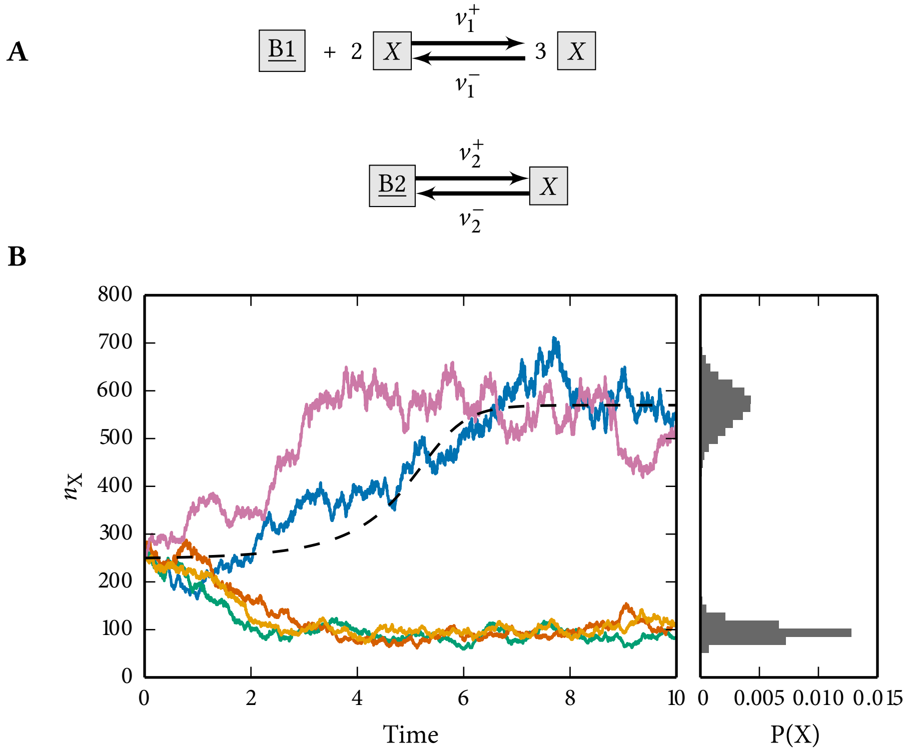


Figure 6.6. Schlögl reaction system. *(A) the Schlögl reaction system is an autocatalytic, trimolecular reaction scheme where species B1 and B2 are considered fixed. (B) simulations of the Schlögl reaction system using deterministic (dashed lines) and stochastic (solid lines) simulations. The left panel shows five different time trajectory simulations and the right panel shows the distribution at* $t = 10$ *after running 250000 simulations.*

Example 3: Schlögl reaction system

Our third example concerns a reaction system shown in Fig. 6.6A developed by Schlögl (1972). We do not discuss this model in detail (we refer interested readers to Vellela and Qian (2009)), but use it as an example to illustrate (potential) differences between output of deterministic and stochastic simulations. In our previous example, we briefly touched upon bistability which means that the system has two stable equilibrium states. The Schlögl reaction system has a bistable steady-state distribution for a specific set of parameter values, and that is also the case for the set of parameter values we use here.

We used stochastic simulations to generate five independent time trajectories that ended-up, as expected, in one of the two stable states (left panel of Fig. 6.6B). The deterministic simulation, in contrast, generates given the set of initial conditions always the same output. This means that a deterministic simulation yields, for a given set of initial conditions, one point from the full distribution and can, therefore, find at best one of the stable states. By generating sufficient time steps stochastic simulations can also reveal the full probability function (right panel of Fig. 6.6B).

Example 4: Stochastic modeling and simulation predicts active-state locking

Our final example concerns an interesting phenomenon that we call “stochasticity-induced active-state locking”, a new behavior—we recently predicted—of two-component signaling systems that arises only in stochastic simulations (Wei et al., 2014). The model (Fig. 6.7A) we propose captures the basic design of two-component systems and differs in several ways from previous models. Upon binding a signaling ligand (L), the sensor S changes conformation and autophosphorylates, yielding the phosphorylate species SP. Next, SP forms a complex with response regulator R, after which phosphotransfer can take place to yield RP and S. Typically, RP acts as a transcription factor inducing a gene expression response (Stock et al., 2000). In our model the sensor is bifunctional: When S is neither phosphorylated nor bound to L it can also dephosphorylate RP. Most sensor proteins have this property, including EnvZ from the EnvZ/OmpR model system responsible for osmoregulation in *Escherichia coli* (Cai and Inouye, 2002).

In our simulations L was considered fixed—equivalent to assuming a large extracellular reservoir of the signaling molecule. Also, the intracellular concentrations of ATP and inorganic phosphate are considered fixed and therefore not explicitly modeled. When the L concentration is high, high levels of RP are expected to be produced and the system is in the “active-state”. Conversely, when the signal is weak (or absent) the amount of RP is low; we refer to this situation as the “off-state”.

We studied the response time of the system—the time that the system requires to attain

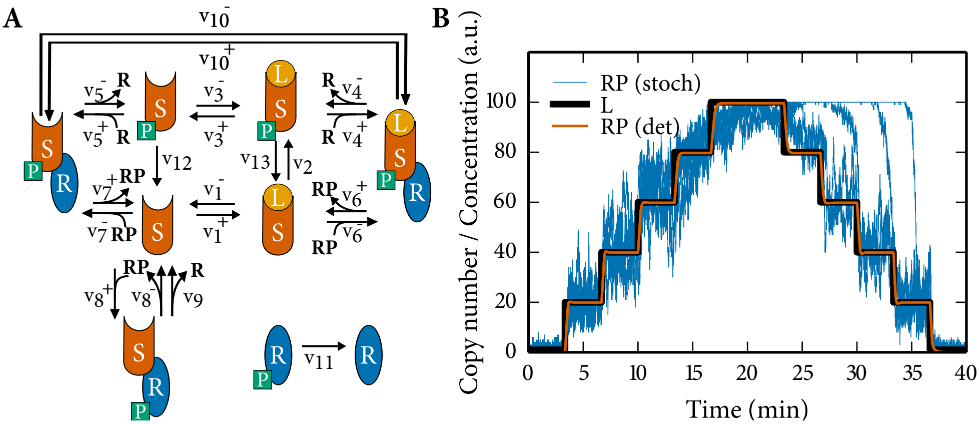


Figure 6.7. Stochastic modeling and simulation predicts active-state locking. *(A) network reaction model where fixed species ATP, ADP, and P_i are not shown. (B) the response of the system to time-varying signal strengths, where copy numbers of R and S are fixed at 100. The concentration of L, shown in black, was increased stepwise at 200 second intervals from 1 to 100 a.u., then subsequently decreased. The deterministic prediction for the output RP concentration is shown in red, and 10 stochastic simulations are shown in blue. The deterministic response time ranges from 20–60 seconds, but for the stochastic simulations active-state locking is seen to occur after L reaches 100 a.u., when saturation occurs.*

a steady-state RP concentration. To address the response time, we performed stochastic simulations of the signaling network (Fig. 6.7A) and subjected the system to a stepwise-varying signal strength, ranging from negligible to saturating L concentrations (Fig. 6.7B). The deterministic simulation shows that the response time ranges from approximately 20 to 60 seconds. The stochastic simulation agrees with this timescale when the signal is increasing. However, when the signal is reduced from a saturating level, some stochastic trajectories show surprising behavior. The RP concentration has a tendency to stay "locked" in the active-state for a prolonged time period—sometimes over ten minutes—after the signal has been reduced. We call this interesting phenomenon "stochasticity-induced active-state locking".

The cause of stochasticity-induced active-state locking is that the number of (unbound, unphosphorylated) S molecules can drop to zero when the system is subjected to saturating signal strengths. As the S molecules are the dominant means of dephosphorylation of RP, their absence prevents the number of RP molecules from declining. In our model this is achieved in the simplest way as only the free sensor state, S, can dephosphorylate RP (reactions v_8^+ and v_9). The autodephosphorylation rates are slow (reactions v_{11} to v_{13}), and although the phosphorylation reactions are reversible, the forward rates (reactions v_6^+ and v_7^+) are overwhelmingly favored. Thus, the new lower equilibrium value of RP at the reduced level of L can only be reached once unbound, unphosphorylated S molecules are recovered by unbinding/dephosphorylation reactions (e.g. reactions v_3^- and v_{12}). Since these reactions occur stochastically and at a low rate, the observed waiting time can be remarkably long; in our simulations, we have observed locking for up to ten minutes. As long as those reactions are reversible and occur at a low enough rate, the time that it takes before S molecules reappear can be quite variable amongst individual cells.

We emphasize that this result is not reproducible in deterministic simulations. The reason is that, deterministically, the concentration of S is a continuous variable that would take a small but non-zero value under saturating conditions and reactions which increase S (and decrease RP) concentrations start to occur immediately (albeit at a low rate) upon decreasing the signal strength.

If each stochastic trajectory is interpreted as an individual cell in an isogenic population, this locking phenomenon should thus create a similarly transient bimodal distribution, which should be experimentally observable. A quantification of the variability of the response time can be obtained by numerically determining the distribution of the time that it takes for cells to reach a threshold % of unphosphorylated R molecules. This resembles a hitting time and is defined in the analysis of the stochastic time series. As long as cells do not communicate via cell-to-cell signaling, stochasticity-induced active state locking can transient diversify populations of cells which can be advantageous in unpredictable environments.

6.4 Concluding Remarks

We described stochastic simulation algorithms and highlighted their importance for computational systems biology in general and more specifically for analyzing fluctuations in biochemical models. This is of increasing interest because of the recent advances in single-cell experiments that indicate the importance of stochastic phenomena in cell biology. Albeit it was not the topic of this chapter, various stochastic simulation software packages have been developed that allow the community to study these stochastic phenomena. In the next chapter, we present one of these packages, StochPy.

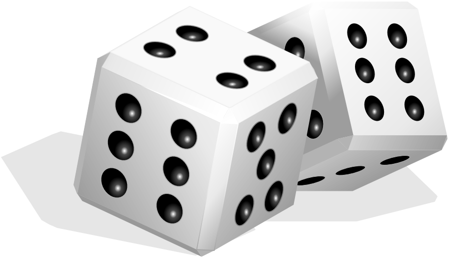
6.5 Supplemental Materials

Dataset S1. Annotated scripts and input files used to generate modeling and simulation results can be found at http://persistent-identifier.org/?identifier=urn:nbn:nl:ui:18-23541.

This chapter is based on the publication:

Maarleveld T.R., Olivier B.G., Bruggeman F.J. (2013) *“StochPy: a comprehensive, user-friendly tool for simulating stochastic biological processes”*, **PLoS ONE**, 8, 11:e79345.

StochPy



Stochastic modeling & simulation in Python

StochPy is a comprehensive and user-friendly software package that provides stochastic simulation algorithms in Python.

Simulating fluctuations in biochemical models with StochPy

7

Abstract

Single-cell and single-molecule measurements indicate the importance of stochastic phenomena in cell biology. Stochasticity creates spontaneous differences in the copy numbers of key macromolecules and the timing of reaction events between genetically-identical cells. Mathematical models are indispensable for the study of phenotypic stochasticity in cellular decision-making and cell survival. There is a demand for versatile, stochastic modeling and simulation environments with extensive, preprogrammed statistics functions and plotting capabilities that hide the mathematics from the novice users and offers low-level programming access to the experienced user. Here we present StochPy (*Stoch*astic modeling and simulation in *Py*thon), which is a flexible software tool for stochastic simulation in cell biology. StochPy provides various stochastic simulation algorithms, SBML support, analyses of the probability distributions of molecule copy numbers and event waiting times, analyses of stochastic time series, and a range of additional statistical functions and plotting facilities for stochastic simulations. We illustrate the functionality of StochPy with stochastic models of gene expression, single-molecule enzyme kinetics, and cell division. StochPy has been successfully tested against the SBML stochastic test suite. StochPy is a comprehensive software package for stochastic simulation of the molecular control networks of living cells. It allows novice and experienced users to study stochastic phenomena in cell biology. The integration with other Python software makes StochPy both a user-friendly and easily extendible simulation tool.

7.1 Introduction

Experiments at the level of single cells indicate large cell-to-cell variability in copy numbers of molecules (Balazsi et al., 2011). Inevitably, this molecular noise originates from stochastic fluctuations and has a large impact on cellular dynamics (Raj and van Oudenaarden, 2008). Traditional deterministic chemical kinetics fail to capture the dynamics of these systems. Stochastic systems are typically mathematically described by the chemical master equation (CME) (Kampen, 1992), which rarely has a closed form solution and therefore numerical simulation is a necessity. Stochastic simulation algorithms (SSAs) generate time trajectories that are in agreement with the CME. Many SSAs have been developed (Gillespie, 2007), however, in order to effectively use them in cell biology a flexible simulation environment is required. This should include, for instance, easy access to functionality for statistical analysis, plotting and interpretation of the raw simulation results, all the while shielding the user from the underlying mathematics.

Stochastic simulation software is used in a variety of methodologies such as ordinary differential equations (Kierzek, 2002; Ramsey et al., 2005; Hoops et al., 2006; Siso-Nadal et al., 2007; Pineda-Krch, 2008; Myers et al., 2009; Sanft et al., 2011) and Petri nets (Rohr et al., 2010), each tool having its own unique strengths and weaknesses. The recent advances in the experimental investigation of single-cells has increased the interest in the analysis of the statistics of event waiting times (Golding et al., 2005; English et al., 2006; Dobrzynski and Bruggeman, 2009; Li and Xie, 2011). Most simulators cannot calculate these event waiting times because they do not return the raw stochastic simulation output (hereafter explicit output). In addition, stochastic simulations are not as straightforward as solving deterministic systems and there is a demand for a versatile stochastic simulation environment that is easy to use and extend. Altogether, this motivated us to develop a flexible and interactive open-source stochastic simulator StochPy: *Stoch*astic modeling and simulation in *Py*thon.

StochPy provides various SSAs for the simulation of stochastic dynamics and supports model definition in either plain text or the Systems Biology Markup Language (SBML) (Hucka et al., 2003). In addition, StochPy includes statistical functions for the numerical analysis of stochastic simulations as well as plotting facilities for the visualization of amongst other features time-correlations, propensities, and event waiting times.

7.2 Results and Discussion

Software Implementation

The StochPy software has been designed around four core principles. Functionality, StochPy should support a variety of SSAs and allow for the intuitive description of models (i.e. reactions and species). Flexibility, StochPy should support high-level statistical and plotting functions for interrogating both model and data as well as provide programmatic access to low-level functions and data structures. User-friendly, StochPy should provide high-level, user-friendly access tailored for interactive use. Open-source, StochPy should be open-source such that the community can inspect, modify and distribute our software.

To satisfy these principles StochPy has been developed as a console application using the Python language, taking advantage of its pure object-oriented nature, portability, extensive standard library, and ability to seamlessly glue together scientific libraries written in compiled languages. Matplotlib (Hunter, 2007) is integrated for plotting, providing publication-quality image generation. An object-oriented design allows for the simultaneous analysis of multiple instances of a model, using state-of-the-art stochastic simulation capabilities either interactively or via user-defined scripts. The high-level functionalities ensure that knowledge of the Python programming language is not required although, any scripting knowledge enhances the modeling and simulation (M&S) experience.

Combining these functionalities with those provided by the many available Python scientific libraries allows for easy extension of StochPy as well as its use as a library in other simulation software. The StochPy software has already been incorporated as a plug-in library for the systems biology simulator software PySCeS (Olivier et al., 2005). This provides a single interactive environment where model properties can be set interactively and simulated in both a stochastic and deterministic manner.

The following SSAs are implemented in StochPy: the direct, first reaction, next reaction, and optimized tau-leaping methods (Gibson and Bruck, 2000; Cao et al., 2006; Gillespie, 2007). Whilst the direct method is selected by default, the next reaction, and tau-leaping methods can be used to boost performance for models that are either sparse or have many fast reactions and/or large molecule numbers, respectively. These SSAs assume that the reactions inside a biological system occur instantly, but this is not always true. Common examples are transcription and translation. Therefore, we also implemented the delayed direct method (Cai, 2007) and delayed next reaction method (Anderson, 2007). These delayed stochastic algorithms extent the SSA to a system with delays. We tested each implementation with the SBML stochastic test suite (Evans et al., 2008), the results of which are given in Dataset S1. All 36 tests were successfully passed by StochPy, except test 3.4 where molecule copy numbers are reset whenever the copy number threshold is reached. A similar observation was done by Erhard et al. (2008).

Model definition is by way of the human readable/writable PySCeS model description language (MDL) (Olivier et al., 2005). A simple and intuitive approach to creating and editing models understandable to experimentalists and theoreticians. libSBML (Bornstein et al., 2008) is used for importing stochastic models written in SBML format which are then subsequently translated into PySCeS MDL. This requires maintaining one MDL that supports features which in principle can be encoded in any SBML instance. Currently, StochPy implements SBML Level 2 version 4 import and export functionalities. SBML Level 3 and package support is now being investigated.

The StochPy software provides interfaces to other widely used simulation tools: CAIN and StochKit2. Through these integrated interfaces, modelers are provided with a choice of SSA implementations that differ in speed and simulation output. For example, using StochKit2 via StochPy allows simulating models defined in SBML up to L2V4 or PySCeS MDL by StochKit2's fast solvers. The output can then directly by analyzed in StochPy, without the need to install any additional software (by default StochKit2 uses MATLAB to provide this functionality).

Performing StochPy simulations

A typical StochPy session consists of first creating a StochPy model object from a (default) input model. Different user-defined models (in SBML or PySCeS MDL) can be loaded into the model object. Once a model is loaded various simulation parameters can be set such as the number of simulation steps and the number of simulation trajectories. Subsequently, kinetic parameter values and species copy numbers can be modified interactively. Simulations can be performed by calling the available analysis methods for model objects. As model objects are fully encapsulated, multiple models can be instantiated from the same (or different) input files at the same time. An example of a short StochPy session is:

```
import stochpy
smod = stochpy.SSA()
smod.Model("dsmts-001-01.xml")
smod.DoStochSim(end=1000,mode="steps")
smod.PlotSpeciesTimeSeries()
```

where we initiate the model object `smod` for the default input model, load a different model depicted in SBML into the model object `smod`, generate one time trajectory of the CME (1000 steps), and plot the corresponding (discrete) species time series data.

In the following sections we discuss the potential uses of explicit output in systems biology, highlight StochPy's capabilities by M&S of different biological systems, and benchmark StochPy against other widely used stochastic tools. All simulations were done with StochPy's implementation of the direct method and Figs. 7.1–7.6 were created with StochPy. Only a single command, a high-level function such as `PlotSpeciesTimeSeries()`, is necessary to create most of the shown (sub)-figures. Annotated scripts and input files used to generate the presented results are available as Scripts S1. More information on installing and using StochPy can be found in the *StochPy User's Guide* which together with additional example sessions is available online at http://stochpy.sf.net.

The potential uses of explicit output in systems biology

StochPy returns explicit output rather than discretized output (hereafter fixed-interval output). With fixed-interval output we mean that the state of the system (i.e. the copy numbers of all molecules) are reported for fixed-time intervals and not at the times in the system when single reaction events occur. Mathematically speaking, molecular reaction systems are modeled by continuous-time discrete state Markov chains and fixed-interval data storage approaches simulate these systems with continuous time but store the output with fixed-time intervals. Returning fixed-interval output likely derives from stochastic simulation practices in mathematical statistics. In systems biology, the requirements for stochastic simulation are often different. Access to exact simulation times allows for the straightforward calculation of event waiting times, species and propensity distributions, and correlation times. As these quantities are in principle observable in single-cell experiments, they should be calculable with simulation software.

Table 7.1 illustrates typical output of a StochPy stochastic simulation: n_i denotes the number of molecules of molecular species index i and a_j denotes the propensity of reaction index j at time points when reaction events occurred. This simulation output can be analyzed using pre-defined statistical functions to plot time series, or calculate distributions, moments, and time-correlations. StochPy returns explicit output and therefore waiting times for particular reactions events or system delays can be calculated. The time between consecutive activities of reactions is called an event waiting time. An example of these event waiting times is shown in Table 7.1 where the times between consecutive "firings" of reaction R4 are shown.

Time	n_1	n_2	n_3	a_1	a_2	a_3	a_4	reaction index	Δt_{R4}
0.000	0	1	0	0	0.05	0	0	–	
39.976	1	0	0	0.05	0	80	0	2	
40.051	1	0	1	0.05	0	80	2.5	3	
40.051	1	0	2	0.05	0	80	5	3	
40.074	1	0	1	0.05	0	80	2.5	4	
40.092	1	0	2	0.05	0	80	5	3	
40.325	1	0	1	0.05	0	80	2.5	4	0.251
40.413	1	0	2	0.05	0	80	5	3	(40.325-40.074)
40.497	1	0	1	0.05	0	80	2.5	4	0.172
40.509	1	0	2	0.05	0	80	5	3	
40.548	1	0	1	0.05	0	80	2.5	4	0.051
40.584	1	0	2	0.05	0	80	5	3	
40.748	1	0	3	0.05	0	80	7.5	3	
40.786	1	0	4	0.05	0	80	10	3	
40.888	1	0	5	0.05	0	80	12.5	3	
40.915	1	0	4	0.05	0	80	10	4	0.367
40.941	1	0	5	0.05	0	80	12.5	3	
40.954	1	0	4	0.05	0	80	10	4	0.039
40.959	1	0	5	0.05	0	80	12.5	3	
40.978	1	0	4	0.05	0	80	10	4	0.024

Table 7.1. StochPy simulation output. *The explicit simulation output of StochPy reports the number of molecules of each molecular species and the reaction propensities at each time point when a reaction occurs. The time differences between consecutive rows indicate waiting times between reaction events. In the last column, the event waiting times for reaction R4 are given. They correspond to the time period between consecutive instances of activity of R4.*

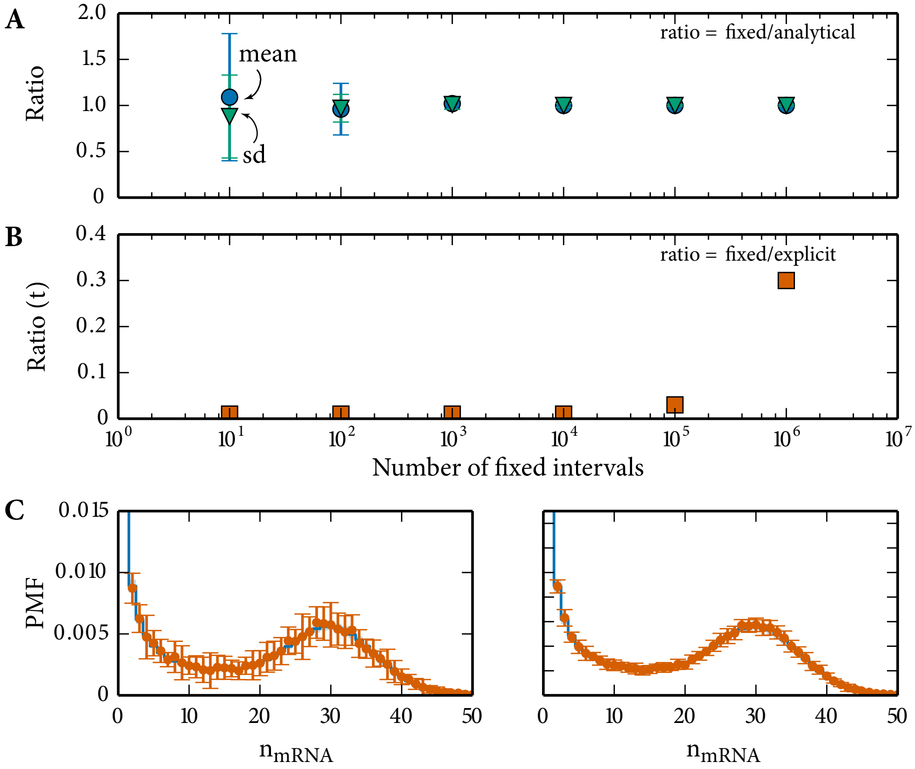


Figure 7.1. Fixed interval versus explicit simulation output. *We performed 100 stochastic simulations until t=60000 min (≈ 1000000 time steps) with* k_{ON} = 0.05 *min*$^{-1}$, k_{OFF} = 0.45 *min*$^{-1}$, k_{syn} = 80 *min*$^{-1}$, *and* k_{deg} = 2.5 *min*$^{-1}$. *(A) accuracy of mean and standard deviation estimates as function of the number of fixed intervals. (B) simulation time with fixed-interval output increases with the number of fixed intervals. We used the StochPy interface to StochKit2 to perform the fixed-interval simulations. The time to calculate the associated probability distributions is included. The stationary mRNA distribution for 10000 (panel C) and 1000000 (panel D) fixed intervals (red error bars, 1.96 SD) vs. explicit output (blue 95% confidence interval); 1.96 SD corresponds to a 95% confidence interval.*

In Fig. 7.1 we highlight several differences between fixed-interval and exact output for a model we further discuss in case study 2: “Stochastic mRNA synthesis by a switching gene”. We ran a stochastic simulation at a stationary state of the model with StochPy to obtain 1000000 reaction events and obtained all the stochastic dynamics in terms of the exact output. Next, we ran StochKit2 until the same model end time, as we obtained with StochPy, and varied the fixed-interval size at which the state of the stochastic system is stored in the report file of StochKit2. In Fig. 7.1A we plot the estimate of the mRNA mean copy number and standard deviation, obtained from 100 simulations, as function of the number of fixed intervals. This plot indicates that for this system ≈ 10000 intervals are enough to accurately estimate the mRNA mean copy number and standard deviation. Fig. 7.1B shows the ratio of the calculation time of StochKit2 and StochPy and indicates that StochKit2 is indeed a lot faster as it only required to write 10000 events to file as compared to StochPy, which stored 1000000 events. StochKit2's C++ implementation makes the simulation about a factor of ten faster than StochPy when they store an equal amount of reaction events. Calculating the associated stationary probability distribution required additional time which reduced the speed difference to a factor of three.

A complication with fixed-interval storage is that the user does not know beforehand what the relevant fixed-interval size and number should be and, for instance, data bootstrapping should be applied to assess the accuracy of the calculation. For instance, 10000 fixed-intervals are not enough to determine the stationary probability distribution with great accuracy, as shown in Fig. 7.1C (the error bars denote the variation in the distributions between the 10 simulations done at each fixed-interval number). Using more fixed-intervals results in determining the stationary probability distribution with a better accuracy. However, Fig. 7.1D shows that a better accuracy was obtained using 1000000 explicit reaction events than with 1000000 fixed-intervals. Therefore, for this particular example, more fixed-intervals than actual reactions events are necessary to obtain a stationary probability distribution with a similar accuracy.

This means that the speed of a fixed-interval algorithm is not set by the end time and the programming language, as is the case for an exact output approach, but also by the chosen number of fixed intervals. Deciding the right number of fixed intervals can only be done by trail and error. Generally, more than 1000000 events should be a good starting value, provided that the time-scale separation between reactions within the network is limited.

Case Study 1: Molecule synthesis and degradation

In this section, we re-use the "synthesis-degradation" model described in detail in Chapter 6 to model single-cell transcription. This model consists of a zero-order mRNA synthesis reaction with rate constant k_{syn} and a first-order mRNA degradation reaction with rate constant k_{deg}. In our initial simulations we set k_{syn} to 10 min^{-1} and k_{deg} to 0.2 min^{-1}, so mean mRNA copy number per cell is 50 as is given by Eq. (6.12).

Fig. 7.2A illustrates that the number of mRNA molecules per cell fluctuates around this steady-state value. High mRNA copy numbers correspond to high mRNA degradation propensities (a_2) and vice versa as is shown in Fig. 7.2B. In contrast, the mRNA synthesis propensity (a_1) is constant through time. Distributions can give us more insight into the size of fluctuations. The mRNA copy number is Poisson distributed (see also Chapter 6), which is confirmed by the stochastic simulations shown in Fig. 7.2C. Next, Fig. 7.2D shows that the distribution of a_2 follows a modified Poisson distribution (Eq. (S7.8)). We also performed an auto-correlation analysis of the mRNA time series (Fig. 7.2E); the auto-correlation time decays exponentially as $e^{k_{deg}t}$ in agreement with theory (Eq. (S7.9)).

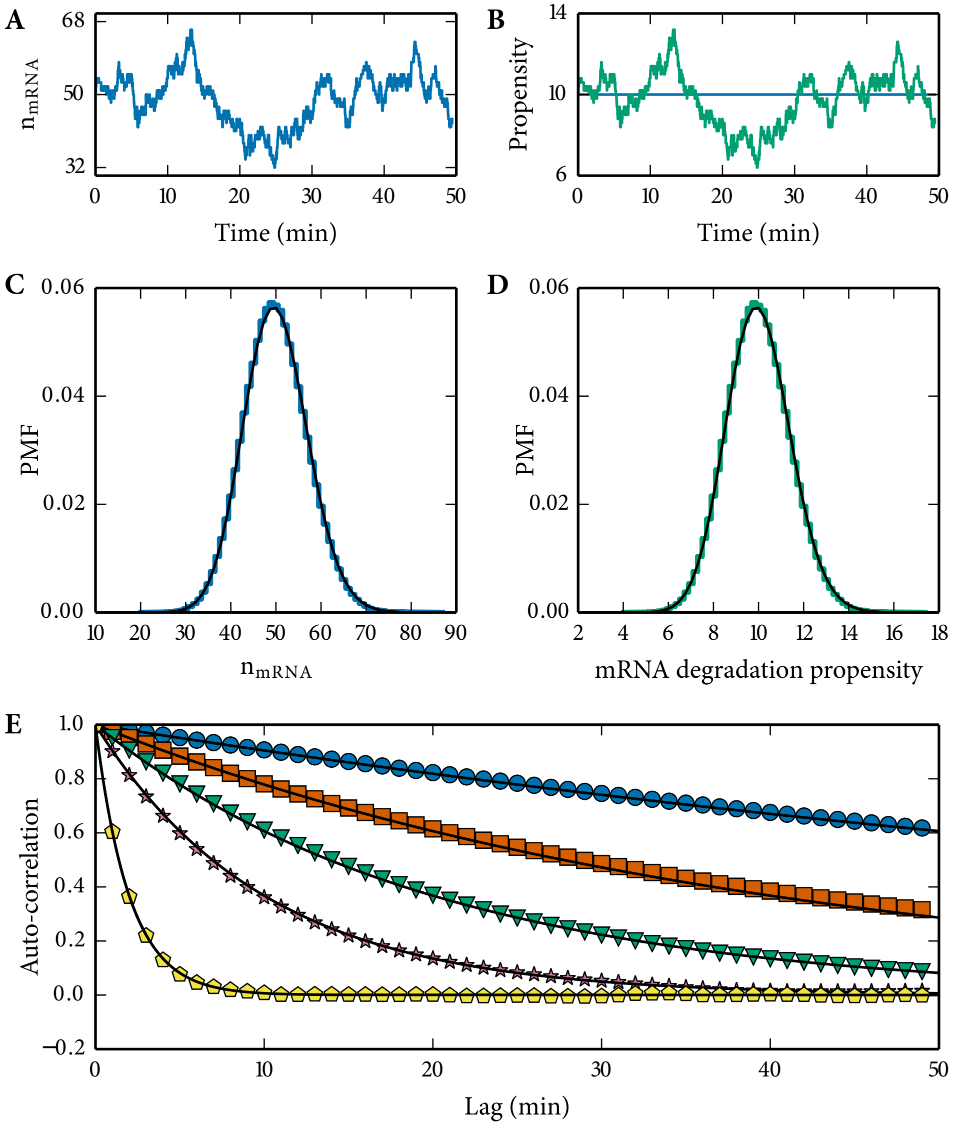


Figure 7.2. Propensities and auto-correlations. *Illustration of several plotting options in StochPy. Colored lines represent StochPy output and black the analytical solutions. (A) species time-series data. (B) propensities time-series data. (C) species distribution. (D) propensities distribution. (E) auto-correlation for different* k_{deg} *values (0.01, 0.025, 0.05, 0.1, 0.5).*

Case Study 2: Stochastic mRNA synthesis by a switching gene

In this section, we consider a model of mRNA synthesis by a gene that switches spontaneously between an inactive (OFF) state and an active (ON) state. The synthesis of mRNA occurs only during the ON state whilst mRNA degradation occurs continuously. Depending on the choice of kinetic parameters, this system can display transcription bursts that can cause significant cell-to-cell variability in mRNA expression levels (Shahrezaei and Swain, 2008; Dobrzynski and Bruggeman, 2009).

We set the mRNA synthesis and degradation rate constants to 80 min^{-1} and 2.5 min^{-1} in our simulations respectively. Switching from ON to OFF state and vice versa occurred at a rate of 0.05 min^{-1} during bursty transcription and at a rate of 5.0 min^{-1} during non-bursty transcription, respectively. Lifetimes of the ON and OFF state ($1/k_{off}$ and $1/k_{on}$) were, therefore, hundred times greater during bursty transcription than during non-bursty transcription.

In Fig. 7.3 we compare the time evolution for the two kinetic parameter settings leading to a bursty and a non-bursty mode of transcription. Long lifetimes of both the ON and OFF state cause bursty transcription (Fig. 7.3A–B), while short lifetimes of both the ON and OFF state cause non-bursty transcription (Fig. 7.3C–D).

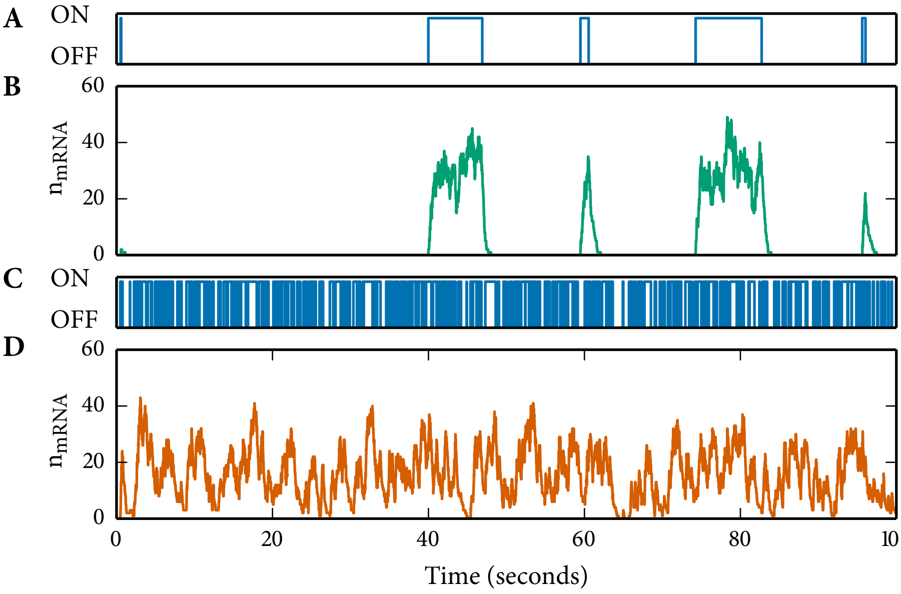


Figure 7.3. Time series of bursty and non-bursty transcription. *StochPy plots of simulating stochastic gene expression. (A) long lifetimes of both the ON and OFF state. (B) bursty transcription. (C) short lifetimes of both the ON and OFF state. (D) non-bursty transcription.*

The mean mRNA copy number per cell and the fraction of time spent in the ON and OFF state are given by

$$\begin{aligned} \langle n_{mRNA} \rangle &= \frac{k_{syn}}{k_{deg}} \cdot \frac{k_{on}}{k_{on} + k_{off}}, \\ \langle n_{ON} \rangle &= \frac{k_{on}}{k_{on} + k_{off}}, \\ \langle n_{OFF} \rangle &= \frac{k_{off}}{k_{on} + k_{off}}. \end{aligned} \tag{7.1}$$

Because both switching probabilities are equal, the fraction of time spent in both states should be equal. Therefore, the mean mRNA copy number per cell during bursty and non-bursty transcription is equal. Based on our parameters the mean mRNA copy number per cell should be equal to 16. The standard deviation of the mRNA copy number, however, differs between bursty and non-bursty transcription, which can be calculated with

$$\sigma_{n_X} = \sqrt{\langle n_X^2 \rangle - \langle n_X \rangle^2}. \tag{7.2}$$

StochPy simulations gave that $\langle mRNA \rangle$ was about 16.0 copy numbers per cell for bursty and non-bursty transcription respectively. The fraction of time spent in the ON and OFF state — $\langle n_{OFF} \rangle$ and $\langle n_{ON} \rangle$ — for both bursty and non-bursty transcription was about 0.50. For a single StochPy simulation of 1000000 steps, standard deviations were $\sigma_{n_{mRNA}} \approx 16.20$ and $\sigma_{n_{off}} = \sigma_{n_{on}} \approx 0.50$ for bursty transcription and $\sigma_{n_{mRNA}} \approx 8.21$ and $\sigma_{n_{off}} = \sigma_{n_{on}} \approx 0.50$ for non-bursty transcription.

Transcription bursts lead to bimodal mRNA copy number distributions (Fig. 7.4A) and two time scales in the distribution of event waiting times of mRNA synthesis times (Fig. 7.4B). These event waiting times depend on both the production during a ON state and the periods of inactivity (Dobrzynski and Bruggeman, 2009). For both bursty and non-bursty transcription we determined the mean waiting times of mRNA synthesis at 0.25 and the mRNA noise at 1.02 and 0.26 respectively. Each of these results are—for the parameter values used—in agreement with the analytical solutions of the mean waiting times of mRNA synthesis (Eq. (S7.12)) and mRNA copy number noise (Eq. (S7.13))

Case Study 3: Spontaneous fluctuations in single-molecule enzyme activity

Next, we consider a completely different model that describes single-molecule enzymology. Michaelis-Menten kinetics have been used already for one-hundred years for modeling enzyme kinetics for a large ensemble of enzyme molecules. The simplest enzyme mechanism considers an enzyme, E, which converts a single substrate S into a single product P by first forming an enzyme-substrate complex ES. An illustration of this system is shown in

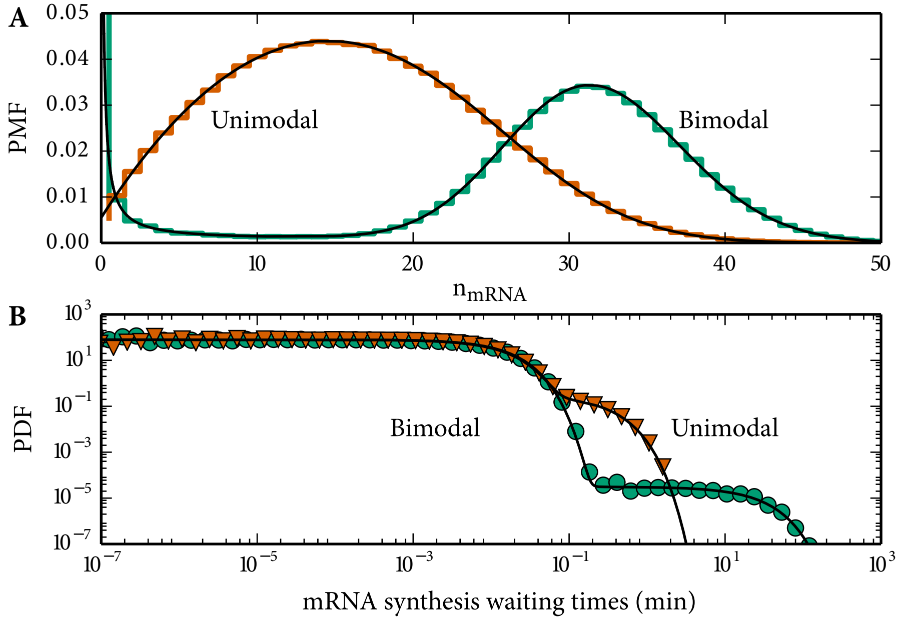


Figure 7.4. mRNA copy number and event waiting times distributions. *StochPy plots of simulating stochastic gene expression with StochPy simulations (step, markers, colored) and analytical solutions (solid, black). (A) probability mass function of the mRNA copy numbers. (B) probability density function of the mRNA synthesis event waiting times. Each of these results is in agreement with the corresponding analytical solution given in Eq.* (S7.10) *and Eq.* (S7.11).

Fig. 7.5A. Recent advances in single-molecule measurements allow the study of enzymatic reactions at the level of single enzymes. Stochastic simulations are required to study single-molecule enzymology. By performing single-molecule experiments the probability density, $f_T(t)$, of event waiting times T for product P arrivals can be determined (Kou et al., 2005; Qian, 2008).

Here, we stochastically modeled single-molecule enzyme kinetics to demonstrate the capabilities of StochPy. Single-molecule enzyme kinetics means that, at all times, the total enzyme copy number $n_E[t] + n_{ES}[t] = 1$ (Figs. 7.5B–C). Because the substrate depletion with one enzyme copy number is negligible, the substrate S copy number was considered constant, $n_S[t] = n_S = c$. Subsequently, we can write the rate equations of this system in terms of time dependent probabilities (Kou et al., 2005),

$$\begin{aligned}\dot{P}_E(t) &= -k_1^+ \cdot P_E(t) + k_1^- \cdot P_{ES}(t),\\ \dot{P}_{ES}(t) &= k_1^+ \cdot P_E(t) - (k_1^- + k_2^+) \cdot P_{ES}(t),\\ \dot{P}_P(t) &= k_2^+ \cdot P_{ES}(t).\end{aligned} \tag{7.3}$$

The probability for a single enzyme to be in the unbounded state is the fraction of time spent in the unbounded state. Similarly, the probability for a single enzyme to be in the bounded state is the fraction of time spent in the bounded state. We can write these probabilities in terms of rate constants and substrate copy numbers,

$$\begin{aligned} P_E &= \frac{\tau_E}{\tau_E + \tau_{ES}} = \frac{k_1^- + k_2^+}{k_1^- + k_2^+ + k_1^+ \cdot n_S}, \\ P_{ES} &= \frac{\tau_{ES}}{\tau_E + \tau_{ES}} = \frac{k_1^+ \cdot S}{k_1^- + k_2^+ + k_1^+ \cdot n_S}. \end{aligned} \tag{7.4}$$

These probabilities equal the analytical $\langle E \rangle$ and $\langle ES \rangle$ copy numbers in steady state. StochPy simulations gave that $\langle n_E \rangle \approx 0.555$ and $\langle n_{ES} \rangle \approx 0.445$ which is in agreement with analytical values for the parameter values used ($n_S \cdot k_1^+ = 2/3$, $k_1^- = 1/12$, and $k_2^+ = 3/4$).

Next, the steady-state flux of this system is given by

$$V_{ss} = k_1^+ \cdot n_S \cdot P_E - k_1^- \cdot P_{ES} = k_2^+ \cdot P_{ES}. \tag{7.5}$$

From this relationship we can derive the famous Michaelis-Menten relationship:

$$\begin{aligned} V_{ss} = k_2^+ \cdot P_{ES} &= \frac{k_2^+ \cdot k_1^+ \cdot n_S}{k_1^- + k_2^+ + k_1^+ \cdot n_S}, \\ &= \frac{k_2^+ \cdot n_S}{\frac{k_1^- + k_2^+}{k_1^+} + n_S}, \\ &= \frac{V_{max} \cdot n_S}{K_m + n_S}, \end{aligned} \tag{7.6}$$

where $V_{max} = k_2^+$ and $K_m = \frac{k_1^- + k_2^+}{k_1^+}$. For large ensembles n_P at time t depends then on

$$P(t) = V_{ss} \cdot t. \tag{7.7}$$

For single-molecule enzymology experiments the product P copy number (n_P) fluctuates around this analytical solution for large ensembles. In Fig. 7.5D three Monte Carlo trajectories that describe the time evolution of this model are shown to illustrate these fluctuations for small ensembles.

The probability density of event waiting times for product P arrivals is given by

$$f_T(t) = k_2^+ \cdot P_{ES}(t), \tag{7.8}$$

which we solved analytically using Mathematica (Eq. (S7.14)). Our simulations confirm that—for single-molecule enzymology—$f_T(t)$ peaks as is shown in Fig. 7.5E. The inverse of the classic Michaelis-Menten equation is the first raw moment of $f_T(t)$—the mean waiting time $\langle T \rangle$ of product P arrivals (Eq. (S7.16)). Theoretically, $\langle T \rangle = 3.0$ and $V_{ss} = 1/3$ given the set of used parameters. StochPy simulations gave that $\langle T \rangle$ was about 3.0.

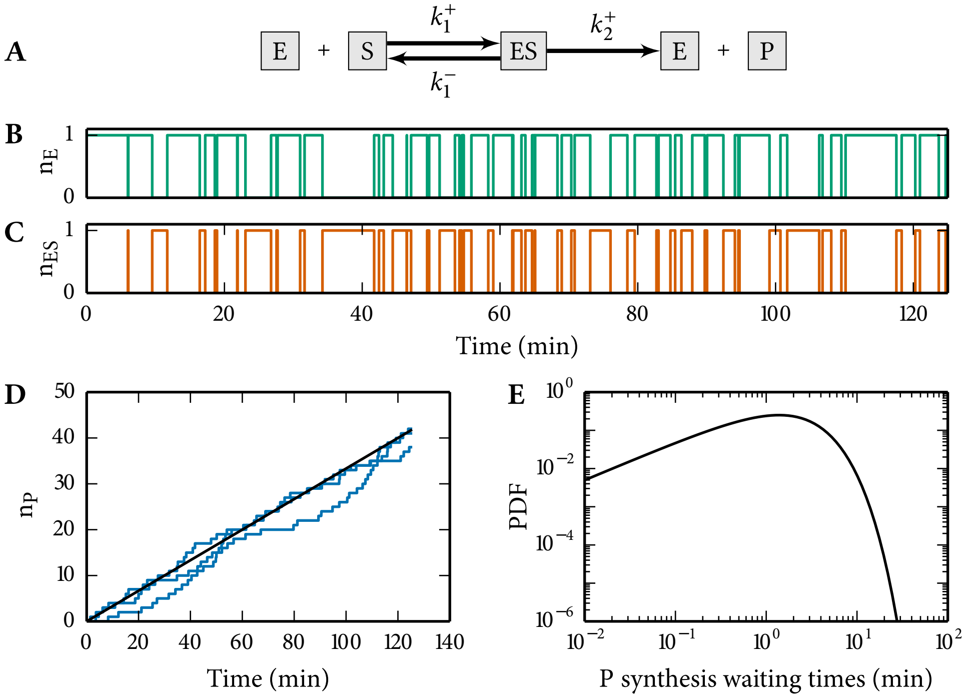


Figure 7.5. Single-molecule enzymology. *(A) the classic enzyme kinetic scheme where* k_1^+, k_1^-, *and* k_2^+ *denote rate constants. (B-C) time-series StochPy plot of the E and ES copy numbers. (D) three stochastic time trajectories (blue) of the P copy number fluctuate around its analytical solution (black). (E) event waiting times of product P formation peak and match the analytical solution (black).*

Case Study 4: Modeling and simulation of explicit and implicit cell growth and division

Each living cell grows, doubles its content, and subsequently divides which has an effect on gene expression dynamics. To accurately describe gene expression dynamics, we therefore need to take into account the effects of cell growth and division. To demonstrate the flexibility of StochPy we briefly illustrate how simple it is to extend a stochastic model of a gene expression network with cell growth and (explicit) cell division events, even though this was not (yet) a standard functionality of StochPy. This model is illustrated in Fig 7.6A and consists besides mRNA and protein, also of active and inactive transcription factors (TF). These transcription factors switch between an inactive and an active state where the active state stimulates mRNA synthesis. Protein synthesis can occur if mRNA is present in the cell. This results in a total of nine reactions where one reaction is not described by mass-action kinetics, which would make this system already hard to simulate for some software packages.

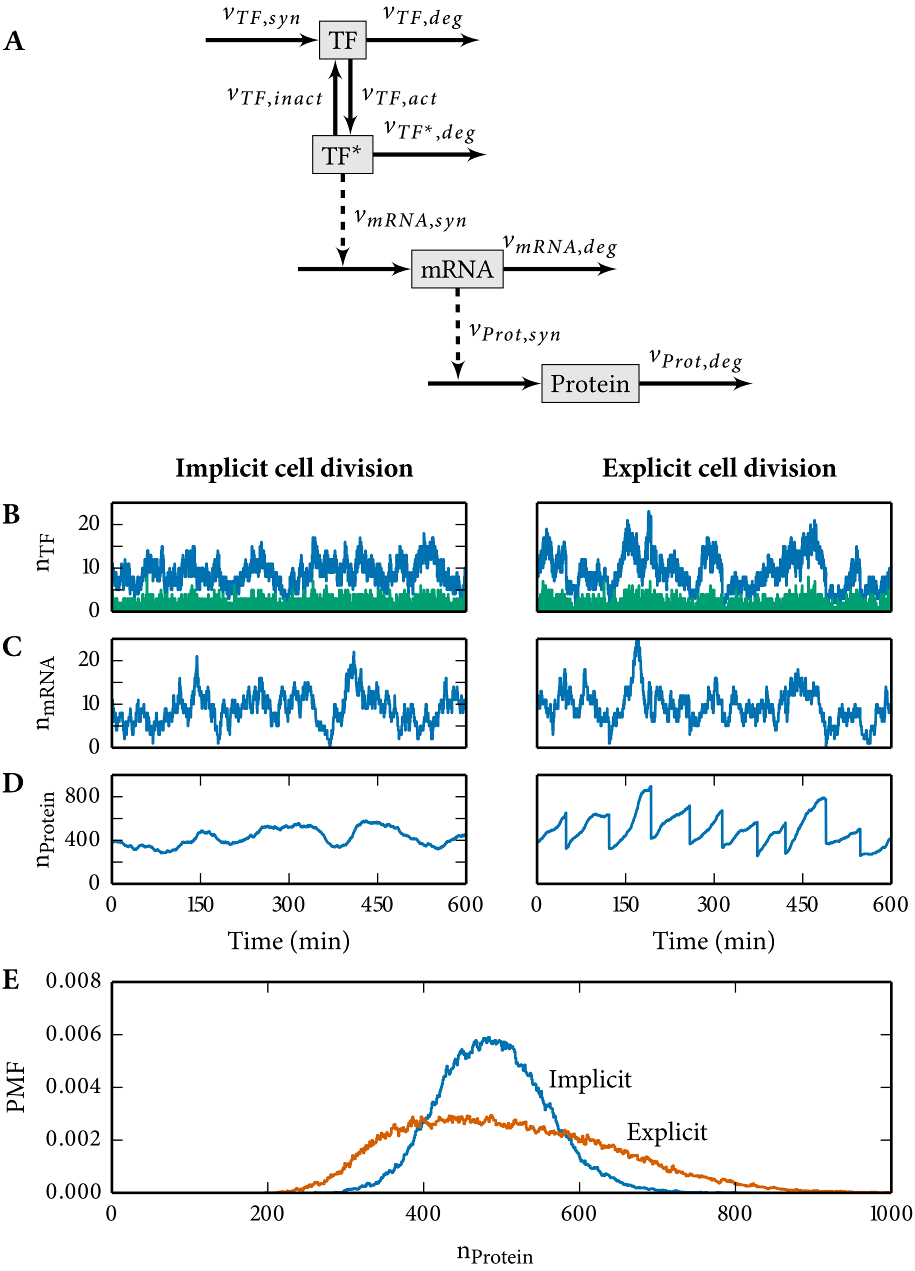


Figure 7.6. Modeling and simulation of single-cell transcription and translation with and without explicit cell growth and division. *The used stochastic model consists of transcription factors, mRNA, and protein (panel A). Time series data of transcription factor copy numbers (panel B), mRNA copy numbers (panel C), and protein copy numbers (panel D). Distributions of protein copy numbers (panel E). The time between two cell division events was gamma-distributed with scale=60.0 and shape= 1.0. These settings result in a doubling time of on average 60 minutes.*

M&S of cell growth and division can be done both explicitly and implicitly. In the explicit approach the cell content is binomially distributed at cell division between two daughter cells after a certain generation time T_g. We used a Γ distribution (scale = 60, shape = 1) to draw the generation time, but on average the cellular content was doubled after $T_g = 60$ minutes. This is in contrast with earlier work from Kierzek et al. (2001) who used a fixed generation time and divided the molecular content into two at cell division. Commonly used model definition formats (e.g. as encodable in SBML) do not support cell division events, which makes a sequential simulation approach a necessity. Population averages also depend on the age distribution in the population. We assumed a constant age distribution (a synchronous population). This allowed us to track only one daughter cell after each cell division rather than tracking the complete phylogenetic tree.

In such a scenario, each simulation starts with the results of the previous simulation. For these reasons, M&S of cell explicit division is inconvenient and non-trivial, especially for graphical user interface (GUI) based simulators. In the implicit approach the specific growth rate is incorporated in the model as a first-order rate constant that continuously dilutes cellular components into new cells. We used the following relationship to determine the specific growth rate (μ):

$$T_g = \ln(2)/\mu, \tag{7.9}$$

which was thus set to approximately 0.012 min^{-1}.

We used StochPy to generate stochastic data for 5000 generations, about 10000000 time steps, which allowed us to accurately determine the mean copy numbers of transcription factors, mRNA, and protein (Table 7.2). Previous research showed that the average copy numbers with a constant age distribution were about 4% greater than with an exponential age distribution (Gomez et al., 2014). The results of our StochPy simulations are in agreement with this finding. We found that explicit cell growth and division yields greater standard deviations, especially for protein copy numbers.

	$\langle n_{TF} \rangle$	$\sigma_{n_{TF}}$	$\langle n_{TF*} \rangle$	$\sigma_{n_{TF*}}$
Explicit M&S	1.04 ± 0.00	1.03 ± 0.00	9.26 ± 0.03	3.39 ± 0.01
Implicit M&S	1.01 ± 0.00	1.01 ± 0.00	9.01 ± 0.03	3.02 ± 0.01
	$\langle n_{mRNA} \rangle$	$\sigma_{n_{mRNA}}$	$\langle n_{protein} \rangle$	$\sigma_{n_{protein}}$
Explicit M&S	9.99 ± 0.02	3.53 ± 0.01	508.13 ± 1.62	132.96 ± 0.59
Implicit M&S	9.79 ± 0.02	3.19 ± 0.01	489.35 ± 0.88	70.18 ± 0.47

Table 7.2. Mean and standard deviation of TF, TF*, mRNA, and protein obtained with explicit and implicit modeling and simulation of cell growth and division. *The error represents 1 SD and is based on ten simulations.*

The differences between M&S of explicit and implicit cell growth and division are further illustrated in Fig. 7.6. The dynamics of transcription factors, mRNA, and proteins are different (Figs. 7.6B-D). Most apparent are the differences for the protein copy numbers (Fig. 7.6D); in the implicit approach they quickly reached a steady state whereas in the explicit approach large variations in protein copy numbers were observed. The distribution of mean protein copy numbers was, therefore, significantly different between the implicit and explicit approach (Fig. 7.6E). These differences are also observable for transcription factors (active, inactive) and mRNA (but not shown).

Benchmarking StochPy

The StochPy software features and speed performance were benchmarked against widely used, existing stochastic software to make a fair and broad comparison of available tools for stochastic simulations.

Feature comparison with existing stochastic tools

In an extensive search for available stochastic simulators, CAIN, COPASI, Facile-EasyStoch, GillespieSSA, and StochKit2 were identified as those with the closest functionality to StochPy. Here, we discuss and compare these tools against StochPy through a feature comparison. A comprehensive feature comparison is provided in Table 7.3.

(i) Implemented solvers. StochPy offers besides the traditional SSAs—which assume that reaction occur instantly and in a constant volume—also SSAs with delay and a SSA with cell growth and division. The latter is the main topic of Chapter 8.

(i) Explicit stochastic simulation output. An important distinction between StochPy and most other stochastic simulators is in their output: Explicit rather than fixed-interval output is used. This has several consequences. First, fixed-interval output does not allow the calculation of event waiting times, a unique feature provided by StochPy. Second, handling of fixed-interval output requires expert knowledge of the modeler (how many fixed intervals are sufficient to do a certain type of analysis properly). And third, determining the minimal number of fixed intervals necessary is time-consuming. In the StochPy software the modeler can exploit the convergence of higher moments to determine the minimal number of time steps necessary to accurately determine certain model properties. However, the benefit of using fixed-interval output is that less data can be stored which gives this approach a significant speed advantage. This can be useful if, for instance, only moments and time series are of interest.

(ii) SBML support and simulating diverse stochastic models. Neither Facile-EasyStoch nor GillespieSSA provide SBML support while CAIN and StochKit2 support only a subset of available SBML models. As a consequence, many models (e.g. those with time and particle-number events or complicated rate laws) cannot be simulated, which limits their general

purpose simulation capabilities. In contrast, both COPASI and StochPy support SBML levels 1–2 (V4), but COPASI does not support (SBML) events in stochastic simulations.

	CAIN	COPASI	EasyStoch	GillespieSSA	StochKit2	StochPy
Implemented solvers:						
- Exact SSA	●	●	●	●	●	●
- Inexact SSA	●	●		●	●	●
- Delayed SSA						●
- Growth & division SSA						●
Simulator options:						
- SBML support	○[i]	●			○[ii]	●
- Human inter. input			●			●
- Stochastic test suite						●
- Extrinsic noise			●			
- Explicit output			●	●	○[iii]	●
- Fixed-interval output	●	●	●		●	●
Output analysis:						
- Auto-correlations						●
- Histogram distance	●				●	
- Propensities		●				●
- Moments						●
- Waiting times						●
Software Design:						
- Plotting facilities	●	●		●	○[iv]	●
- Data exportation	●	●	●		●	●
- GUI	●	●				
- Flexible environment				●		●
- Open source	●	●	●	●	●	●

Table 7.3. Feature comparison between StochPy and existing (stochastic) software. *Summary of features offered in StochPy and other stochastic M&S software. In the fourth case study we illustrated how cell growth and division can be added to the SSA. In Chapter 8, we present a generic SSA with growth and division of which the outcomes are in exact agreement with theory.*

●*: Feature is present.*

○*: Feature is partially present or requires additional dependencies.*

Notes:

i. Limited ability to parse kinetic laws: Complicated expressions may not parsed.

ii. Not all SBML documents can be converted into the StochKit2 model format.

iii. Provided as an add-on functionality of StochKit2, whereas with limited options compared to the default installation of StochKit2.

iv. Only if proprietary software (MATLAB) is installed.

(iii) Analysis of stochastic data. Besides various (unique) numerical analysis techniques StochPy also provides preprogrammed plotting functions. Calculation of event waiting times, propensity probability distributions, (co-)variances, and auto-correlations for one or more generated trajectories are currently unique numerical analysis techniques of StochPy. In addition, analysis of time series of species and propensities, probability distributions of species copy numbers, and moments can be done within StochPy. While not shown in the case studies, StochPy can calculate and visualize the average of multiple time trajectories for time series, auto-correlations, and distributions.

(iv) Flexible environment for interactive modeling and simulation. Due to its flexible design StochPy's functionality can be extended far beyond saving data files and preprogrammed plotting capabilities. By integrating its functionality with industrial strength Python scientific libraries (e.g. Matplotlib (Hunter, 2007), NumPy (Oliphant, 2006) and SciPy (Jones et al., 2001)) sophisticated user-defined analysis methods can be seamlessly applied to the output of StochPy simulations.

Direct solver performance

We benchmarked the direct solver of StochPy against the direct solvers of two widely used and high-performance stochastic tools: CAIN and StochKit2 (both implemented in C++). The results of this benchmark are shown in Table 7.4. We specifically choose CAIN and StochKit2, because those are the fastest solvers currently available. To fairly compare StochPy's performance against these other tools, we set the number of fixed-intervals equal to number of time steps in the stochastic simulation. Determining the minimal number of fixed-intervals necessary to perform a particular analysis requires doing multiple simulations (as shown in Fig. 7.1), which makes this fixed-interval approach less efficient and more time-consuming than Table 7.4 reports.

The first conclusion from this benchmark is that no single solver was the fastest in any case. Second, StochPy is the only stochastic simulator that was able to correctly simulate all stochastic models tested in this benchmark. This in contrast to the CAIN API which can accept models consisting of only mass-action kinetics. Third, for different numbers of simulation time steps we found significant differences in simulation time between StochPy, CAIN, and StochKit2 simulations. For relatively short simulations, StochKit2's performance is reduced, as StochKit2 requires substantial time to compile models with events and non mass-action kinetics. This effect is negligible for relatively long simulations. Both the CAIN and StochKit2 solvers outperformed StochPy's direct solver for most tested models if we generated a relatively large number of time steps (except e.g. the simulation of events with CAIN). And fourth, StochPy's performance increased with respect to the performance of both CAIN and StochKit2 when we considered larger models. For instance, CAIN requires about four minutes to parse the largest model tested (parsing time was omitted from the

	CAIN	CAIN (API)	StochKit2	*StochPy*^{i}
Small model	0.7 – 0.10	0.24 – 0.10	0.24 – 0.07	1.0 – 0.31
Non mass-action	0.5 – 0.10	N/A	96 – 0.16	1.0 – 0.45
Parallel	0.04 – 0.07	0.04 –0.06	0.03 – 0.18	0.28 – 0.33
Parallel & time events	1.9 – 1.9^{ii}	N/A	1.7 – 0.18	1.0 – 0.3
Parallel & particle number events	3.2 – 3.7^{ii}	N/A	1.5 – 0.18	1.0 – 0.3
Assignments	N/A	N/A	N/A	1.0 – 1.0
Large model	0.14	0.28	0.09	0.56
XL model	0.15	0.31	0.11	0.66
XXL model	0.24	0.37	0.11	0.93

Table 7.4. Speed performance benchmark between StochPy and existing (stochastic) software. *Results of benchmarking the direct method of StochPy. We divided the simulation time of the respective solver by the simulation time of the StochPy solver: StochPy's solver is faster if the reported ratio's are greater than one and vice versa. A "–" indicates that we performed short and long simulations to illustrate the potential difference between them. N/A is shown if the simulator was not able to perform the simulation. We generated 100 trajectories for the parallel simulations. In each comparison the number of fixed intervals was equal to the number of time steps in the simulation. We performed the simulations on an Intel Core i5-2430M CPU 2.40 GHz×4 64-bit with Ubuntu 12.04 LTS as operating system. Stochastic models and a script to simulate these models within StochPy are available in Scripts S2.*
Notes:
i. StochPy with interfaces to CAIN and StochKit2. Simulation time includes time to parse results into StochPy.
ii. Cain cannot parse events, so the user most specify them in the GUI.

benchmark), while both StochKit2 and StochPy are able to parse this model within seconds. For relatively long simulations of models with many species, StochKit2's solvers are about ten times faster than those of StochPy, which is expected because our software is written in Python rather than C++.

Since, we also offer access to CAIN and StochKit2 solvers directly from StochPy, we also tested the performance of CAIN and StochKit2 for this mode of operation. While exploiting these solvers in StochPy appears slower than the native application, this time only includes parsing of the simulation output for post-simulation analysis. This can take a significant amount of time for big data sets.

As StochPy provides access to multiple SSAs, SSA implementations, and simulation tools and as discussed above, there is no 'one size fits all' approach, we provide a decision tree to help guide prospective modelers in how best to select a method that suits their model (see Fig. 7.7). Here, decisions are made depending on the simulation time and the output of the solver. Insights into time series or moments can be easily obtained with solvers that provide fixed-interval output, whereas solvers that provide explicit output are, in principle, better suited for determining probability distributions of molecule copy numbers and event waiting times.

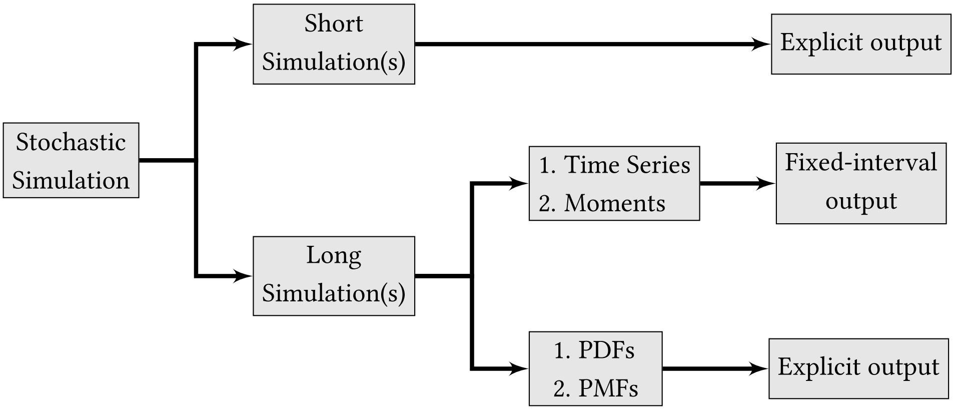


Figure 7.7. Stochastic modeling and simulation decision tree. *Both fixed-interval and explicit output have their advantages and disadvantages. The decision whether to use fixed-interval or explicit output depends on the type of analysis.*

7.3 Conclusions

Stochastic modeling and simulation in systems biology demands a certain level of flexibility in simulation, management of stochastic models and the handling of simulation data. Depending on the size of the system of interest and its degrees of time-scale separation, the different SSAs each have their particular (dis-)advantages. The differences in simulation time between stochastic simulation packages are often due to the fixed-interval reporting of simulation data versus the use of explicit output. To achieve the accuracy of explicit solvers the differences in simulation time greatly reduce, and ultimately boil down to, differences in the programming languages. In systems biology applications, often the pure simulation data rather than the fixed-interval simulation data is of interest. The pure simulation data allows for the accurate determination of various time and copy number associated probability measures.

We presented StochPy as a versatile modeling and simulation package for stochastic simulation of molecular control networks inside living cells that provides solvers which return explicit stochastic simulation output. Its integration with Python's scientific libraries and PySCeS makes it an easily extendible and a user-friendly stochastic simulator package. We highlighted this by implementing both the solvers of CAIN and StochKit2 that return only fixed-interval output, which can be useful for obtaining insight into time series and moments. The high-level statistical and plotting functions of StochPy allow for quick and interactive model interrogation at the command-line. Python's scripting capabilities allow for more complicated and in-depth analysis of stochastic models and meets many of the demands for systems biology.

7.4 Supplemental Information

The following materials are available at http://persistent-identifier.org/?identifier=urn:nbn:nl:ui:18-23542 and in the online version of this article on which this chapter is based on.

Dataset S1. The results of testing StochPy against the SBML stochastic test suite (doi:10.1371/journal.pone.0079345.s001).

Scripts S1. Annotated scripts and input files used to the generate M&S results (doi:10.1371/journal.pone.0079345.s002).

Scripts S2. Stochastic models and a script to simulate the models used for benchmarking StochPy (doi:10.1371/journal.pone.0079345.s003).

This supplemental information further contains an explanation of analyzing stochastic simulation output and additional analytical results. We set for convenience in each model the cell volume to 1 (cell volume of *Escherichia coli* is about 1 fl). Within StochPy simulations were done with 2000000 time steps unless stated otherwise.

Analyzing stochastic simulation output

The stochastic simulation output is not discretized and, as a consequence, StochPy cannot calculate the mean of a particular species by taking the average over the vector of species copy numbers. We first have to calculate the species state probability—the probability of species with index i in a given state n. StochPy does this by determining the fraction of time spent in this particular state,

$$P(n_i = n) = \frac{\sum_{s=1}^{S} (T_s | n_i = n)}{T_{sim}}. \tag{S7.1}$$

Here T_s is a time period that species n_i spent in state n, and T_{sim} is the total simulation time. For the species n_1 shown in Table 7.1, as first sight, one might think that $P(n_1 = 0) << P(n_1 = 1)$ because $n_1 = 1$ during 19 of the 20 simulated time steps. In stochastic simulations, reaction events are irregular due to their stochastic nature. In this particular example, the first reaction event took 39.976 time units, while the next 19 reaction events together took only 1.002 time units. Therefore, the probabilities of being in state zero and one are $P(n_1 = 0) = 39.976/40.978 \approx 0.975$ and $P(n_1 = 1) = 1.002/40.978 \approx 0.025$ and therefore $P(n_1 = 0) >> P(n_1 = 1)$.

Subsequently, StochPy determines the mean copy number of this species by taking the sum of the product of species state probabilities and species state values for each n,

$$\langle n_i \rangle = \sum_{n \in N} P(n_i = n) \cdot n. \tag{S7.2}$$

Since we already calculated the species state probabilities of species n_1, we can simply determine its mean for the done simulation, $\langle n_1 \rangle = P(n_1 = 0) \cdot 0 + P(n_1 = 1) \cdot 1 \approx 0.025$.

Both Eqs. (S7.1) and (S7.2) can also be exploited for calculating propensity probabilities and means. Then, a_j is the propensity of reaction index j and $P(a_j = n)$ is the propensity probability of reaction index j for state n. For instance, $P(a_1 = 0) = 39.976/40.978 \approx 0.975$ and $P(a_1 = 0.05) \approx 0.025$ and therefore $\langle a_1 \rangle = P(a_1 = 0) \cdot 0 + P(a_1 = 0.05) \cdot 0.05 \approx 1.25 \cdot 10^{-3}$.

Returning explicit output allows StochPy to provide the unique feature of the calculation and analysis of event waiting times. The event waiting times for a particular reaction is a vector of inter-event times of that particular reaction. For a given reaction with index j these waiting times can be calculated by determining all its inter-event times,

$$\boldsymbol{w}(j) = \forall\ e : (t_{e+1}|j) - (t_e|j). \tag{S7.3}$$

Here $\boldsymbol{w}(j)$ are the event waiting times of reaction index j and t_e is an event time.

StochPy also provides the unique feature of auto-covariance and auto-correlation analysis of species and propensities. Covariance is a measure of two random variables of their joint variability. Therefore, auto-covariance is the covariance of a variable against a time-shifted version of itself. We can subsequently determine the auto-correlation by dividing the auto-covariance by the variance of the signal. The (auto-)correlation is thus a dimensionless property. This makes the (auto-)correlation often a more useful measure of correlation than (auto-)covariance. For our explicit stochastic output (of which an example is shown in Table 7.1), the upper limit of the number of time lags is given by

$$\frac{L!}{2!(L-2)!} = 0.5L \cdot (L-1), \tag{S7.4}$$

where L is the number of simulation time points. That is, the upper limit is the number of 2-combinations when we ignore the order. To calculate both auto-covariances and auto-correlations we thus transfer the explicit output to a regular grid with sample interval δt.

We can now calculate both the (sample) auto-covariance and the auto-correlation by determining the signal difference for any arbitrary time lag τ. This τ equals $l\delta t$ where l is an integer between 0, 1, ..., L. We illustrate this for species n_1 (the same method can be used for propensities). The auto-covariance of species n_1 for time lag τ cam be determined with

$$acov(\tau) = \frac{1}{L} \cdot \sum_{t=0}^{L-\tau} (n_1(t) - \bar{n}_1)(n_1(t+\tau) - \bar{n}_1), \tag{S7.5}$$

where $n_1(t)$ is the species state at sample time t and $\bar{n}_1$ is the sample mean. Accordingly, the auto-correlation of n_1 for a given time lag τ can be determined by dividing $acov(\tau)$ with the signal sample variance,

$$acor(\tau) = \frac{acov(\tau)}{\frac{1}{L} \cdot \sum_{t=0}^{L} (n_1(t) - \bar{n}_1)^2}. \tag{S7.6}$$

Analytical solutions: Molecule synthesis and degradation

The mRNA copy number is Poisson distributed (Chapter 6): By multiplying the mean and variance with a constant (k_{deg}) and the squared of this constant, we can calculate the mean and variance of a_2 as

$$\begin{aligned} E(a_2) &= k_{deg} \cdot E(mRNA), \\ Var(a_2) &= k_{deg}^2 \cdot Var(mRNA). \end{aligned} \tag{S7.7}$$

Hence, the mean and variance of a_2 are ten and two respectively. The simulation results of StochPy are in agreement with these analytical results.

Within StochPy, we can also determine the probability of a certain propensity. The probability mass function of propensity a_2 is given by

$$\begin{aligned} P(a_2 = y) &= P(\text{mRNA} \cdot k_{deg} = y) = P(\text{mRNA} = \frac{y}{k_{deg}}), \\ &= f(y,\lambda) = \frac{\lambda^{y/k_{deg}} \cdot e^{-\lambda}}{(y/k_{deg})!}, \qquad \forall\ \frac{y}{k_{deg}} \in n, \end{aligned} \tag{S7.8}$$

and otherwise zero. This is, in fact, a modified Poisson distribution obtained by distorting the x-axis.

For this model the auto-correlation function is also analytically known (Gardiner, 1997),

$$acor(\tau) = e^{-k_{deg} \cdot \tau}, \tag{S7.9}$$

which shows that the auto-correlation of mRNA copy numbers does not depend on its synthesis rate.

Analytical solutions: Stochastic mRNA synthesis by a switching gene

The exact probability of having n mRNA copy numbers at steady state was determined by Shahrezaei and Swain (2008) and is given by

$$P_n(n) = \frac{n_s^n e^{-n_s}}{n!} \frac{\Gamma(\zeta_0 + n)\Gamma(\zeta_0 + \zeta_1)}{\Gamma(\zeta_0 + \zeta_1 + n) + \Gamma(\zeta_0)} 1F_1(\zeta_1, \zeta_0 + \zeta_1 + n; n_s), \tag{S7.10}$$

where Γ denotes the gamma function, $n_s = k_{syn}/k_{deg}$, $\zeta_0 = k_{on}/k_{off}$, $\zeta_1 = k_{off}/k_{deg}$, and $1F_1(a,b;z)$ is the Kummer confluent hyper-geometric function of the first kind.

Dobrzynski and Bruggeman (2009) determined the waiting time probability density function, which is given by

$$\begin{aligned} f(t) &= P[X \in (t, t+dt)]/dt = w_1 r_1 e^{-r_1 t} + w_2 r_2, \\ r_{1,2} &= (K \pm \sqrt{K^2 - 4k_{syn}k_{on}}), r_1 > r_2, \end{aligned} \tag{S7.11}$$

where $w_1 = 1 - w_2 = (k_{syn} - r_2)/(r_1 - r_2) \in (0,1)$, $K = k_{off} + k_{on} + k_{syn}$, and stochastic variable X as the waiting time with value t. The analytical mean waiting time for the mRNA synthesis is given by

$$\langle t_{mRNA} \rangle = \int_0^{\infty} t \cdot f(t) dt = \frac{w_1}{r_1} + \frac{w_2}{r_2} = \tau_{syn} \left(\frac{\tau_{on}}{\tau_{on} + \tau_{off}} \right)^{-1} \tag{S7.12}$$

and the noise in the mRNA copy number is equal to

$$\frac{\sigma^2_{n_{mRNA}}}{\langle n_{mRNA} \rangle^2} = \frac{1}{\langle n_{mRNA} \rangle} + \frac{\tau_o}{\tau_o + \tau_{mRNA}} \cdot \frac{\sigma^2_{n_{ON}}}{\langle n_{ON} \rangle^2} \tag{S7.13}$$

where $\tau_o = 1/(k_{off} + k_{on})$, $\tau_{mRNA} = 1/(k_{deg})$, and $\sigma^2_{n_{ON}} / \langle n_{ON} \rangle^2 = k_{off}/k_{on}$.

Analytical solutions: Spontaneous fluctuations in single-molecule enzyme activity

The rate equations given in Eq. (7.3) represent a system of linear first-order differential equations which can be solved analytically for $P_E(t)$, $P_{ES}(t)$, and $P_P(t)$. Therefore, using Mathematica we can analytically determine $f_T(t)$ as

$$f_T(t) = k_2^+ \cdot \frac{e^{-0.5(k_1^+ \cdot S + k_1^- + k_2^+ + \sqrt{A}) \cdot t} \cdot (-1 + e^{\sqrt{A} \cdot t}) \cdot k_1^+ \cdot S}{\sqrt{A}}, \tag{S7.14}$$

where A depends on the rate constants and the substrate copy number S as

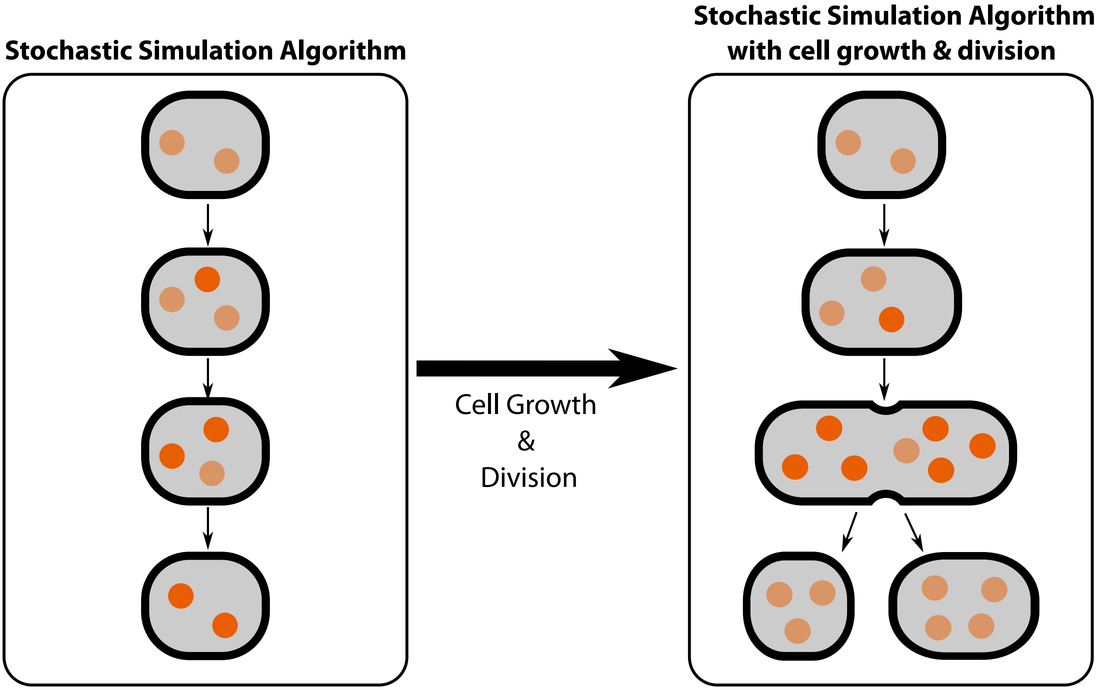
$$A = -4 \cdot k_1^+ + k_2^+ \cdot S + (k_1^+ \cdot S + k_1^- + k_2^+). \tag{S7.15}$$

Rearrangement shows that this analytical result is identical to those obtained by Kou et al. (2005). The first raw moment of $f_T(t)$—the mean waiting time $\langle T \rangle$ of product P arrivals—is the inverse of the classic Michaelis-Menten equation (Eq. (7.6)),

$$\langle T \rangle = \frac{k_1^- + k_2^+}{k_1^+ \cdot k_2^+ \cdot S} + \frac{1}{k_2^+} = \frac{k_1^+ \cdot S + k_1^- + k_2^+}{k_2^+ \cdot k_1^+ \cdot S} = \frac{1}{V_{ss}}. \tag{S7.16}$$

This chapter is based on:

Maarleveld T.R.*, Schwabe A.*, Kempe M., Moinat M., Nordholt N., van Heerden J., Bruggeman F.J. (in preparation) *"Stochastic simulation of single-cell growth and gene expression matches theory and time-lapse microscopy data"*,
⋆ shared first authors


Stochastic Simulation Algorithm
Stochastic Simulation Algorithm with cell growth & division
Cell Growth & Division

Stochastic simulation algorithms (as discussed in Chapters 6 and 7) assume a constant system volume. The effects of cell growth and division can have a significant impact on the system behavior, especially when the timescale over which the system settles to a steady state exceeds the systems generation time.

8 Cell growth and division in stochastic simulation algorithms

Abstract

Cell-to-cell variability arises from the inherent stochasticity of biochemical reactions and of cell growth and division. Fluctuations in metabolic proteins cause fluctuations in cell-volume growth, which in turn affects activities of biochemical processes. Noise in molecule concentrations have a biochemical component and another associated with molecule partitioning during cell division. Theoretical approaches for disentangling those different noise components are limited when dealing with realistic, complicated molecular circuits. Stochastic simulation is therefore an attractive alternative. However, a generic stochastic simulation algorithm, which combines the stochasticity of single-cell growth and molecular processes, is not available. Here we present such an algorithm. Our algorithm is in exact agreement with analytically solvable examples, and in good agreement with time-lapse microscopy data of growing *Escherichia coli* and *Bacilus subtilis* cells, each harboring a fluorescent gene-activity reporter.

8.1 Introduction

Single-cell experiments show that isogenic cells differ notably in their molecular composition (Eldar and Elowitz, 2010; Balazsi et al., 2011; Kempe et al., 2015). This cell-to-cell variability arises from different stochastic phenomena such as the inherent stochasticity of biochemical reactions (Raj and van Oudenaarden, 2008), transcriptional bursting (Golding et al., 2005), and molecule partitioning during cell division (Huh and Paulsson, 2011).

Gillespie's exact stochastic simulation algorithm (SSA) (Gillespie, 1977) has been widely used to simulate the stochastic dynamics of (bio-)chemical systems (Arkin et al., 1998; Zwicker et al., 2010; Wei et al., 2014), as also illustrated in Chapters 6 and 7. These SSAs take into account the stochastic effects of molecule synthesis and degradation (i.e. the inherent stochasticity of biochemical reactions) but assume that these reactions occur instantly and in a constant volume. Biochemical reactions take place in constantly growing and dividing cells, which gives rise to additional sources of non-genetic variability. We can discriminate, amongst others, between sources of stochasticity of (i) biochemical reactions and (ii) cell growth and division (Schwabe and Bruggeman, 2014).

We distinguish the following sources of non-genetic cellular heterogeneity:

(i) stochasticity of net molecule synthesis,

(ii) imprecise partitioning of molecules at cell division,

(iii) heterogeneity in the mother cell volume at cell division, and

(iv) imprecise volume division from mother to daughter cells.

Each of these can have a significant impact on the outcome of stochastic simulations and single-cell experiments, especially when the timescale over which the system settles to a steady state exceeds the cellular generation time.

As discussed in the previous chapter, several different stochastic simulators exist, but none of them incorporates the stochastic contributions of cell growth and division into their algorithms. An initial attempt to model stochastic gene expression in growing cells was done by Gomez et al. (2014), but they simulated an artificial lineage from which no information about the statistical properties of either the whole lineage tree or a sample of extant cells can be obtained. Additionally, they did not include heterogeneity in the mother cell volume at cell division and imprecise volume division from mother to daughter cells.

We present a time-efficient stochastic simulation algorithm with cell growth and division. Although we also simulate a single lineage (and are therefore time-efficient), we can—under certain conditions—obtain statistical properties of either the whole lineage tree or of a sample of extant cells from a single lineage simulation. Our algorithm is implemented in our stochastic simulator StochPy (presented in Chapter 7). The algorithm can be used in combination with exact SSAs, exact SSAs with delay, and inexact SSAs. We illustrate the functionality of our SSA with cell growth and division by comparing stochastic simulations with cell growth and division to theory and time-lapse microscopy data of *Escherichia coli* (*E. coli*) and *Bacillus subtilis* (*B. subtilis*) for which we found an excellent and good agreement, respectively.

8.2 Implementation

The stochastic simulation algorithm

Before we explain in detail how we extended stochastic simulation algorithms (SSAs) with cell growth and division, we first briefly explain the SSA. In Chapter 6 we introduced the SSA as a Monte Carlo method that numerically generates stochastic dynamics that are in agreement with the chemical master equation.

While various SSAs exist (e.g. the direct method (Gillespie, 1976, 1977)), we use a general description that holds for all exact SSAs. We consider a system of m molecular species and r reactions. The state of the system is described by the state vector $\boldsymbol{x}(t)$,

$$\boldsymbol{x(t)} = (n_1(t), n_2(t), \ldots, n_m(t)), \tag{8.1}$$

which contains the (copy) number of each molecular species per cell (denoted by n_i for species index i) at a given point in time. The SSA uses the propensity function,

$$a_j(\boldsymbol{x})dt = \text{the probability that reaction with index } j \text{ occurs once in } V \text{ in the time interval } [t, t+dt) \text{ given that the system is currently in state } \boldsymbol{x}, \tag{8.2}$$

and the state-change vector,

$$\boldsymbol{v} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r), \tag{8.3}$$

to explicitly simulate every single reactive reaction in the system. In short, the SSA executes the following steps:

(i) initialize $\boldsymbol{x}(t_0)$,

(ii) use a Monte Carlo method to determine **when** and **which** reaction fires next,

(iii) update the system by adjusting t and $\boldsymbol{x}$, and

(iv) return to step (ii), or else end the simulation.

We refer to Chapter 6 for more details about SSAs.

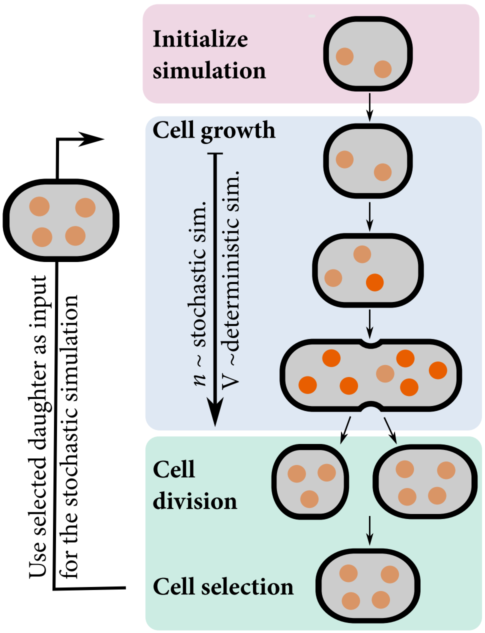
Adding cell growth and division to the SSA

The SSA with cell growth and division we describe here is implemented in StochPy (Chapter 7) and available for download from http://stochpy.sf.net. Fig. 8.1 gives an overview of how we extended the SSA with cell growth and cell division. We discuss the steps outlined in Fig. 8.1 step-by-step. Notations are explained in Table 8.1.

(i) Initialize simulation. In the extended SSA, containing cell growth and division, we simulate a single lineage, so the simulation starts with a single cell. Before performing a simulation we have to initialize the simulation by setting various parameters (Table 8.1). For instance, we have to set the cell volume of the initial cell, $V(0)$. While a cell can be in any cell-cycle stage ($0 \leq a \leq T$) where a is the cell age and T the interdivision time—the time period between two consecutive cell divisions—the initial cell is by default parameterized to start at the beginning of the cell-cycle stage (i.e. $a = 0$). During the simulation, the cell grows deterministically at a specified specific volume-growth rate (μ) and according to a particular type of growth—our algorithm supports both exponential and linear volume growth rates.

Notation	Explanation	StochPy Input
n_i	(copy) number of species with index i	• (at $t = 0$)
V	Cell volume	• (at $t = 0$)
T	Interdivision time	
f	Interdivision time PDF	
a	Cell-cycle stage ($0 \leq a \leq T$)	
μ	Specific growth rate of cell volume	•
k	Specific growth rate of the population	
$\Phi_m(V)$	Volume distribution at division for a sample of mother cells	•
K	Volume partitioning distribution	•
$\Psi_b(V)$	Volume distribution at birth for a sample of baby cells	
$\lambda_e(V)$	Volume distribution at present size for a sample of extant cells	

Table 8.1. Notations. *The specific growth rate of cell volume and the population are identical for exponential growth; they also have the same units (1/time).*



Start with an initial (daugther) cell.
Set $V(0)$, growth rate and type.
Draw $V(T)$ from Φ_{m}.

$n(0) = 2$, # of molecules at cell birth (a=0).
$V(0) = 0.9$, volume at cell birth (a=0).

$n(a) = 3$, # of molecules at age a.
$V(a) = 1.3$, volume at age a.

$n(T) = 7$, # of molecules at division (a=T).
$V(T) = 2.05$, volume at division (a=T).

Volume partitioning from $K(p)$.
Molecule partitioning.

Select daughter for next generation.
Draw $V(T)$ from Φ_{m}.

Figure 8.1. The SSA with cell growth and division as incorporated in StochPy. *The simulation starts with one daughter cell of which a single lineage is tracked through time. The stochastic simulation continues until the interdivision time T is reached. Then, the mother volume is partitioned between both daughter cells. Molecules are subsequently partitioned (volume dependent) between both daughter cells. Faded red molecules are inherited from the mother cell. This procedure is repeated until the number of generations is reached, the desired end time is reached, the desired number of time steps is reached, or all reactions are exhausted.*

Next, we have to determine when the tracked cell divides. We do this by drawing a volume, $V(T)$, at which the mature mother cell divides into two daughter cells. The volume at which the mother cell divides is drawn from a probability density function (PDF) Φ_m. This PDF is independent from the volume at birth, which makes the volume at division independent from the volume at birth as was hypothesized by Koch and Schaechter (1962). Given values for $V(0)$, $V(T)$, μ, and the type of growth, the interdivision time T can be calculated. We illustrate this here for an exponential growth rate:

$$V(T) = V(0) \cdot e^{\mu T} \rightarrow T = \frac{\ln(V(T)/V(0))}{\mu}. \tag{8.4}$$

(ii) SSA with cell growth. Knowing T allows us to start the SSA until the division event at $a = T$. We explicitly write start, because we cannot just run the SSA until the next division event at $t = T$. Modeling cell growth implies that V increases during the simulation from birth ($a = 0$) to division ($a = T$). In a growing cell, reacting molecules require more time to find each other, thus the reaction waiting times for these kind of reactions increases. This means that the propensity functions of diffusion limited reactions (i.e. second and higher-order reactions) depend on V. We therefore inserted V as a variable in the respective propensity function,

$$a_{j,V(t)} = \frac{a_j(t)}{V(t)^{order-1}} \text{ with order } \geq 2. \tag{8.5}$$

The propensity functions of zero and first-order reactions are unaffected by V. While we simulate V deterministically we only update V at each stochastic event in the simulation. This means that we calculate the time until the next reaction fires based on the V at the time of firing of the previous reaction; V is larger at the moment of reaction execution which results in a underestimation of the reaction time of second and higher-order reactions. This effect is negligible if the volume difference between the two consecutive firings, ΔV, is small. An alternative method could be to add an additional stochastic reaction for V that fires frequently as was by done Gomez et al. (2014), but this slows down the simulation.

(iii) Cell division. The SSA continues until the interdivision time T is reached (and $V = V(T)$). Both the volume and the molecular copy numbers for each species are then partitioned between the two daughter cells. The partitioning ratio is drawn from the PDF $K(p)$. Daughter one and two receive a volume of $V_{d1}(0) = p \cdot V(T)$ and $V_{d2}(0) = (1-p) \cdot V(T)$ respectively. The K distribution should be symmetrically distributed around a mean of 0.5, otherwise a bias for one daughter is created. Theoretically, this also works for microorganisms that divide asymmetrically like *Saccharomyces cerevisiae* (then a bimodal but symmetrical distribution of K should be used). The partitioning of molecules between both daughter cells—which is done next—is also a stochastic process and depends on the cell volumes of both daughters. This partitioning of molecules is, therefore, modeled with a volume-dependent binomial distribution. More specifically, the probability that a specific

molecule is inherited by daughter one is modeled as $V_{d1}(0)/V(T)$. This means that the number of molecules, with copy number n, inherited by daughter one can be drawn as a random sample from a binomial distribution with n number of trials and success probability $V_{d1}(0)/V(T)$. The process is repeated for each species. Not all cellular constituents should be binomially distributed between both daughter cells. DNA is an example of this; each daughter cells receives one copy of the chromosome in a normal cell division event. These kind of cellular constituents divide exact. Stochastic simulations also allow the definition of fixed species—species that do not change their copy number over time—which are not divided during a cell division event.

(iv) Cell selection. Starting the SSA with a single cell and simulating the entire population tree (Fig. 8.2) is computationally difficult or impossible. Tracking a single cell lineage (Fig. 8.2) through time allows us to incorporate cell growth and division in stochastic simulation algorithms in an efficient manner. While simulating a single lineage was also done by Gomez et al. (2014), we are also able to get the statistical properties of either the whole tree or a sample of extant cells from a single lineage over time. This novel theory is explained in the next section.

From a single lineage simulation to extant-cell population distributions

In order to relate the statistical properties of a single lineage over time to the statistical properties of either the whole tree or a sample of extant cells, the lineage that is chosen must reflect a well defined cell sample. During balanced growth three different types of samples can be distinguished for which the statistics of cell age, interdivision time, and cell volume are interrelated: These are samples of extant, mother, and baby cells (Painter and Marr, 1968). Briefly, extant cells are all cells that exist at a specific moment in time, a sample of mother cells consists of all cells that divide within a defined time interval, and a sample of baby cells consists of all cells that are born within a defined time interval. The interdivision times for these three types of samples are different but interrelated (Painter and Marr, 1968):

$$f_e(t) = 2\left(1 - e^{-kt}\right) f_b(t) = \left(e^{kt} - 1\right) f_m(t). \tag{8.6}$$

Here k denotes the specific growth rate of the population and f_s the PDF of interdivision times for the different types of samples (with s as either e: extant, b: baby, or m: mother). For our simulations we select at each division the daughter cell to be followed in such a way that the resulting simulated lineage corresponds to a sample of mother cells. This is possible under the following two conditions:

(i) The volume at which cells divide is independent of the volume at birth and any other cell-specific properties like concentrations of certain molecules. To ensure that volumes at division are independent of volumes at birth, the distribution of division ratios (K) and the distribution of volumes at division (Φ_m) should be chosen such that the largest possible

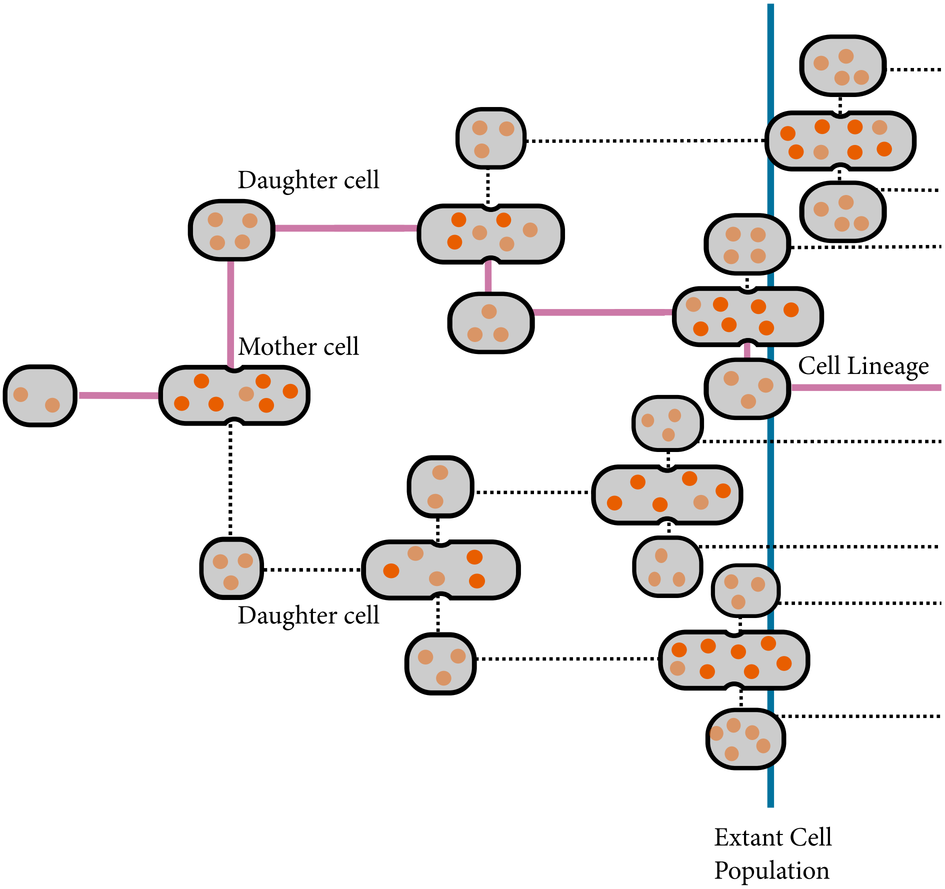


Figure 8.2. Cell lineage and a cell population. *A cell lineage (purple) follows one cell through time, choosing at each cell division event—with a certain probability—one of the daughters to follow. Simulation of the complete lineage tree results in following all daughters after each cell division event, which requires simulation of* 2^{n+1} *cells where* n *is the number of generations. The extant cell population (blue) are the cells in a population at a given moment in time. Faded red molecules are inherited from the mother cell.*

volume at birth is smaller than the smallest possible volume at division (i.e. there is no overlap between the two volume distributions). By default we use beta distributions for both the division ratio as well as the volume at division. Beta distributions are bounded and symmetric if its two positive shape parameters are identical.

(ii) The growth law for a single cell is deterministic and independent of concentrations. The current implementation allows exponential and linear growth laws for single cells. For an exponential growth law the volume specific growth rate (μ) is equal to the specific growth rate of the population (k). Otherwise, the specific growth rate of the population needs to be calculated. We do this by using the known relations between the volume at division, volume at birth, the growth law, the volume distribution of extant cells (Eq. (8.7) (Painter and Marr, 1968)), and the fact that this volume distribution has to integrate to one:

$$\int_0^{V_{max}} \lambda_e(V)dV = \int_0^{V_{max}} e^{-R(V)} \cdot \left(\int_0^V \frac{k \cdot e^{R(\bar{V})}[2\Psi_b(\bar{V}) - \Phi_m(\bar{V})]}{g(\bar{V})} d\bar{V} + C \right) dV = 1 \tag{8.7}$$

where

$$R(\bar{V}) = \int_0^{V_{max}} \frac{(g'(\bar{V}) + k)}{g(\bar{V})} d\bar{V}. \tag{8.8}$$

Here, V is the cell volume, C an integration constant (which can be calculated from boundary conditions of the volume distribution), $\lambda_e(V)$ the volume distribution of a sample of extant cells, $\Psi_b(V)$ the volume distribution at birth for a sample of baby cells, $\Phi_m(V)$ the volume distribution of a sample of mother cells, and $g'(V)$ the differential of the formula of cell volume growth—$g(V) = \mu \cdot V$ for exponential growth and $g(V) = \mu$ for linear volume growth, with μ as the volume specific growth rate. We know $\Phi_m(V)$, $g(V)$, $g'(V)$, and C from the model parameters. $\Psi_b(V)$ can be calculated from $\Phi_m(V)$ and the partition distribution $K(V)$ using

$$\Psi_b(V) = \int_V^{\infty} \frac{\Phi_m(\theta)}{\theta} \cdot K(\frac{V}{\theta}) d\theta. \tag{8.9}$$

A solution for Eq. (8.7) is approximated by using the secant or Newton-Raphson method.

To generate a lineage that is representative for a sample of mother cells, at each division the daughter to be followed by the simulation is chosen with a probability according to the fraction of descendants it can be expected to contribute to the population. Mathematically, if daughter one is expected to have n_{d1} descendants at a later time point t_x and daughter two n_{d2} descendants, the probability p to choose daughter one is given by

$$p = \frac{n_{d1}}{n_{d1} + n_{d2}}. \tag{8.10}$$

Let T_1 and T_2 be the interdivision times of daughters one and two respectively. In balanced growth the number of expected descendants at time t_x is then given by $n_{d1} = e^{k\cdot(t_x - T_1)}$ and $n_{d2} = e^{k\cdot(t_x - T_2)}$. This is possible because the growth law does not depend on molecule concentrations or previous history. Inserting these relationships for n_{d1} and n_{d2} in Eq. (8.10) yields

$$p = \frac{e^{k\cdot(t_x - t_1)}}{e^{k\cdot(t_x - T_1)} + e^{k\cdot(t_x - T_2)}} = \frac{e^{k\cdot(T_2 - T_1)}}{1 + e^{k\cdot(T_2 - T_1)}} = \frac{e^{k\cdot\Delta T}}{1 + e^{k\cdot\Delta T}} \tag{8.11}$$

where $\Delta T = T_2 - T_1$. In short, the larger daughter cell is more probable to be chosen, because this cell will reach the next division volume faster and is therefore likely to have more descendants in the population at a later time point. Another way to derive this probability is given in Eq.(S8.25).

With as simulation result a lineage that represents a sample of mother cells, statistical properties of other defined samples can be calculated based on the known relationships of interdivision time and cell age between these samples. For example, we can calculate the copy number distributions for a sample of extant cells (corresponding to any experiment that records molecule copy numbers at a fixed moment in time as smFISH) from the simulation of a single lineage by using

$$p(n_x = n) = \int_0^{T_{max}} \int_0^{a_{max}} g(a,T) \frac{I(n,a,T)}{\sum_i I(n_i,a,T)} dadT. \tag{8.12}$$

Here, $\frac{I(n,a,T)}{\sum_i I(n_i,a,T)}$ is the relative frequency of copy number equal to n_x in the simulated lineage at age a for a cell with interdivision time T approximating $p(n_x = n|a,T)$ and $g(a,T)$ is the joint PDF of cell age and interdivision time,

$$g(a,T) = kf_m(T) \cdot e^{k(T-a)}. \tag{8.13}$$

The derivation of $g(a,T)$ is given in Eq. (S8.26) and (S8.27). Using Eq. (8.13) we can rewrite Eq. (8.12) to

$$p(n_x = n) = \int_0^{T_{max}} \int_0^{a_{max}} f_m(T) \cdot k \cdot e^{k \cdot (T-a)} \frac{I(n,a,T)}{\sum_i I(n_i,a,T)} \cdot dadT, \tag{8.14}$$

where $I(n,a,T)$ is an indicator function equal to the number of occurrences of copy number $n_x = n$ at cell age a in a cell with interdivision time T. Obtaining the statistics for an extant cell population consists of two main steps. First, in order to work with the indicator functions, the simulation time series needs to be binned. The binning needs to be performed at regular intervals for cell age. And second, a double integral has to be taken which is a slow procedure for infinitesimal small steps of a and T. We approximate this double integral by using the Riemann sum (or the 2D trapezoidal rule).

Additionally, the statistics for a sample of extant cell volumes can be calculated from the simulation of a single lineage by re-using Eq. (8.14) and replacing copy numbers with volume data. Calculating the statistics for sample of extant cell volumes requires one additional step—the continuous volume data must be binned to make the distribution discrete.

With deterministic interdivision times, the samples of baby, mother, and extant cells are identical. However, we have to correct for the fact that all cell divisions in the simulation (of a lineage or the complete three) are synchronized, which can be done by correcting both copy number and volume distributions for the cell age distribution (which is theoretically known for this specific condition (Eq. (S8.23)).

8.3 Results and Discussion

Performing StochPy simulations with cell growth and division

A typical StochPy session consists of first creating a StochPy cell division model object. Once a model (written in SBML or PySCeS MDL) is loaded various simulation parameters that are specific for the SSA with cell growth and division can be set (e.g. μ, Φ_m, and K; see also Tabel 8.1). Similar as described in Chapter 7, kinetic parameter values and molecule copy numbers can be modified interactively, and simulations can be performed by calling the available analysis methods for model objects. As model objects are fully encapsulated, multiple models can be instantiated from the same (or different) input files at the same time. An example of a short StochPy session within Python is the following:

```
import stochpy
cmod = stochpy.CellDivision(model_file = "ImmigrationDeath.psc")
cmod.ChangeParameter("Ksyn",2)
cmod.ChangeParameter("Kdeg",0.1)
cmod.ChangeInitialSpeciesCopyNumber("mRNA",10)
cmod.SetGrowthFunction(growth_rate=0.1,growth_type="exponential")
cmod.SetVolumeDistributions(Phi = ("beta",5,5), K = ("fixed",0.5),
                                          Phi_beta_mean=2)
cmod.DoCellDivisionStochSim(end=10,mode="generations")
cmod.PlotSpeciesVolumeTimeSeries()
```

where we start by initiating the model object `cmod` for the immigration-death model depicted in PySCeS MDL, and modifying the kinetics parameters and mRNA copy number at $t = 0$ interactively. Next, we set volume specific growth rate characteristics and volume distributions. Then, we generate one time trajectory with cell growth and division (ten generations) and plot the corresponding (discrete) molecule copy number and volume time series data.

More generations are necessary to get an accurate prediction of species and volume distributions:

```
cmod.DoCellDivisionStochSim(end=1000,mode="generations")
cmod.PlotVolumeOverview()
cmod.PlotSpeciesOverview()
```

There is interdivision time heterogeneity, so the three different samples (mother, baby, and extant) can be distinguished. The simulated lineage corresponds to a sample of mother cells and StochPy pre-calculates the statistical properties of a sample of extant cells. By default, the statistical properties of a sample of extant cells are plotted. The high-level functions `PlotVolumeOverview()` and `PlotSpeciesOverview()` generate figures such as Figs. 8.3 and 8.4. For each "panel", a unique high-level function can be used to plot the respective data. In this example we used the default number of bins for both age and interdivision time. StochPy generates a warning if the numerical integration was not accurate enough. Using the high-level function `AnalyzeExtantCells()` with a different number of bins for both age and interdivision time typically solves this issue.

In the following sections, we compare StochPy simulations first to theory and then to single-cell experimental data that was obtained with time-lapse microscopy experiments (see Material and Methods for more details). In each comparison, we highlight the included contributions of cell growth and division to non-genetic cell variability. The analytical solutions for the first three case studies are given in the Supplemental Information. We re-used the immigration-death model (i.e."synthesis-degradation model") to study single-cell transcription with cell growth and division, because this is the only model for which we know analytical solutions. This model consists of a zero-order mRNA synthesis reaction with rate constant k_{syn} and a first-order mRNA degradation reaction with rate constant k_{deg}. We refer to Chapter 6 where this model is discussed in detail. We used the settings given in Table 8.2 to—for molecule copy numbers and cell volume—compare time series, distributions at birth ($a = 0$) and division ($a = T$), and distributions for a sample of extant cells.

Parameter (unit)	Case study 1 (theory)	Case study 2 (theory)	Case study 3 (theory)	Case study 4 (*E. coli*)	Case study 5 (*B. subtilis*)
k_{syn} (min^{-1})	2.0	2.0	2.0	2.0	0.12
k_{deg} (min^{-1})	0.1	0	0.1	0.1	0.0
μ (min^{-1})	0.1	0.1	0.1	0.01	0.01
$V(0)$ (μm^3)	1.0	1.0	1.0	1.6^{ii}	2.28^{ii}
Φ_m (μm^3)	Beta(5,5)^{i}	2	2	$\mathcal{N}$(2.98,0.35)^{ii}	$\mathcal{N}$(4.73,0.58)^{ii}
K	0.5	0.5	0.5	Beta(157,157)	Beta(108,108)

Table 8.2. Parameters used in the different case studies. *We simulated each SSA with cell growth and division for 10000 generations. The time-lapse microscopy data obtained for E. coli and B. subtilis was used to fit the input for the stochastic simulation (Material and Methods).*

i. We used a beta distribution—defined at interval [0,1]—and scaled it to a mean of 2.0. The two positive shape parameters, α and β of the beta distribution must be identical ($\alpha = \beta$) to get a symmetric distribution defined at interval [0,1] with a mean of 0.5.

ii. Time-lapse microscopy experiments were done in 2D so we measured length (μm) rather than volume.

Comparing StochPy simulations with cell growth and division to analytical solutions

Case Study 1: Volume statistics are independent of the model

We first analyzed the volume statistics generated by StochPy simulations. While we used the synthesis-degradation model we could have used any model because the volume statistics are completely independent of the model used. Using a different model and/or parameter settings has only an effect on the run time of the simulation. The settings that we used to generate the results are given in the second column of Table 8.2.

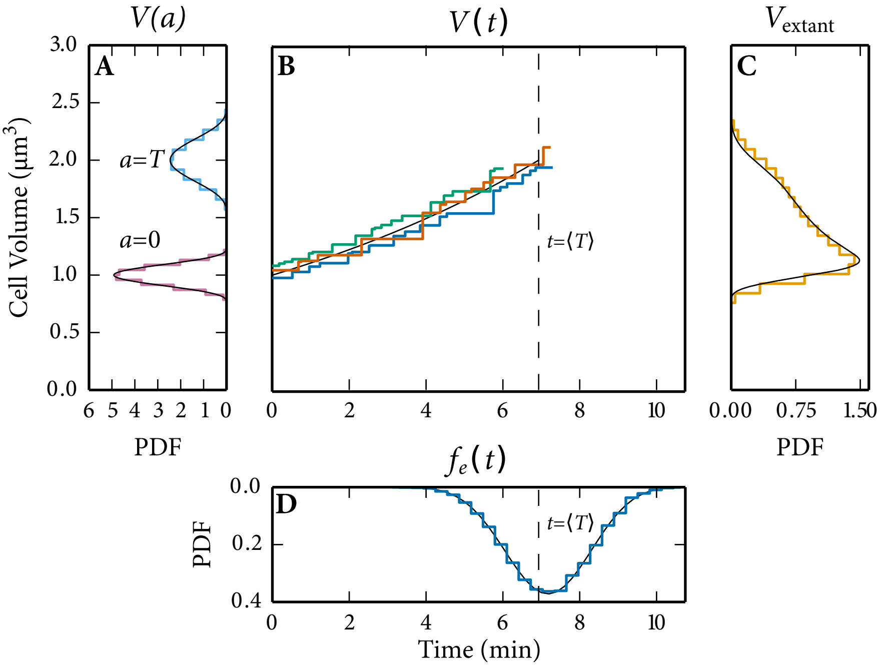


Figure 8.3. Volume statistics obtained via the SSA with cell growth and division match analytical solutions (black). *(A) volume distributions at* $a = 0$ *and* $a = T$ *of a sample of baby cells and mother cells respectively. (B) three stochastic time trajectories of cell volume fluctuate around its analytical solution with* $V(0) = 1$*. Each of the time trajectories has a interdivision time that is distributed (see panel D) around its mean,* $\langle T \rangle$ *(dashed line). (C) the extant volume distribution. (D) the interdivision time distribution of a sample of extant cells.*

Comparison of the volume statistics generated by StochPy to analytical solutions shows an excellent overall agreement (Fig. 8.3). In our stochastic simulation we took into account the following contributions of cell growth and division to non-genetic cellular heterogeneity: imprecise, binomial, partitioning of molecules at cell division and heterogeneity in the mother cell volume at cell division. The latter was modeled by drawing division volumes from a beta distribution (Eq. (S2)). Imprecise volume division from mother to daughter cells was not taken into account because we partitioned the cellular volume exactly between both daughter cells ($K = 0.5$). Heterogeneity in the cell volume at division has several consequences. First, the cellular volume at a given age is variable as is illustrated in Fig. 8.3A for cells at division ($a = T$) and at birth ($a = 0$). As expected, the cell volumes at birth and division do not overlap, so they are independent of each other. This allowed us to calculate the volume distribution of a sample of extant cells (Fig. 8.3C) which is in agreement with theory. And second, the interdivision time T is not deterministic as is illustrated in Fig. 8.3B and D. In the latter, we compare the interdivision time distribution of a sample of extant cells (f_e) obtained by simulation with theory and found an excellent agreement.

In Fig. 8.3B we simulated three distinct generations that each started with a different $V(0)$ drawn from $\Psi_b(V)$. We selected the parameter values in our model such that the number of firings (reactions that occur) per generation are limited. This allows us to illustrate that StochPy updates V only when a reaction fires (Fig. 8.3B). Our model does not contain any second or higher-order reactions, so this has no effect on the accuracy of the simulation. As explained, this can have an effect if second-order reactions are included and when the rate of firing is slow (then the volume difference becomes significant).

Case Study 2: mRNA synthesis

In addition to predicting accurate volume statistics, we also aimed at predicting accurate molecule copy number statistics when we include the stochastic contributions of cell growth and division to non-genetic cell variability. Unfortunately, we do not know (yet) the analytical solutions if we include all sources of stochasticity of cell growth and division. Hence, we used a deterministic T and simplified the model by using no active degradation ($k_{deg} = 0$). Using a deterministic T was done by using a fixed Φ_m and K—the specific settings are given in the third column of Table 8.2. Or in other words, we did not take into account the stochastic contributions of the mother cell volume at division and of the imprecise volume division from mother to daughter cells. This means that the only contribution of cell growth and division to non-genetic cell variability that we took into account was the imprecise partitioning of molecules at cell division.

The results given in Fig. 8.4 show that predictions made with the stochastic simulation are consistent with analytical solutions, which can be found in the Supplemental Information. More specifically, the mRNA copy number distributions at birth and division obtained with stochastic simulations overlap with the theoretical distributions (Fig. 8.4A) and the time series output of stochastic simulations fluctuate, as expected, around the theoretical time series (Fig. 8.4B). The deviation shown in Fig. 8.4B is explained by inherent stochasticity in net molecule synthesis in stochastic simulations. We also found an excellent agreement between the simulated and theoretical mRNA copy number distribution of a sample of extant cells (Fig. 8.4C). This result demonstrates that we can simulate a single lineage and accurately predict statistical properties of well defined samples (by in this case correcting for the cell age distribution).

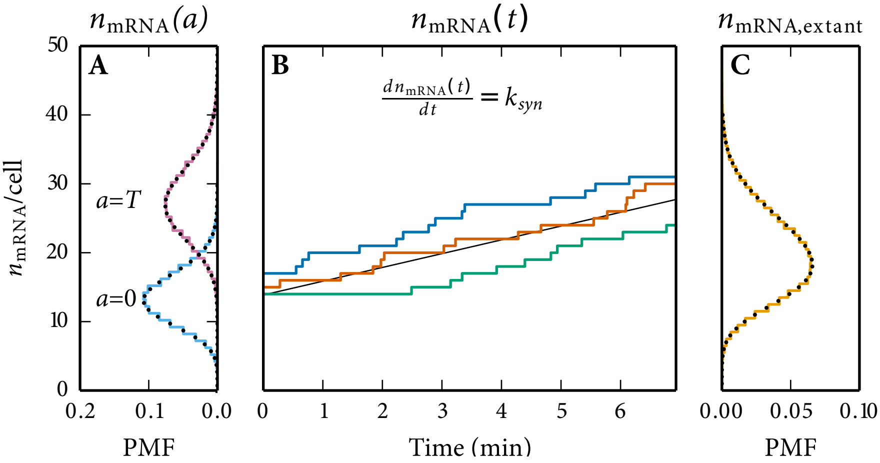


Figure 8.4. Molecule copy number statistics obtained for the mRNA synthesis model via the SSA with cell growth and division match analytical solutions (black). *(A) n_{mRNA} distributions at $a = 0$ and $a = T$ (identical for all three samples). (B) three stochastic time trajectories of n_{mRNA}. (C) the extant n_{mRNA} distribution.*

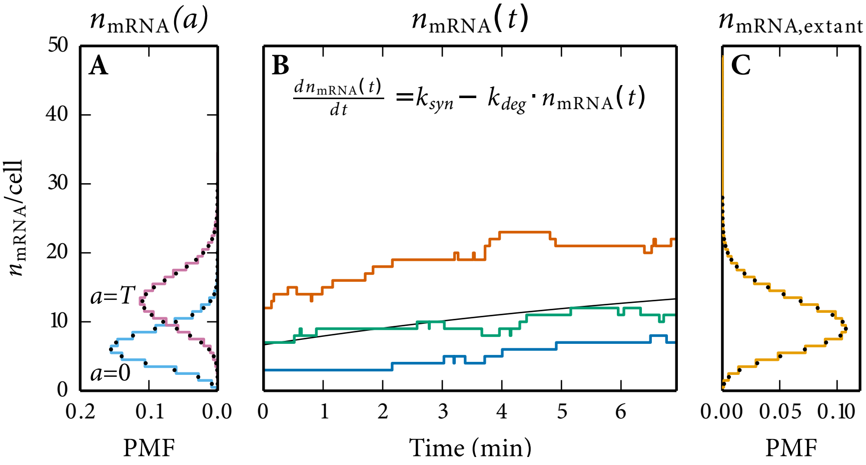


Figure 8.5. Molecule copy number statistics obtained for the mRNA synthesis and degradation model via the SSA with cell growth and division match analytical solutions (black). *(A) n_{mRNA} distributions at $a = 0$ and $a = T$ (identical for all three samples). (B) three stochastic time trajectories of n_{mRNA} fluctuate around its analytical solution. (C) the extant n_{mRNA} distribution.*

Case Study 3: mRNA synthesis and degradation

Proteins are actively degraded into the cell, so we decided to extent our simple model with active degradation. Except for the degradation rate, we used exactly the same settings (fourth column of Table 8.2) as in the example without active degradation. The imprecise partitioning of molecules at cell division was, therefore, again the only contribution of cell growth and division to non-genetic variability that was taken into account.

The results given in Fig. 8.5 show again an almost perfect agreement between stochastic simulation and analytical solutions (the mRNA copy number distribution of a sample of extant cells was numerically solved). The analytical solutions can be found in the Supplemental Information. Comparison of Figs. 8.4 and 8.5 shows that, as expected, adding active degradation results in lower mRNA copy numbers at different ages in the lineage and also in the extant cell population.

Comparing the SSA with cell growth and division to experimental data

In the previous sections we demonstrated that the predictions of the SSA with cell growth and division are consistent with theory. Here, we compare the outcome of the SSA with cell growth and division to time-lapse microscopy measurements (Material and Methods) of *E. coli* and *B. subtilis*. All sources of stochasticity defined in the introductory section of this chapter—net molecule synthesis, imprecise partitioning of molecules at cell division, heterogeneity in the mother cell volume at cell division, and imprecise volume division from mother to daughter cells—were taken into account in the stochastic simulations. For both organisms, we used the time-lapse microscopy measurement data to fit the required input of our algorithm: Φ_m, K, μ, and "$V(0)$" (Table 8.2). More details about this fitting procedure can be found in Material and Methods. The time-lapse microscopy measurements were done in 2D which made it difficult to get an accurate measure of cell volume. Hence, our analysis are limited to measurements of cell length (L) which increased exponentially in the population (data not shown).

Case Study 4: *E. coli* length measurements

We first focus on the time-lapse microscopy data obtained with *E. coli* (Fig. 8.6). Since we fitted the length distribution at division Φ_m and the length partitioning ratio K, we can expect an excellent agreement between the length distribution at birth and division (Fig. 8.6A). While both Φ_m and K were drawn from a (beta) distribution, K is in comparison to Φ_m a narrow beta distribution, with a mean of 0.5 and a standard deviation of ≈0.03. This suggests that, in *E. coli*, the stochastic contribution of the mother cell volume at cell division is greater in comparison to the imprecise volume division from mother to daughter cells.

The results displayed in Fig. 8.6A and B also show that the time-lapse microscopy data obeys the two requirements, which were specified earlier, for the simulated lineage to correspond to a sample of mother cells. That is, the experimental distributions in Fig. 8.6A show that there is practically no overlap between the length distributions at birth and division. Additionally, the time series data shown in Fig. 8.6B illustrates that assuming a deterministic growth rate is a valid assumption. While close inspection shows that there is some variability in the instantaneous growth rate, this variability is negligible to the variability in length at birth and at division as illustrated in Fig. 8.6A. We note that for some, more recent, studies it is this variability though that is of most interest (Kiviet et al., 2014).

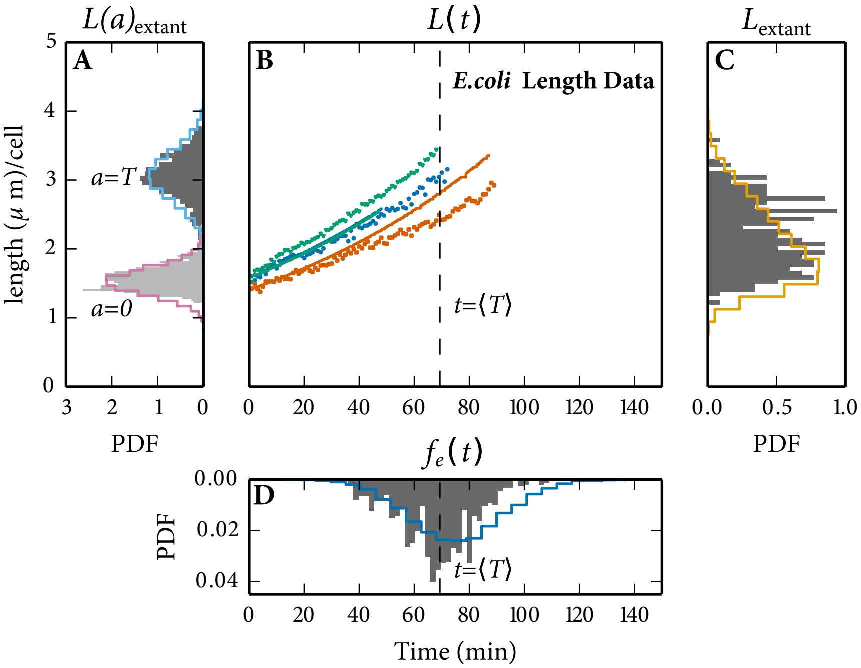


Figure 8.6. *E. coli* cell length statistics obtained via the SSA with cell growth and division match a time-lapse microscopy measurement (grey). *(A) length distributions at $a = 0$ and $a = T$ of a sample of extant cells. (B) stochastic time trajectories (lines) and time-lapse microscopy measurements (dots) of E. coli cell length. We measured a mean interdivision time, $\langle T \rangle$, of about 69 minutes, which corresponds to a growth rate of about 0.01 min^{-1}. (C) the extant length distribution. (D) the interdivision time distribution of a sample of extant cells.*

The detected heterogeneity in the mother-cell length at division, and to a lesser extent also the imprecise length division of mother cells gives variable interdivision times (Fig. 8.6D)—since we assume deterministic cell-volume growth rate. Although our simulations predicted a broader and slightly shifted distribution of interdivision times, the coefficient of variation of the simulated interdivision times distribution is with 0.25 in agreement with measurements—coefficients of variation between 0.23–0.32 were found for *E. coli* interdivision time distributions (Harvey et al., 1967; Reshes et al., 2008; Tsukanov et al., 2011). A first candidate explanation for a broader and slightly shifted predicted distribution of interdivision times is that the ratio of cell length at division and birth is on average greater in the simulation than in the experiment (see Fig. 8.6A). A greater ratio results in a greater interdivision time (Eq. (8.4)). A second possible explanation for this difference could be that we measured, and therefore simulated, cell length in stead of cell volume; the interdivision time is based on the growth rate and the volumes at birth and division. A less broad distribution of volume than that of length would result in a less broad distribution of interdivision times. This reasoning is further supported by data since we found a good agreement between simulation and measurement for the length distribution of a sample of extant cells, which again demonstrates that tracking a single lineage is sufficient to get information about the extant cell population.

Case Study 5: *B. subtilis* length and fluorescence measurements

Finally, we focus on the time-lapse microscopy data obtained with *B. subtilis*. Also, this data is consistent with stochastic simulations (Fig. 8.7). In addition to length data we also measured fluorescence of a GFP protein (Material and Methods), which we simulated as a zero-order synthesis reaction. The contributions of heterogeneity in the mother cell length at division, and to a lesser extent also of the imprecise length division from mother cells are again visible in Fig. 8.7A-D. The analysis of *B. subtilis* cell length data is highly comparable to the *E. coli* cell length data. In short, we found again that the stochastic contribution of the mother cell length at cell division is greater in comparison to the imprecision of the length division of mother cells. We found a coefficient of variation of 0.26 for the simulated interdivision times. To our knowledge, no coefficients for variation of *B. subtilis* are known, but recent measurements have shown that they are (highly) variable (Gregory et al., 2008).

To describe the fluorescence data we found, surprisingly, a better agreement between experiment and simulation with exact partitioning of molecules at cell division (Fig. 8.7E-G). The greatest discrepancy between experiment and simulation is for the fluorescent-signal distribution at birth (Fig. 8.7E). Here, we see that the stochastic simulation predicts a broader distribution of the fluorescent signal at birth. More specifically, the stochastic simulation predicts more cells with a lower fluorescent signal at birth than was experimentally measured. We attribute this difference to the fact that the fluorescent signal was measured only every ten minutes and therefore typically not at cell birth ($a = 0$) and cell division ($a = T$). Our current data set was not sufficiently large to extract only the cells where we measured fluorescence at birth and cell division, thus we included all 346 cells and used the first and last fluorescence measurement as $a = 0$ and $a = T$, respectively. This gives a five minute deviation at $a = 0$ and $a = T$, which is significant given that the interdivision time was about 69 minutes. The fluorescent signal increases (on average) with a factor of two during the cell cycle (Fig. 8.7F). Measuring the fluorescent signal at $a = 5$ rather than at $a = 0$ results therefore in an overestimation of the fluorescent signal in the experiment (and an underestimation for $a = T - 5$ rather than at $a = T$). This hypothesis is further supported by data because we found a good agreement between experiment and simulation for the fluorescent signal in a sample of extant cells (Fig. 8.7G).

8.4 Conclusions

We presented an efficient stochastic simulation algorithm with cell growth and division that takes into account net molecule synthesis, imprecise partitioning of molecules at cell division, heterogeneity in the mother cell volume at cell division, and imprecise volume division from mother to daughter cells as sources of stochasticity. Our algorithm is efficient because we generate a single lineage that is representative for a sample of mother cells, which allows us to get the statistics of other samples such as a sample of baby and extant cells. We demonstrated this by comparing stochastic simulations to both theory and time-lapse microscopy data. We implemented our stochastic simulation algorithm with cell growth and division into our comprehensive and user-friendly stochastic simulator `StochPy`. This allows the systems biology community to profit from our stochastic simulation algorithm with cell growth and division.

Even though this algorithm is, in its current version, already remarkably capable of simulating experimental data, it does have a number of limitations. First, we model the growth of a cell during a single generation as a deterministic process. This is not of major consequence because the variability in the growth behavior of single cells is mostly due to heterogeneity of birth and division volumes than due to noise in the instantaneous volume growth rate; as evident from the experimental data reported in this chapter. Noise in the instantaneous volume growth rate is, however, an interesting quantity since it a readout of net, propagated noise of all the metabolic activities of a cell. Second, the two-way coupling between biochemical and growth noise, which may in some cases reinforce each other, is not taken into account. The biochemical circuitry is, in the current version of the algorithm, implemented in a growing cell environment, with its own stochasticity that is insensitive to the simulated biochemical circuitry; the circuitry does however respond to the stochasticity associated with cell growth. Third, the noise in the molecule copy numbers of a biochemical circuit generally has an extrinsic noise contribution that originates from stochastic processes that are external to the circuit (the exact origins are not completely clear). We did not add such a noise source to the simulation algorithm. This is however not complicated and should be an extension of the next version. And fourth, since we simulate a single lineage and rely on the balanced growth assumption to arrive at the statistics of the whole lineage tree, we cannot simulate the probability distributions associated with shifts in growth rate from one balanced growth condition to the next. This is also of great experimental interest

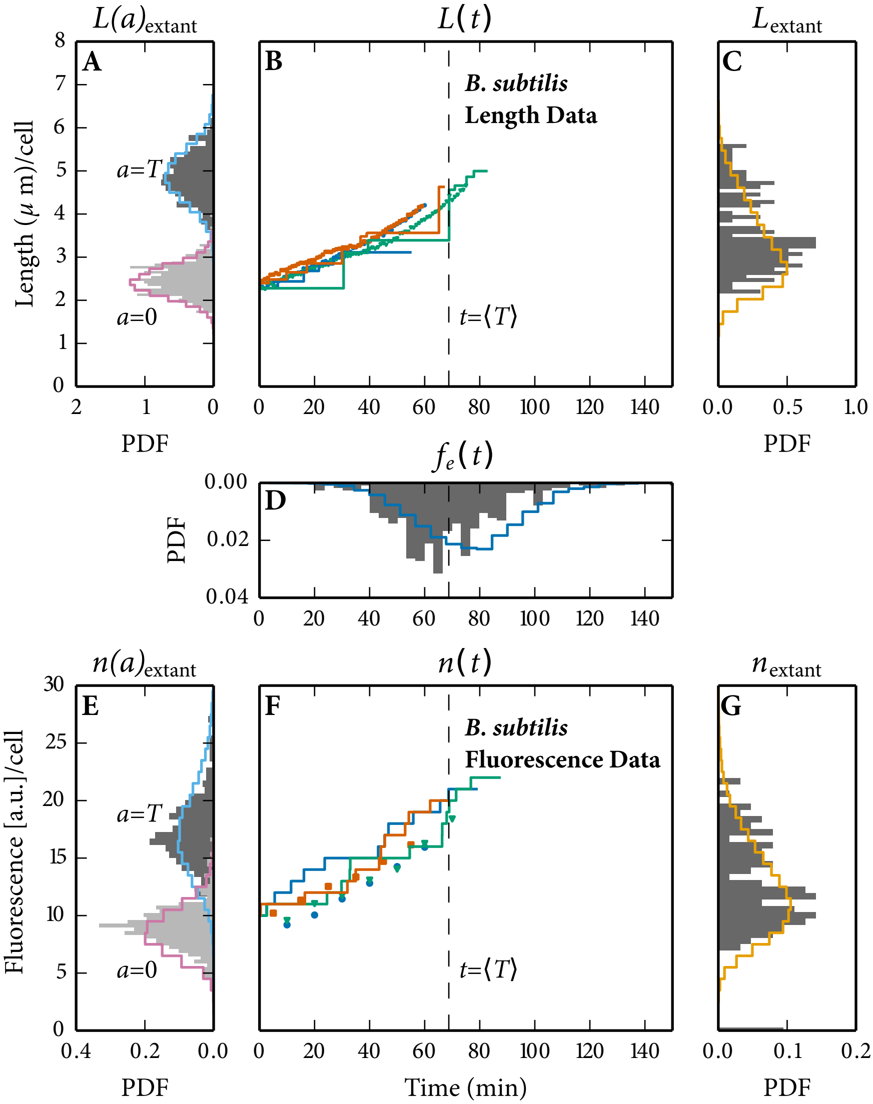


Figure 8.7. ***B. subtilis*** **cell length and molecule copy number statistics obtained via the SSA with cell growth and division match a a time-lapse microscopy measurement (grey).** *(A) length distributions at* $a = 0$ *and* $a = T$ *of a sample of extant cells. (B) stochastic time trajectories (solid) and time-lapse microscopy measurements (dots) of B. subtilis cell length. We measured a mean interdivision time,* $\langle T\rangle$*, of about 69 minutes, which corresponds to a growth rate of about 0.01* min^{-1}*. (C) the extant length distribution. (D) the interdivision time distribution of a sample of extant cells. (E) GFP fluorescence distributions at* $a = 0$ *and* $a = T$ *of a sample of extant cells. (F) three stochastic time trajectories (lines) and time-lapse microscopy measurements (dots) of GFP fluorescence. (G) the extant GFP fluorescence distribution.*

at the moment. Unfortunately, simulating such growth-transition dynamics is not trivial; mostly because the associated theory is missing such that the entire tree needs to be simulated, unless stringent assumptions are made or the algorithm is partially parameterized by experimental data on the growth rate dynamics of single cells. The latter is possible but computationally challenging. So even though we have made great progress in this chapter, several open problems remain that are of interest for future research.

8.5 Material and Methods

Stochastic simulations

All stochastic simulations were done with the direct method algorithm (Gillespie, 1976, 1977) extended with cell growth and division, as implemented in the CellDivision module of the stochastic simulation software package StochPy (version 2.3), as presented in Chapter 7. Each stochastic simulation was continued for 10000 generations.

Annotated scripts and input files used to generate modeling and simulation results are available as Scripts S1. More information on installing and using StochPy can be found in the *StochPy User's Guide* which together with additional example sessions is available online at http://stochpy.sf.net.

Microscopy experiment

Strain, medium and culturing

Escherichia coli strain AB460 (kindly provided by D. Kiviet) was revived from glycerol stock by inoculating directly into M9 minimal medium (42.2 mM Na_2HPO_4, 22 mM KH_2PO_4, 8.5 mM NaCl, 11.3 mM $(NH_4)_2SO_4$, 2.0 mM MgSO4, 0.1 mM $CaCl_2$), supplemented with trace elements (63 μM $ZnSO_4$, 70 μM $CuCl_2$, 71 μM $MnSO_4$, 76 μM $CoCl_2$, 0.6 μM $FeCl_3$), 0.2 mM uracil (all chemicals from Sigma) and 20 mM glucose (Boom B.V., NL) as carbon source (M9-Glu). At intervals of 3 hours, the pre-culture was transferred twice to fresh M9-Glu before inoculating an overnight culture to a final optical density (OD, 600 nm) of 2.5×10^{-7} in M9-Glu. After 16 hours, the culture was again diluted to an OD600 of 0.0025. When the culture reached an OD600 of 0.01, 2μL was transferred to a 1.5% agarose pad freshly prepared with M9-Glu.

Bacillus subtilis strain B1115 (amyE::Phyper-spank-sfGFP, Spcr) was constructed from parental strain BSB1 168 *trp+* using pDR111 (kindly provided by D. Rudner). B1115 was revived from glycerol stocks by directly inoculating into MM minimal medium (40 mM MOPS, 1.23 mM K_2HPO_4, 0.77 mM KH_2PO_4, 15 mM $(NH_4)_2SO_4$, 0.8 mM $MgSO_4$), supplemented with 5 mM glucose, 1 mM IPTG and trace elements (80 nM $MnCl_2$, 5 μM $FeCl_3$, 10 nM $ZnCl_2$, 30 nM $CoCl_2$, 10 nM $CuSO_4$) to a final OD600 of 4.5×10^{-7}. After incubation for

16 hours, when the culture reached an OD600 of about 0.1, the culture was diluted in TSS minimal medium (37.4 mM NH_4Cl, 1.5 mM K_2HPO_4, 49.5 mM TRIS, 1mM $MgSO_4$, 0.004% $FeCl_3$ / 0.004% Na_3-citrate × $2H_2O$), supplemented with 5 mM glucose, 1 mM IPTG and trace elements to a final OD600 of 0.001. When the culture reached an OD600 of 0.02, 2 μL was transferred to a 1.5% agarose pad freshly prepared with TSS.

All cultures were incubated at 37 ℃, shaking at 200 rpm. The agarose pad was inverted onto a glass bottom microwell dish (35 mm dish, 14 mm microwell, No. 1.5 coverglass) (Matek, USA), which was sealed with parafilm and taken to the microscope for imaging.

Microscopy and data analysis

Imaging was performed with a Nikon Ti-E inverted microscope (Nikon, Japan) equipped with 100X oil objective (Nikon, CFI Plan Apo λ NA 1.45 WD 0.13), Zyla 5.5 sCmos camera (Andor, UK), brightfield LED light source (CoolLED pE-100), fluorescence LED light source (Lumencor, SOLA light engine), GFP filter set (Nikon Epi-Fl Filter Cube GFP-B), computer controlled shutters, automated stage and incubation chamber for temperature control. Temperature was set to 37 ℃ three hours prior to starting an experiment. Nikon NIS-Elements AR software was used to control the microscope.

Brightfield images (80 ms exposure time at 3.2% power) were acquired every minute for 7.5 hours to 10 hours. GFP fluorescence images (1s exposure at 25% power) were acquired every 10 min. Time-lapse data was processed with custom MATLAB functions developed within our group. Briefly, an automated pipeline segmented every image, identifying individual cells and calculating their spatial properties. Cells were assigned unique identifiers and were tracked in time, allowing for the calculation of time-dependent properties including cell ages, elongation rates, and interdivision times.

All measurements presented here originate from the exponential growth phase of the population. We fitted the length distribution at division for a sample of mother cells (Φ_m) from experimental data with a normal distribution ($\mathcal{N}(\mu, \sigma)$), and the length partitioning distribution (K) with a symmetric beta distribution. Note that we measured Φ_e rather than Φ_m. If sufficient data is available, the interdivision time distribution can be used to relate the experimentally measured Φ_e to Φ_m, but we corrected for this by subtracting a small value from Φ_m. The growth rate was obtained from the average interdivision time (Eq. (8.4)). For cells expressing a fluorescent construct, the fluorescence at birth and division were taken as the first an last measurement during the single cell time trace. Because we measured the GFP fluorescence once every ten minutes, an average deviation of five minutes from $a = 0$ and $a = T$ is therefore present. We stochastically modeled the GFP fluorescence as a zero-order reaction. The parameters of the stochastic model ($n(0)$ and k_{syn}) were derived by linear fitting the mean fluorescence at a given age ($n|a$) and assuming doubling of the fluorescence during a average cell cycle (Eq. (S8.13)).

8.6 Supplemental Information

Scripts S1. Annotated scripts and input files used to generate modeling and simulation results can be found at http://persistent-identifier.org/?identifier=urn:nbn:nl:ui:18-23543.

This supporting information contains analytical results that we used to demonstrate the functionality of cell growth and division in stochastic simulation algorithms (SSAs) as implemented in StochPy. In all examples, we used exponential cell volume growth which follows the exponential growth function,

$$V(a) = V(0) \cdot e^{\mu \cdot a}, \tag{S8.1}$$

where $0 \leq a \leq T$.

Analytical solutions: Case Study 1 Volume distributions

We drew the cell volume at division (Φ_m) from the following scaled beta distribution:

$$V_{division} \sim \Phi_m(V) = \frac{(V-1.5)^{\alpha-1}(1-(V-1.5))^{\beta-1}}{B(\alpha,\beta)}, \qquad V \in [1.5, 2.5], \tag{S8.2}$$

with V as the cell volume and B as the Beta function that normalizes the distribution. Since the beta distribution is defined on the interval [0,1], we scaled this distribution to the interval [1.5,2.5]. When the division volume was reached, we divided the mature mother cell volume equally between both daughter cells (volume partitioning distribution $K = 0.5$). The distribution of the volume at birth was obtained by calculating the distribution of the product of two random variables, using Φ_m and the partitioning distribution K. With K as the Dirac delta distribution such that $K(0.5) = 1$. The cell volume distribution at cell birth is then given by

$$V_{birth} \sim \Psi_b(V) = 2\Phi_m(2V) = \frac{2^{\alpha}(V-0.75)^{\alpha-1}(2.5-2V)^{\beta-1}}{B(\alpha,\beta)}, \qquad V \in [0.75, 1.5]. \tag{S8.3}$$

The relationship between the mean of both distributions is given by

$$\langle V_{division} \rangle = \langle V_{birth} \rangle \cdot e^{\mu \langle T \rangle}, \tag{S8.4}$$

with $T = \ln(2)/\mu$ thus

$$\langle V_{birth} \rangle = \frac{\langle V_{division} \rangle}{e^{\ln 2}} = \frac{1}{2} \langle V_{division} \rangle. \tag{S8.5}$$

The volume distribution of a sample of extant cells can be calculated as function of Ψ_b and Φ_m following the equations deduced by Collins and Richmond (1962) as shown in Painter and Marr (1968):

$$g'(V)\lambda_e(V) + \lambda_e'(V)g(V) = k[2\Psi_b(V) - \Phi_m(V) - \lambda_e(V)] \tag{S8.6}$$

with $g(x)$ as the growth rate of cells with size V, $\lambda_e(V)$ as volume distribution of a sample of extant cells, Ψ_b as the volume distribution of a sample of baby cells at birth, Φ_m as the volume distribution at division of a sample of mother cells, and k as the specific growth rate of the population. Assuming $g(V) = k \cdot V$, this function can be simplified to

$$\lambda'_e(V) = \frac{1}{V}[2\Psi_b(V) - \Phi_m(V) - 2\lambda_e(V)] \tag{S8.7}$$

which can be solved to give the volume distribution of a sample of extant cells.

Since $V_{division}$ and V_{birth} are independent, the distribution of the ratio $r = V_{division}/V_{birth}$ can be calculated from (Curtiss, 1941)

$$\frac{V_{division}}{V_{birth}} \sim l(r) = \int_{-\infty}^{+\infty} |V|\Psi_b(rV) \cdot \Phi_m(V)dx. \tag{S8.8}$$

By using the change-of-variable technique we can transform $l(r)$ and obtain the distribution of the interdivision time:

$$r = e^{\mu T}, \tag{S8.9}$$

$$f(T) = \left|\frac{\partial}{\partial T}\left(e^{\mu T}\right)\right| l(e^{\mu T}). \tag{S8.10}$$

Analytical solutions: Case Study 2 mRNA synthesis

In the second case study, we used a zero-order mRNA synthesis model in exponentially growing cells. With a deterministic interdivision time, the theoretical mRNA copy number distributions of this model are known (Schwabe and Bruggeman (2014)).

Average molecule copy numbers at birth and division

The mRNA copy number in a cell at a certain age (a) is given by

$$n(a) = n(0) + N(a), \tag{S8.11}$$

where $n(a)$ is the mRNA copy number at cell age a, $n(0)$ as the mRNA copy number obtained at cell division, and $N(a)$ as the mRNA molecules synthesized since the last cell division at age a. The following relationship holds for the averages:

$$\langle n(a)\rangle = \langle n(0)\rangle + \langle N(a)\rangle. \tag{S8.12}$$

When symmetric partitioning at the end of a cell cycle ($t = T$) is assumed, the average mRNA copy number at birth is half the average of mRNA copy number at division,

$$\langle n(T)\rangle = 2\langle n(0)\rangle. \tag{S8.13}$$

This means that cells double their average number of molecules during one cell cycle ($a = 0$ to $a = T$). The average amount of molecules produced at $t = T$ equals the average number molecules obtained at $t = 0$.

Poisson distributed molecule copy numbers at a specific age

The waiting time distribution of a first-order synthesis process is exponentially distributed, the time between consecutive events is then given by

$$t_s \sim g(t_s) = k_s \cdot e^{-t_s k_s}. \tag{S8.14}$$

The time to make N molecules follows a gamma distribution and can be derived from the N-th convolution of $g(t)$ using the generating function, which is the Laplace transform ($\mathcal{L}$)

$$\begin{aligned} G(\chi = N|t) &= \mathcal{L}^{-1}(\mathcal{L}(g(t))^N), \\ &= \frac{e^{-k_s t} k_s^N t^{n-1}}{\Gamma(N)}, \end{aligned} \tag{S8.15}$$

with $\Gamma[\cdot]$ as the incomplete gamma function. The probability to produce more than N molecules in time t equals

$$h(\chi \geq N|t) = \int_0^t G(\chi = N|t)dt. \tag{S8.16}$$

The probability mass function for the production of N molecules at time t is given by

$$\begin{aligned} h(\chi = N|t) &= h(\chi \geq N|t) - h(\chi \geq N+1|t), \\ &= \frac{e^{-k_s t}(k_s t)^N}{N!}, \end{aligned} \tag{S8.17}$$

which is a Poisson distribution with mean $k_s t$. Of course, time can be interchanged by age, $t = a$.

mRNA copy number distribution of a sample of extant cells

The molecule copy number distribution of a sample of extant cells was earlier determined by Schwabe and Bruggeman (2014) and is given by

$$\begin{aligned} p(n) &= \int_0^T u(a) \cdot p(n|a)da, \\ &= \int_0^T u(a) \cdot \frac{e^{-k_s(a+T)}(k_s(a+T))^n}{n!} da, \\ &= \frac{4(k_s T)^n (\Gamma[1+n, k_s T + \ln(2)] - \Gamma[1+n, 2k_s T + \ln(4)]) \ln(2)(k_s T + \ln(2))^{-1-n}}{n!}, \end{aligned} \tag{S8.18}$$

with $\Gamma[\cdot]$ as the incomplete gamma function, $u(a)$ as the cell age distribution for a population of cells with a deterministic interdivision time T, and k_s as the synthesis rate constant.

Analytical solutions: Case Study 3 mRNA synthesis and degradation

In the third case study, we considered a model consisting of a zero-order mRNA synthesis reaction and first order mRNA degradation reaction, in exponentially growing cells. When we assume a deterministic interdivision time, the theoretical molecule copy number distributions at a given age of this model are known (Schwabe and Bruggeman (2014)).

Poisson distributed molecule copy numbers at a specific age

With a synthesis and degradation reaction, the average mRNA copy number in a cell of a given age a depends on the mRNA copy number at $t = 0$ and the *net* synthesis until time a. For a linear model with an constant production and degradation rate the average copy number can be written as

$$\langle n(a) \rangle = \frac{\int_0^T \left(\langle n(0) \rangle e^{-k_d a} + \frac{k_s}{k_d}(1 - e^{-k_d a}) \right) da}{T}, \tag{S8.19}$$

where T is the (deterministic) interdivision time ($T = \ln(2)/\mu$) and $\langle n(0) \rangle$ as

$$\langle n(0) \rangle = \frac{(e^{k_d T} - 1)k_s}{(2e^{k_s T} - 1)k_d}. \tag{S8.20}$$

The copy number distribution at age a is given by a Poisson distribution (see Schwabe and Bruggeman (2014) for derivation),

$$n(a) \sim \text{Poisson}\Big[\kappa(a) + \frac{p(a)\frac{1}{2}\kappa(T)}{1 - \frac{1}{2}} p(T)\Big], \tag{S8.21}$$

with the number of molecules produced during a cell cycle as up until age a as $\kappa(a) = \frac{k_s}{k_d}(1 - e^{-k_d a})$ and the survival probability of molecules as $p(a) = e^{k_d a}$.

Molecule copy-number distribution of a sample of extant cells

The molecule copy-number distribution of the sample of extant cells was determined by numerically solving

$$n \sim \int_0^T p(n|a) \cdot u(a) da, \tag{S8.22}$$

with the $p(n|a)$ determined by Eq. (S8.21) and the age distribution, $u(a)$, taken from Painter and Marr (1968) in combination with a deterministic interdivision time, which then for $u(a)$ gives

$$u(a) = 2\mu \cdot e^{-\mu \cdot a}, \qquad a \in [0, \frac{\ln(2)}{\mu}]. \tag{S8.23}$$

From a single lineage simulation to extant-cell population distributions

In Eq. (8.11) we derived the probability to choose daughter one such that the simulated lineage is representative for a sample of mother cells. Another way to derive this probability is via the following route: The two interdivision times that are calculated each generation are from the distribution $f_b(t)$. As we want to simulate a lineage representative of a sample of mother cells, the probability of choosing T_1 or T_2, and correspondingly of daughter cell one or daughter cell two should be biased in order to reflect $f_m(t)$ instead of $f_b(t)$. This can be done in the following way:

$$\frac{p(T_1)}{p(T_2)} = \lim_{\epsilon\to 0} \frac{\int_{T_2-\epsilon}^{T_2} f_b(t)dt}{\int_{T_1-\epsilon}^{T_1} f_b(t)dt} \cdot \frac{\int_{T_1-\epsilon}^{T_1} f_m(t)dt}{\int_{T_2-\epsilon}^{T_2} f_m(t)dt} = \frac{e^{-kT_1}}{e^{-kT_2}} = e^{k\Delta T}. \tag{S8.24}$$

The the probability p of choosing daughter one is then given by

$$p = \frac{p(T_1)}{p(T_1)+p(T_2)} = \frac{e^{k\cdot\Delta T}}{1+e^{k\cdot\Delta T}}, \tag{S8.25}$$

which is the same result as Eq. (8.11).

The joint distribution of cell age and interdivision time

In order to obtain the copy number distributions for a sample of extant cells the joint distribution of cell age and interdivision time for extant cells is required which can be calculated from the specific growth rate of the population and the interdivision time distribution. The derivation presented here is analogous to the one for the distribution of extant cell interdivision times in (Painter and Marr, 1968) (p. 528). With a population size of $n(0)$ at $t = 0$ the rate of formation of new cells equals $2k\cdot n(0)\cdot e^{kt}$. The fraction of these cells with interdivision time less than T which survive at least until time t is given by $\int_{-t}^{T} f_b(t)dt$. Therefore, the total number of cells at time t with interdivision time $< T$ and age $< a$ equals

$$\begin{aligned} \underbrace{n(0)\int_0^T\int_0^a g(t,x)dtdx}_{\substack{\#\text{ cells at } t=0 \text{ with age}< a\\ \text{and interdivision time}< T.}} &= \int_{-a}^{0}\underbrace{2k\cdot n(0)\cdot e^{kt}}_{\#\text{ cells with age}< a.} \underbrace{\int_{-t}^{T} f_b(x)dx}_{\substack{\text{Fraction of cells}\\ \text{with interdivi-}\\ \text{sion time}< T.}} dt, \\ &= 2n(0)\int_0^T f_b(x)\left(e^{kx}-e^{-ak}\right)dx. \end{aligned} \tag{S8.26}$$

Differentiation yields

$$g(a,T) = 2kf_b(T)e^{-ak} = kf_m(T)e^{k(T-a)}, \tag{S8.27}$$

the joint PDF of cell age and interdivision time.

Calculating copy number distributions at defined cell ages for samples of extant, mother, and baby cells

From the simulations (representative of a sample of mother cells) the copy number distributions at defined cell ages for samples of extant and baby cells can also be calculated (extracting the distribution for a sample of mother cells follows immediately from the trajectory):

$$p_s(n_x = n|a) = \int_0^{T_{max}} f_s(T) \frac{I(n, a, T)}{\sum_j I(j, a, T)} dT, \tag{S8.28}$$

where s denotes the type of sample (extant, baby cells) and f_s the corresponding interdivision time distribution (Eq. (8.6)).

Division is not a defined cell age (as interdivision times are distributed), but nevertheless the distribution at division for a sample of baby or extant cells can be calculated in a similar manner:

$$p_s(n_x = n|a = T) = \int_0^{T_{max}} f_s(T) \frac{I(n, a = T, T)}{\sum_j I(j, a = T, T)} dT. \tag{S8.29}$$

For both calculations interdivision times need to be binned in order to work with the indicator function.

Part IV

Epilogue

Discussion and prospects

9

9.1 Using the four key players in systems biology

We have seen that there is use for the four key players of this Ph.D. dissertation—models, simulation, algorithms, and software—within systems biology to study fluxes and fluctuations in biochemical models. This Ph.D. dissertation started with the notion that all models (as defined in Chapter 1) are wrong, but some are useful. We explained that these models are by definition a simplified description of reality and, therefore, always wrong. Realism is thus not the most desired property of any model. Useful models successfully capture the thin line between realism and simplicity.

To study fluxes and fluctuations in biochemical models, we used (genome-scale) **stoichiometric models** and **stochastic models**, respectively. These models are, as we showed in the preceding chapters, in many ways each other's opposite. While genome-scale stoichiometric models (GSSMs) consist generally of many reactions and species, stochastic models generally consists of only a few reactions and species. The predictions made by GSSMs are deterministic, whereas stochastic models possess inherent randomness. Kinetic parameters are not used in GSSMs, but they are a necessity for our stochastic models. Being each other's opposite means that, for simulation, they also rely on completely different algorithms (and require therefore typically different software tools).

Despite that both stoichiometric and stochastic models are a simplified description of the underlying process, they led to many valuable insights for a wide variety of biological questions. To study **fluxes** in biochemical models, we used **stoichiometric** models. These models have proven useful for metabolic engineering, biological discovery, and network property analysis (Feist and Palsson, 2008; O'Brien et al., 2015). Its proven usefulness and the relative ease of making these models led to a rapid increase in the reconstruction of these GSSMs of, among others, *Escherichia coli* (Feist et al., 2007), *Saccharomyces cerevisiae* (Heavner et al., 2013) and *Homo sapiens* metabolism (Thiele et al., 2013). We presented a state-of-the-art GSSM of the cyanobacterium *Synechocystis* metabolism in Chapter 3 and a corresponding interactive metabolic map. We subsequently used (also in Chapter 3) this GSSM to study (i) the physiological adaptation of *Synechocystis* to light and dark conditions, (ii) the properties of *Synechocystis* photosynthesis, and (iii) the production of the bulk chemical 2,3-butanediol. We greatly simplified this study by the development of a graphical and interactive map of *Synechocystis* metabolism. To study flux distributions, we can now just look at the graphical representation rather than the tedious process of studying a long list of difficult to understand identifiers and corresponding flux values. The availability of these graphical maps greatly simplifies the data analysis and integration challenge that we are currently facing with the increasing amount of genomic, metabolomic, proteomic, and flux data.

We have studied the possible flux distributions that give rise to production of 2,3-butanediol and its consequences (e.g. enhanced activity of the RuBisCO enzyme). We did not find any promising metabolic engineering strategies that predict obligatory coupling of growth and product synthesis under autotrophic growth conditions. These results are in agreement with a recent study (Erdrich et al., 2014) which indicates that achieving an obligatory coupling of growth and biofuel product synthesis in cyanobacteria requires more difficult and fundamentally different metabolic engineering strategies compared to heterotrophic micro-organisms (Klamt and Mahadevan, 2015). Perhaps that this is the reason why a heterologous xylose assimilation pathway was recently introduced into the cyanobacterium *Synechocystis* (Lee et al., 2015).

In Chapters 4 and 5 we showed that GSSMs typically have enormous (optimal) solution spaces. Kelk et al. (2012) proposed a method, CoPE-FBA, that offers the premise of a simplified biological understanding of the optimal solution space of metabolic models. In Chapters 4 and 5 we further developed this method and present its successor, CoPE-FBA 2.0, which allows for fast and efficient characterization of the optimal solution space. Our method is a great asset to the systems biology community, because it provides much more information about the optimal solution space than can be obtained with other popular methods such as Flux Variability Analysis (FVA) (Mahadevan and Schilling, 2003) and Monte Carlo sampling of the optimal solution space (Wiback et al., 2004; Bordel et al., 2010). It should be noted, however, that CoPE-FBA cannot be used on the solution space without optimization, while FVA and Monte Carlo sampling can be used on both the solution space and the optimal solution space. In the solution space, FVA can be used to predict the flux variability given a set of experimentally measured fluxes. Also, we have to keep in mind that the optimal solution space is a result of the underdetermined nature of the problem which raises an important question: Are these multiple solutions an artifact of the problem formulation or a biological property of the network? We can safely say that the enormous size of GSSM optimal solution spaces is an artifact of the problem formulation. Despite the popular proverb *"all roads lead to Rome"* which means that there exist many routes to the same goal, we expect that if we had all the details (e.g. kinetics, regulation) there would be only one optimal solution. The secondary objectives analysis done in Chapter 4 supports this expectation by showing that the optimal solution space shrinks enormously when additional objectives—e.g. those related to pathway expression costs—are introduced.

To study **fluctuations** in biochemical models, we used **stochastic** models—models that were until recently often overlooked. This is also why we created StochPy: A comprehensive and user-friendly tool for simulating stochastic biological processes. We subsequently used StochPy to study:

- stochastic simulation of a prokaryotic two-component signaling system (Wei et al., 2014);
- the spontaneous fluctuating activity of single enzyme molecules (Schwabe et al., 2013); and
- the histone modification pattern formation (Anink-Groenen et al., 2014).

In Chapters 6 and 7, we briefly discussed the work described in (Wei et al., 2014) and (Schwabe et al., 2013) to demonstrate the usefulness of discrete stochastic simulations and to illustrate our package StochPy, respectively. In Chapter 7 we already touched upon the potential effect of cell growth and division during stochastic simulations. We further improved the stochastic simulation algorithms with cell growth and division in Chapter 8. Taking cell growth and division into account is very time consuming if we simulate the complete lineage tree. We presented an algorithm which tracks only a single cell lineage and can, under certain realistic conditions, provide information about different samples of cells such as all cells in the lineage tree at a given point in time. The predictions of our algorithms are in agreement with theory and experimental measurements. This means that we can use our software now to make predictions of single-cell behavior in (rapidly) growing and dividing cells, especially in cases where the timescale to reach a steady state exceeds the organisms' generation time.

In conclusion, we succeeded in our aim: Use the four key players of this Ph.D. dissertation—models, simulation, algorithms, and software—to study fluxes and fluctuations in biochemical models. Of course, there are still many open questions in systems biology that remain to be solved. The models, simulations, algorithms, and software we present here form another important piece that can aid in solving some of these open questions in the future. For example, our stochastic simulator StochPy—that contains stochastic simulation algorithms with delay and with cell growth and division—can aid studying the largely unexplored effects of stochasticity at the level of single-cells and at the population level. Since this project was carried out within the BioSolarCells fundamental research program: "Expanding society's toolbox to harvest solar energy", we are particularly interested in the production of biofuels for which the cyanobacterium *Synechocystis* is an important candidate. Here, an important open question is: does *Synechocystis* have an optimal metabolic backbone for the production of biofuels? The production of biofuels could be mainly dependent on metabolic co-factors such as NADPH, NADH, and/or ATP. This would mean that rather than one metabolic backbone different intervention strategies are necessary. Our metabolic model and corresponding interactive map can directly aid in studying metabolic engineering strategies for different biofuels.

9.2 The future of the four key players in systems biology

I expect that the four key players of this Ph.D. dissertation—models, simulation, algorithms, and software—become increasingly important in the future of systems biology, but also in many other (scientific) communities. While I call them key players, they are merely the key players of this Ph.D. dissertation. I do not expect that they become the key players in systems biology, but they are going to be a core research component. In an ideal systems biology research world, both experimental and computational methods are used in an integrated approach, just as Kitano described systems biology in 2002 (Kitano, 2002a). Below, I discuss the future of each key player individually.

Currently, there exist two main modeling philosophies. On the one hand, models get updated, refined, and extended step-by-step with the ultimate goal of getting a whole-cell computational model as was recently done for the first time for the human pathogen *Mycoplasma genitalium* (Karr et al., 2012). This step-by-step procedure is probably best illustrated by the advancement of current genome-scale models that are extended reaction-by-reaction. These models have, however, also various shortcomings which is why scientists all over the world are developing the next generation of genome-scale models (King et al., 2015). A first shortcoming is that costs for growth and maintenance are largely ignored. Enzymes are for instance free, which means that two distinct pathways with the same input-output relationship, but with a different number of reactions (e.g. ten versus five reactions) have an identical enzyme production cost. A few years ago, the first genome-scale models of metabolism and gene expression (ME-models) were created (Lerman et al., 2012; Thiele et al., 2012). These ME-models require that all metabolic enzymes are synthesized by the gene expression network. Although this idea is promising, the process of enzyme synthesis was included in agonizing detail (transcription, translation, protein folding, protein complex formation etc. were included) which makes these models not transparent and (computationally) difficult to create, refine, use, and analyze. Another important shortcoming is that the kinetic limitations of enzymes are not or only poorly represented in current genome-scale models. The availability of high-throughput fluxomic and metabolic data paves the way to incorporate kinetics into genome-scale models. This also opens the door for specific growth rate optimization as was recently developed by Wortel et al. (2014) for genome-scale models.

On the other hand, there is a renewed interest in making models—used to study the often emergent properties of biological systems—as simple as possible (i.e. coarse-grained models). This reductionists approach is in accordance with George E. P. Box saying: *“the ability to devise simple but evocative models is the signature of the great scientist so overelaboration and overparameterization is often the mark of mediocrity”* (see Chapter 1). Coarse-grained models are especially popular in the field of molecular dynamics, because of molecular dynamics' computational challenges. These coarse-grained kinetic models have also been proven useful to study specific aspects of cell's physiology (Molenaar et al., 2009; Scott et al., 2014; Bosdriesz et al., 2015). Also, the stochastic models that we used are coarse-grained (e.g. by assuming that mRNA is synthesized instantly by one specific zero-order reaction).

I believe that both philosophies (very detailed and coarse-grained models) will continue to be used in the future of systems biology. Each model has its own advantages and disadvantages. The type of model used should depend on the type of research question and not the other way around. An important difference is that the development and refinement of these very detailed models is labor intensive and requires more and more a community effort as was recently illustrated for the human genome-scale model (Thiele et al., 2013).

While novel algorithms are continuously developed for existing models, the next generation of models to study biological systems also requires novel algorithms. This imposes new challenges. For example, various genome-scale modeling and simulation (M&S) approaches exist to study single species, but genome-scale M&S approaches to study communities of species is still in its infancy. It is therefore an open question of how to use predictive M&S in microbial communities. The development of these algorithms subsequently will require the development of new software or the extension of existing software.

Without useful software packages, the majority of the scientific community cannot profit from the developed computational models and algorithms. Or in other words, software packages can form the *bridge* between experimental and computational research. This, and the fact that software is also required to use the fancy experimental methods that exist these days and to analyze the data that is generated by these experimental methods, makes software indispensable in systems biology research. Strangely, those that develop software are still undervalued in the systems biology community. Creating useful software that can be used by the community is an art; useful software is not something that you create during a coffee break. Many software packages that are intended to be used by the community do contain bugs, but software is never a finished product. Instead of not using this software anymore (and perhaps make a quick and dirty "in-house" alternative), bugs should be reported such that the developers can fix these bugs and present a new and improved version of the software package.

Currently, the majority of the scientific software is not developed to be used by a community. This is true in the systems biology community but also in various other communities. Software is often developed and used within a research group to produce and/or analyze data. To make it even worse, many software projects are developed during a Ph.D. project and are not or poorly maintained afterwards. An important reason is that there are currently no good incentives for contributing to and improving existing software projects. Sharing software projects on GitHub could help. Alternatively, someone within the research facility should stay responsible for the developed software. Strangely, many research facilities have several wet-lab technicians, while dry-lab technicians are a rarity. These dry-lab technicians could be involved during the development of the project and could be responsible for the project after the main developer(s) leave the research facility.

To solve these issues with scientific software, we should come up with some guide lines about how to develop and how to use software packages. In the final section of this discussion, I present my view on how to develop useful (scientific) software that can form the bridge between experimental and computational research.

9.3 Most software works correctly, but some are useful

I started this Ph.D. dissertation with a famous quote of George E.P. Box who wrote: *"Essentially, all models are wrong, but some are useful."* We now reach the end of this Ph.D. dissertation and I want to end this discussion with a similar statement about software:

> Most software works correctly, but some are useful.

This statement can in my opinion can be made about the (scientific) software that is around these days. If you ever used scientific software, you probably remember the painful installation procedure that was required before being able to profit from the software package. Let alone those software packages where you did not manage to successfully complete the painful installation procedure at all. This is only one of the many examples of why scientific software, while working correctly, is not useful for the majority of the scientific community.

That is why I develop scientific software around four core principles:

(i) functionality,

(ii) user-friendliness,

(iii) flexibility, and

(iv) open source

which I discuss in detail here. For me the first prerequisite of scientific software is (of course) functionality. Scientific software can satisfy all the other core principles, but software cannot be useful if it does not function properly. This is not as easy as it may sound, because developing scientific software requires a complete understanding of the generally novel task that should be carried out by the software.

Second, I believe that scientific software should be user-friendly. Or in other words, software that functions correctly should also be easy-to-use. I define software as user-friendly if its installation procedure is simple, the software is well documented, contains simple-to-use and illustrative examples that work "out of the box". A graphical user interface (GUI) is not a prerequisite for user-friendly software. Admittedly, GUIs can be intuitive to use if they are properly designed, but scientific software packages with a properly designed GUI are a rarity.

Third, I propose that flexibility should be a requirement of scientific software. For software to be flexible, I argue that scientific software should (also) have a command-line interface (CLI). Compared to GUIs, CLIs have a steep learning curve but they allow for more flexibility, efficiency, and productivity. Not every programming language allows for a flexible CLI. I favor high-level programming languages that can be used interactively. Examples of high-level programming languages that can be used interactively are Python, MATLAB, Mathematica, Perl, and R. A disadvantage of (interactive) high-level programming languages is the lower speed at which simulations are done but this is—at least in computational systems biology—often not the most important feature of a programing language.

Additionally, software with (also) a CLI makes it generally simpler to reproduce the computational scientific findings presented in a scientific article. In contrast to "click-and-run" software (i.e. GUI only software), CLIs allow for preprogrammed scripts that contain a set of instructions that can—after installation—be used to directly reproduce the scientific findings presented in the scientific article. Publicly funded scientific research should be reproducible. The Nature Publishing Group, for instance, states: *"An inherent principle of publication is that others should be able to replicate and build upon the authors' published claims. A condition of publication in a Nature journal is that authors are required to make materials, data, code, and associated protocols promptly available to readers without undue qualifications."* However, this is generally not true for used software (and in-house developed computational methods). For example, many scientific findings obtained via M&S are still presented without the corresponding model. If the model is part of the publication, a set of instructions that is required to reproduce the results is typically lacking or incomplete. I advocate that scientific findings obtained via a computational method should also be reproducible, as is also argued by others (Peng, 2011; Ince et al., 2012; Morin et al., 2012; Bergmann et al., 2014; Hucka et al., 2015).

And fourth, I argue that scientific software should be open source. Closed-source software is used as "black boxes" whereas open-source software allows the community to inspect, modify and distribute the developed software for any purpose. We can therefore get a look under the hood. Open-source software also improves the chances of reproducibility (of which I discussed the importance above). Ideally, the programming language is also open-source and not proprietary such as MATLAB or Mathematica.

Even though I regularly used, among others, both MATLAB and Mathematica, I am a big fan of the programming language Python because it is simply awesome for most scientific software packages. Python is open source, indentation is obligatory which in combination with a proper syntax results in highly readable code; this in contrast to Perl and Mathematica. Many (scientific) libraries are available such as Numerical Python (NumPy), Scientific

Python (SciPy), MatPlotLib for plotting, Cython for low-level optimization, and Sphinx for automatic documentation of your software project. We can make use of IPython for interactive programming and since recently also IPython notebook which is a great asset for teaching duties and code sharing. For these reasons, we developed VoNDA, CoPE-FBA 2.0, and StochPy in Python—the three tools we developed to study fluxes and fluctuations in biochemical models.

Unquestionably, Python is—just as the software packages we developed—of course not perfect. The programming language C outperforms Python at both speed and memory usage, as demonstrated by Fourment and Gillings (2008) where they used both programming languages to execute various tasks that are common in systems biology and bioinformatics. If run time speed is key, C could be a better choice than Python, whereas we can also use the Cython library which gives you the combined power of Python and C. I end the discussion of my Ph.D. dissertation with the notion that no software is perfect, but that this is also not required:

> Remember that all software is imperfect; the practical question is how imperfect do they have to be to not be useful.

Bibliography

Aderem A (2005). Systems biology: its practice and challenges. *Cell*, 121(4):511–513.

Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, et al. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–3402.

Anderson DF (2007). A modified next reaction method for simulating chemical systems with time dependent propensities and delays. *J Chem Phys*, 127(21):214107.

Angermayr SA, Gorchs Rovira A, and Hellingwerf KJ (2015). Metabolic engineering of cyanobacteria for the synthesis of commodity products. *Trends Biotechnol*.

Angermayr SA, Hellingwerf KJ, Lindblad P, and de Mattos MJ (2009). Energy biotechnology with cyanobacteria. *Curr Opin Biotechnol*, 20(3):257–263.

Angermayr SA, Paszota M, and Hellingwerf KJ (2012). Engineering a cyanobacterial cell factory for production of lactic acid. *Appl Environ Microbiol*, 78(19):7098–7106.

Angermayr SA, van der Woude AD, Correddu D, Vreugdenhil A, Verrone V, et al. (2014). Exploring metabolic engineering design principles for the photosynthetic production of lactic acid by *Synechocystis* sp. PCC6803. *Biotechnol Biofuels*, 7:99.

Anink-Groenen LC, Maarleveld TR, Verschure PJ, and Bruggeman FJ (2014). Mechanistic stochastic model of histone modification pattern formation. *Epigenetics Chromatin*, 7(1):30.

Applegate DL, Cook W, Dash S, and Espinoza DG (2007). Exact solutions to linear programming problems. *Operations Research Letters*, 35(6):693 – 699. ISSN 0167-6377.

Arkin A, Ross J, and McAdams HH (1998). Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected *Escherichia coli* cells. *Genetics*, 149(4):1633–1648.

Atsumi S, Higashide W, and Liao JC (2009). Direct photosynthetic recycling of carbon dioxide to isobutyraldehyde. *Nat Biotechnol*, 27(12):1177–1180.

Balazsi G, van Oudenaarden A, and Collins JJ (2011). Cellular decision making and biological noise: from microbes to mammals. *Cell*, 144(6):910–925.

Bauwe H, Hagemann M, Kern R, and Timm S (2012). Photorespiration has a dual origin and manifold links to central metabolism. *Curr Opin Plant Biol*, 15(3):269–275.

Beg QK, Vazquez A, Ernst J, de Menezes MA, Bar-Joseph Z, et al. (2007). Intracellular crowding defines the mode and sequence of substrate uptake by *Escherichia coli* and constrains its metabolic activity. *Proc Natl Acad Sci USA*, 104(31):12663–12668.

Bekker M, Alexeeva S, Laan W, Sawers G, Teixeira de Mattos J, et al. (2010). The ArcBA two-component system of *Escherichia coli* is regulated by the redox state of both the ubiquinone and the menaquinone pool. *J Bacteriol*, 192(3):746–754.

Berg OG and von Hippel PH (1985). Diffusion-controlled macromolecular interactions. *Annu Rev Biophys Biophys Chem*, 14:131–160.

Berger B, Peng J, and Singh M (2013). Computational solutions for omics data. *Nat Rev Genet*, 14(5):333–346.

Bergmann FT, Adams R, Moodie S, Cooper J, Glont M, et al. (2014). COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project. *BMC Bioinformatics*, 15:369.

Bernard T, Bridge A, Morgat A, Moretti S, Xenarios I, et al. (2012). Reconciliation of metabolites and biochemical reactions for metabolic networks. *Brief Bioinformatics*.

Bilu Y, Shlomi T, Barkai N, and Ruppin E (2006). Conservation of expression and sequence of metabolic genes is reflected by activity across metabolic states. *PLoS Comput Biol*, 2(8):e106.

Blank LM, Kuepfer L, and Sauer U (2005). Large-scale 13C-flux analysis reveals mechanistic principles of metabolic network robustness to null mutations in yeast. *Genome Biol*, 6(6):R49.

Boele J, Olivier BG, and Teusink B (2012). FAME, the Flux Analysis and Modeling Environment. *BMC Syst Biol*, 6:8.

Bordel S, Agren R, and Nielsen J (2010). Sampling the solution space in genome-scale metabolic networks reveals transcriptional regulation in key enzymes. *PLoS Comput Biol*, 6(7):e1000859.

Bornstein BJ, Keating SM, Jouraku A, and Hucka M (2008). LibSBML: an API library for SBML. *Bioinformatics*, 24(6):880–881.

Bosdriesz E, Molenaar D, Teusink B, and Bruggeman FJ (2015). How fast-growing bacteria robustly tune their ribosome concentration to approximate growth-rate maximization. *FEBS J*, 282(10):2029–2044.

Box GE and Draper NR (1987). *Empirical model-building and response surfaces.* John Wiley & Sons.

Branco dos Santos F, de Vos WM, and Teusink B (2013). Towards metagenome-scale models for industrial applications–the case of Lactic Acid Bacteria. *Curr Opin Biotechnol*, 24(2):200–206.

Burgard AP, Nikolaev EV, Schilling CH, and Maranas CD (2004). Flux coupling analysis of genome-scale metabolic network reconstructions. *Genome Res*, 14(2):301–312.

Burgard AP, Pharkya P, and Maranas CD (2003). Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnol Bioeng*, 84(6):647–657.

Butcher EC, Berg EL, and Kunkel EJ (2004). Systems biology in drug discovery. *Nat Biotechnol*, 22(10):1253–1259.

Cai SJ and Inouye M (2002). EnvZ-OmpR interaction and osmoregulation in *Escherichia coli*. *J Biol Chem*, 277(27):24155–24161.

Cai X (2007). Exact stochastic simulation of coupled chemical reactions with delays. *J Chem Phys*, 126(12):124108.

Canfield DE, Glazer AN, and Falkowski PG (2010). The evolution and future of Earth's nitrogen cycle. *Science*, 330(6001):192–196.

Cao Y, Gillespie DT, and Petzold LR (2006). Efficient step size selection for the tau-leaping simulation method. *J Chem Phys*, 124(4):044109.

Chisti Y (2007). Biodiesel from microalgae. *Biotechnol Adv*, 25(3):294–306.

Collins JF and Richmond MH (1962). Rate of growth of *Bacillus cereus* between divisions. *J Gen Microbiol*, 28:15–33.

Collins SB, Reznik E, and Segrè D (2012). Temporal expression-based analysis of metabolism. *PLoS Comput Biol*, 8(11):e1002781.

Cornish-Bowden A (2013). *Fundamentals of enzyme kinetics.* John Wiley & Sons.

Curtiss J (1941). On the distribution of the quotient of two chance variables. *The Annals of Mathematical Statistics*, 12(4):409–421.

Deutscher D, Meilijson I, Kupiec M, and Ruppin E (2006). Multiple knockout analysis of genetic robustness in the yeast metabolic network. *Nat Genet*, 38(9):993–998.

Dismukes GC, Klimov VV, Baranov SV, Kozlov YN, DasGupta J, et al. (2001). The origin of atmospheric oxygen on Earth: the innovation of oxygenic photosynthesis. *Proc Natl Acad Sci USA*, 98(5):2170–2175.

Dobrzynski M and Bruggeman FJ (2009). Elongation dynamics shape bursty transcription and translation. *Proc Natl Acad Sci USA*, 106(8):2583–2588.

Ducat DC, Way JC, and Silver PA (2011). Engineering cyanobacteria to generate high-value products. *Trends Biotechnol*, 29(2):95–103.

Ebrahim A, Lerman JA, Palsson BØ, and Hyduke DR (2013). COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Syst Biol*, 7:74.

Edwards JS and Palsson BØ (1999). Systems properties of the *Haemophilus influenzae* Rd metabolic genotype. *J Biol Chem*, 274(25):17410–17416.

Edwards JS and Palsson BØ (2000). The *Escherichia coli* MG1655 *in silico* metabolic genotype: its definition, characteristics, and capabilities. *Proc Natl Acad Sci USA*, 97(10):5528–5533.

Eisenhut M, Ruth W, Haimovich M, Bauwe H, Kaplan A, et al. (2008). The photorespiratory glycolate metabolism is essential for cyanobacteria and might have been conveyed endosymbiontically to plants. *Proc Natl Acad Sci USA*, 105(44):17199–17204.

Eldar A and Elowitz MB (2010). Functional roles for noise in genetic circuits. *Nature*, 467(7312):167–173.

Elowitz MB, Levine AJ, Siggia ED, and Swain PS (2002). Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186.

English BP, Min W, van Oijen AM, Lee KT, Luo G, et al. (2006). Ever-fluctuating single enzyme molecules: Michaelis-Menten equation revisited. *Nat Chem Biol*, 2(2):87–94.

Erdrich P, Knoop H, Steuer R, and Klamt S (2014). Cyanobacterial biofuels: new insights and strain design strategies revealed by computational modeling. *Microb Cell Fact*, 13:128.

Erhard F, Friedel CC, and Zimmer R (2008). FERN - a Java framework for stochastic simulation and evaluation of reaction networks. *BMC Bioinformatics*, 9:356.

Evans TW, Gillespie CS, and Wilkinson DJ (2008). The SBML discrete stochastic models test suite. *Bioinformatics*, 24(2):285–286.

Famili I and Palsson BØ (2003). The convex basis of the left null space of the stoichiometric matrix leads to the definition of metabolically meaningful pools. *Biophys J*, 85(1):16–26.

Feist AM, Henry CS, Reed JL, Krummenacker M, Joyce AR, et al. (2007). A genome-scale metabolic reconstruction for *Escherichia coli* K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. *Mol Syst Biol*, 3:121.

Feist AM and Palsson BØ (2008). The growing scope of applications of genome-scale metabolic reconstructions using *Escherichia coli*. *Nat Biotechnol*, 26(6):659–667.

Feist AM and Palsson BØ (2010). The biomass objective function. *Curr Opin Microbiol*, 13(3):344–349.

Fell DA and Small JR (1986). Fat synthesis in adipose tissue. An examination of stoichiometric constraints. *Biochem J*, 238(3):781–786.

Fourment M and Gillings MR (2008). A comparison of common programming languages used in bioinformatics. *BMC Bioinformatics*, 9:82.

Fukuda K (1997). cdd/cdd+ reference manual. *Institute for Operations Research, ETH-Zentrum*.

Funahashi A, Tanimura N, Morohashi M, and Kitano H (2003). CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *BIOSILICO*, 1:159–162.

Gagneur J and Klamt S (2004). Computation of elementary modes: a unifying framework and the new binary approach. *BMC Bioinformatics*, 5:175.

Gao Z, Zhao H, Li Z, Tan X, and Lu X (2012). Photosynthetic production of ethanol from carbon dioxide in genetically engineered cyanobacteria. *Energy Environ Sci*, 5:9857–9865.

Gardiner C (1997). *Stochastic methods*. Springer-Verlag, Berlin–Heidelberg–New York–Tokyo.

Gauges R, Rost U, Sahle S, and Wegner K (2006). A model diagram layout extension for SBML. *Bioinformatics*, 22(15):1879–1885.

Ghosh S, Matsuoka Y, Asai Y, Hsin KY, and Kitano H (2011). Software for systems biology: from tools to integrated platforms. *Nat Rev Genet*, 12(12):821–832.

Gibson MA and Bruck J (2000). Efficient exact stochastic simulation of chemical systems with many species and many channels. *J Phys Chem A*, 104(9):1876–1889.

Gillespie DT (1976). A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403 – 434. ISSN 0021-9991.

Gillespie DT (1977). Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361.

Gillespie DT (2001). Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733.

Gillespie DT (2007). Stochastic simulation of chemical kinetics. *Annu Rev Phys Chem*, 58(1):35–55.

Gillespie DT and Petzold LR (2003). Improved leap-size selection for accelerated stochastic simulation. *The Journal of Chemical Physics*, 119(16):8229–8234.

Goffin P, van de Bunt B, Giovane M, Leveau JH, Hoppener-Ogawa S, et al. (2010). Understanding the physiology of *Lactobacillus plantarum* at zero growth. *Mol Syst Biol*, 6:413.

Golding I, Paulsson J, Zawilski SM, and Cox EC (2005). Real-time kinetics of gene activity in individual bacteria. *Cell*, 123(6):1025–1036.

Gomez D, Marathe R, Bierbaum V, and Klumpp S (2014). Modeling stochastic gene expression in growing cells. *J Theor Biol*, 348:1–11.

Grabherr MG, Haas BJ, Yassour M, Levin JZ, Thompson DA, et al. (2011). Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat Biotechnol*, 29(7):644–652.

Grafahrend-Belau E, Schreiber F, Koschutzki D, and Junker BH (2009). Flux balance analysis of barley seeds: a computational approach to study systemic properties of central metabolism. *Plant Physiol*, 149(1):585–598.

Gregory JA, Becker EC, and Pogliano K (2008). *Bacillus subtilis* MinC destabilizes FtsZ-rings at new cell poles and contributes to the timing of cell division. *Genes Dev*, 22(24):3475–3488.

Grötschel M, Lovász L, and Schrijver A (1988). *Geometric algorithms and combinatorial optimization.* Springer-Verlag.

Gstaiger M and Aebersold R (2009). Applying mass spectrometry-based proteomics to genetics, genomics and network biology. *Nat Rev Genet*, 10(9):617–627.

Harvey RJ, Marr AG, and Painter PR (1967). Kinetics of growth of individual cells of *Escherichia coli* and *Azotobacter agilis*. *J Bacteriol*, 93(2):605–617.

Heavner BD, Smallbone K, Price ND, and Walker LP (2013). Version 6 of the consensus yeast metabolic network refines biochemical coverage and improves model performance. *Database (Oxford)*, 2013:bat059.

Henry CS, DeJongh M, Best AA, Frybarger PM, Linsay B, et al. (2010). High-throughput generation, optimization and analysis of genome-scale metabolic models. *Nat Biotechnol*, 28(9):977–982. ISSN 1087-0156.

Hernandez-Prieto MA and Futschik ME (2012). CyanoEXpress: A web database for exploration and visualisation of the integrated transcriptome of cyanobacterium *Synechocystis* sp. PCC6803. *Bioinformation*, 8(13):634–638.

Hofmeyr JHS (2001). Metabolic control analysis in a nutshell. In *Proceedings of the 2nd International Conference on systems Biology*, pp. 291–300.

Holzhütter HG (2004). The principle of flux minimization and its application to estimate stationary fluxes in metabolic networks. *Eur J Biochem*, 271(14):2905–2922. ISSN 1432-1033.

Hoops S, Sahle S, Gauges R, Lee C, Pahle J, et al. (2006). COPASI–a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067–3074.

Hornberg JJ, Bruggeman FJ, Westerhoff HV, and Lankelma J (2006). Cancer: a Systems Biology disease. *BioSystems*, 83(2-3):81–90.

Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, et al. (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531.

Hucka M, Nickerson DP, Bader GD, Bergmann FT, Cooper J, et al. (2015). Promoting Coordinated Development of Community-Based Information Standards for Modeling in Biology: The COMBINE Initiative. *Front Bioeng Biotechnol*, 3:19.

Huh D and Paulsson J (2011). Random partitioning of molecules at cell division. *Proc Natl Acad Sci USA*, 108(36):15004–15009.

Hunter JD (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering*, 9(3):90–95. ISSN 1521-9615.

Ibarra RU, Edwards JS, and Palsson BØ (2002). *Escherichia coli* K-12 undergoes adaptive evolution to achieve *in silico* predicted optimal growth. *Nature*, 420(6912):186–189.

Ideker T, Galitski T, and Hood L (2001). A new approach to decoding life: systems biology. *Annu Rev Genomics Hum Genet*, 2:343–372.

Ince DC, Hatton L, and Graham-Cumming J (2012). The case for open computer programs. *Nature*, 482(7386):485–488.

Jol SJ, Kummel A, Terzer M, Stelling J, and Heinemann M (2012). System-level insights into yeast metabolism by thermodynamic analysis of elementary flux modes. *PLoS Comput Biol*, 8(3):e1002415.

Jones E, Oliphant T, Peterson P, et al. (2001). SciPy: Open source scientific tools for Python.

Kahn SD (2011). On the future of genomic data. *Science*, 331(6018):728–729.

Kampen NV (1992). *Stochastic Processes in Chemistry and Physics*. North Holland, 2nd edition edition.

Kanehisa M and Goto S (2000). KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res*, 28(1):27–30.

Karr JR, Sanghvi JC, Macklin DN, Gutschow MV, Jacobs JM, et al. (2012). A whole-cell computational model predicts phenotype from genotype. *Cell*, 150(2):389–401.

Kelk SM, Olivier BG, Stougie L, and Bruggeman FJ (2012). Optimal flux spaces of genome-scale stoichiometric models are determined by a few subnetworks. *Sci Rep*, 2:580.

Kempe H, Schwabe A, Cremazy F, Verschure PJ, and Bruggeman FJ (2015). The volumes and transcript counts of single cells reveal concentration homeostasis and capture biological noise. *Mol Biol Cell*, 26(4):797–804.

Kholodenko BN, Sauro HM, and Westerhoff HV (1994). Control by enzymes, coenzymes and conserved moieties. A generalisation of the connectivity theorem of metabolic control analysis. *Eur J Biochem*, 225(1):179–186.

Kierzek AM (2002). STOCKS: STOChastic Kinetic Simulations of biochemical systems with Gillespie algorithm. *Bioinformatics*, 18(3):470–481.

Kierzek AM, Zaim J, and Zielenkiewicz P (2001). The effect of transcription and translation initiation frequencies on the stochastic fluctuations in prokaryotic gene expression. *J Biol Chem*, 276(11):8165–8172.

King ZA, Lloyd CJ, Feist AM, and Palsson BØ (2015). Next-generation genome-scale models for metabolic engineering. *Curr Opin Biotechnol*, 35C:23–29.

Kitano H (2002a). Computational systems biology. *Nature*, 420(6912):206–210.

Kitano H (2002b). Systems biology: a brief overview. *Science*, 295(5560):1662–1664.

Kiviet DJ, Nghe P, Walker N, Boulineau S, Sunderlikova V, et al. (2014). Stochasticity of metabolism and growth at the single-cell level. *Nature*, 514(7522):376–379.

Klamt S, Gagneur J, and Von Kamp A (2005). Algorithmic approaches for computing elementary modes in large biochemical reaction networks. *Syst Biol (Stevenage)*, 152(4):249–255.

Klamt S and Mahadevan R (2015). On the feasibility of growth-coupled product synthesis in microbial strains. *Metab Eng*.

Klamt S and Stelling J (2003). Two approaches for metabolic pathway analysis? *Trends Biotechnol*, 21(2):64–69.

Knoop H, Zilliges Y, Lockau W, and Steuer R (2010). The metabolic network of *Synechocystis* sp. PCC 6803: systemic properties of autotrophic growth. *Plant Physiol*, 154(1):410–422.

Koch AL and Schaechter M (1962). A model for statistics of the cell division process. *J Gen Microbiol*, 29:435–454.

Kou SC, Cherayil BJ, Min W, English BP, and Xie XS (2005). Single-molecule Michaelis-Menten equations. *J Phys Chem B*, 109(41):19068–19081.

Kramer DM and Evans JR (2011). The importance of energy balance in improving photosynthetic productivity. *Plant Physiol*, 155(1):70–78.

Kreeger PK and Lauffenburger DA (2010). Cancer systems biology: a network modeling perspective. *Carcinogenesis*, 31(1):2–8.

Kumar A, Suthers PF, and Maranas CD (2012). MetRxn: a knowledgebase of metabolites and reactions spanning metabolic models and databases. *BMC Bioinformatics*, 13:6.

Lang M, Stelzer M, and Schomburg D (2011). BKM-react, an integrated biochemical reaction database. *BMC Biochem*, 12:42.

Le Novère N, Hucka M, Mi H, Moodie S, Schreiber F, et al. (2009). The Systems Biology Graphical Notation. *Nat Biotechnol*, 27(8):735–741.

Lee JM, Gianchandani EP, and Papin JA (2006). Flux balance analysis in the era of metabolomics. *Brief Bioinformatics*, 7(2):140–150.

Lee TC, Xiong W, Paddock T, Carrieri D, Chang IF, et al. (2015). Engineered xylose utilization enhances bio-products productivity in the cyanobacterium *Synechocystis* sp. PCC 6803. *Metab Eng*.

Lerman JA, Hyduke DR, Latif H, Portnoy VA, Lewis NE, et al. (2012). *In silico* method for modelling metabolism and gene product expression at genome scale. *Nat Commun*, 3:929.

Lewis NE, Hixson KK, Conrad TM, Lerman JA, Charusanti P, et al. (2010). Omic data from evolved *E. coli* are consistent with computed optimal growth from genome-scale models. *Mol Syst Biol*, 6:390.

Ley RE, Peterson DA, and Gordon JI (2006). Ecological and evolutionary forces shaping microbial diversity in the human intestine. *Cell*, 124(4):837–848.

Li GW and Xie XS (2011). Central dogma at the single-molecule level in living cells. *Nature*, 475(7356):308–315.

Lopez CF, Muhlich JL, Bachman JA, and Sorger PK (2013). Programming biological models in Python using PySB. *Mol Syst Biol*, 9:646.

Mahadevan R, Edwards JS, and Doyle FJ (2002). Dynamic flux balance analysis of diauxic growth in *Escherichia coli*. *Biophys J*, 83(3):1331–1340.

Mahadevan R and Schilling CH (2003). The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metab Eng*, 5(4):264–276.

Mazumdar V, Snitkin ES, Amar S, and Segrè D (2009). Metabolic network model of a human oral pathogen. *J Bacteriol*, 191(1):74–90.

McCloskey D, Palsson BØ, and Feist AM (2013). Basic and applied uses of genome-scale metabolic network reconstructions of *Escherichia coli*. *Mol Syst Biol*, 9:661.

McQuarrie D (1967). Stochastic approach to chemical kinetics. *Journal of Applied Probability*, 4:413.

Milo R, Jorgensen P, Moran U, Weber G, and Springer M (2010). BioNumbers–the database of key numbers in molecular and cell biology. *Nucleic Acids Res*, 38(Database issue):D750–753.

Molenaar D, van Berlo R, de Ridder D, and Teusink B (2009). Shifts in growth strategies reflect tradeoffs in cellular economics. *Mol Syst Biol*, 5:323.

Morin A, Urban J, Adams PD, Foster I, Sali A, et al. (2012). Research priorities. Shining light into black boxes. *Science*, 336(6078):159–160.

Müller A, Bruggeman F, Olivier B, and Stougie L (2014). Fast flux module detection using matroid theory. In *Research in Computational Molecular Biology*, vol. 8394 of *Lecture Notes in Computer Science*, pp. 192–206. Springer International Publishing. ISBN 978-3-319-05268-7.

Müller AC and Bockmayr A (2014). Flux modules in metabolic networks. *J Math Biol*, 69(5):1151–1179.

Myers CJ, Barker N, Jones K, Kuwahara H, Madsen C, et al. (2009). iBioSim: a tool for the analysis and design of genetic circuits. *Bioinformatics*, 25(21):2848–2849.

Nogales J, Gudmundsson S, Knight EM, Palsson BØ, and Thiele I (2012). Detailing the optimality of photosynthesis in cyanobacteria through systems biology analysis. *Proc Natl Acad Sci USA*, 109(7):2678–2683.

Nogales J, Gudmundsson S, and Thiele I (2013). Toward systems metabolic engineering in cyanobacteria: opportunities and bottlenecks. *Bioengineered*, 4(3):158–163.

Oberhardt MA, Palsson BØ, and Papin JA (2009). Applications of genome-scale metabolic reconstructions. *Mol Syst Biol*, 5:320.

O'Brien EJ, Monk JM, and Palsson BØ (2015). Using Genome-scale Models to Predict Biological Capabilities. *Cell*, 161(5):971–987.

Oliphant TE (2006). *A Guide to NumPy*, vol. 1. Trelgol Publishing USA.

Oliver JW, Machado IM, Yoneda H, and Atsumi S (2013). Cyanobacterial conversion of carbon dioxide to 2,3-butanediol. *Proc Natl Acad Sci USA*, 110(4):1249–1254.

Olivier B and Bergmann F (2013). Flux balance constraints, version 1 release 1. *Available from COMBINE.*

Olivier BG (2011). PySCeS CBMPy: Constraint Based Modelling in Python. http://cbmpy.sourceforge.net.

Olivier BG, Rohwer JM, and Hofmeyr JH (2005). Modelling cellular systems with PySCeS. *Bioinformatics*, 21(4):560–561.

Orth JD, Thiele I, and Palsson BØ (2010). What is flux balance analysis? *Nat Biotechnol*, 28(3):245–248.

Painter PR and Marr AG (1968). Mathematics of microbial populations. *Annu Rev Microbiol*, 22:519–548.

Palsson BØ (2006). *Systems Biology: Properties of Reconstructed Networks*. Cambridge university press.

Papin JA, Price ND, and Palsson BØ (2002). Extreme pathway lengths and reaction participation in genome-scale metabolic networks. *Genome Res*, 12(12):1889–1900.

Papin JA, Stelling J, Price ND, Klamt S, Schuster S, et al. (2004). Comparison of network-based pathway analysis methods. *Trends Biotechnol*, 22(8):400–405.

Peng RD (2011). Reproducible research in computational science. *Science*, 334(6060):1226–1227.

Peregrin-Alvarez JM, Sanford C, and Parkinson J (2009). The conservation and evolutionary modularity of metabolism. *Genome Biol*, 10(6):R63.

Pineda-Krch M (2008). GillespieSSA: Implementing the Gillespie Stochastic Simulation Algorithm in R. *J Stat Softw*, 25(12):1–18.

Poolman MG (2006). ScrumPy: metabolic modelling with Python. *Syst Biol (Stevenage)*, 153(5):375–378.

Price ND, Papin JA, and Palsson BØ (2002). Determination of redundancy and systems properties of the metabolic network of *Helicobacter pylori* using genome-scale extreme pathway analysis. *Genome Res*, 12(5):760–769.

Price ND, Reed JL, and Palsson BØ (2004). Genome-scale models of microbial cells: evaluating the consequences of constraints. *Nat Rev Microbiol*, 2(11):886–897.

Qian H (2008). Cooperativity and specificity in enzyme kinetics: a single-molecule time-based perspective. *Biophys J*, 95(1):10–17.

Quintana N, Van der Kooy F, Van de Rhee MD, Voshol GP, and Verpoorte R (2011). Renewable energy from Cyanobacteria: energy production optimization by metabolic pathway engineering. *Appl Microbiol Biotechnol*, 91(3):471–490.

Raj A and van Oudenaarden A (2008). Nature, nurture, or chance: stochastic gene expression and its consequences. *Cell*, 135(2):216–226.

Raman K and Chandra N (2009). Flux balance analysis of biological systems: applications and challenges. *Brief Bioinformatics*, 10(4):435–449.

Raman K, Rajagopalan P, and Chandra N (2005). Flux balance analysis of mycolic acid pathway: targets for anti-tubercular drugs. *PLoS Comput Biol*, 1(5):e46.

Ramsey S, Orrell D, and Bolouri H (2005). Dizzy: stochastic simulation of large-scale genetic regulatory networks. *J Bioinform Comput Biol*, 3(2):415–436.

Rathinam M, Petzold LR, Cao Y, and Gillespie DT (2003). Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *The Journal of Chemical Physics*, 119(24):12784–12794.

Reder C (1988). Metabolic control theory: a structural approach. *J Theor Biol*, 135(2):175–201.

Reed JL and Palsson BØ (2004). Genome-scale *in silico* models of *E. coli* have multiple equivalent phenotypic states: assessment of correlated reaction subsets that comprise network states. *Genome Res*, 14(9):1797–1805.

Reshes G, Vanounou S, Fishov I, and Feingold M (2008). Cell shape dynamics in *Escherichia coli*. *Biophys J*, 94(1):251–264.

Roels JA (2009). Application of macroscopic principles to microbial metabolism. *Biotechnol Bioeng*, 103(1):2–59.

Rohr C, Marwan W, and Heiner M (2010). Snoopy–a unifying Petri net framework to investigate biomolecular networks. *Bioinformatics*, 26(7):974–975.

Saha R, Verseput AT, Berla BM, Mueller TJ, Pakrasi HB, et al. (2012). Reconstruction and comparison of the metabolic potential of cyanobacteria *Cyanothece* sp. ATCC 51142 and *Synechocystis* sp. PCC 6803. *PLoS ONE*, 7(10):e48285.

Sanft KR, Wu S, Roh M, Fu J, Lim RK, et al. (2011). StochKit2: software for discrete stochastic simulation of biochemical systems with events. *Bioinformatics*, 27(17):2457–2458.

Santos F, Boele J, and Teusink B (2011). A practical guide to genome-scale metabolic models and their analysis. *Meth Enzymol*, 500:509–532.

Sauro HM and Ingalls B (2004). Conservation analysis in biochemical networks: computational issues for software writers. *Biophys Chem*, 109(1):1–15.

Savakis P and Hellingwerf KJ (2014). Engineering cyanobacteria for direct biofuel production from CO2. *Curr Opin Biotechnol*, 33:8–14.

Savakis P, Tan X, Du W, Branco dos Santos F, Lu X, et al. (2015). Photosynthetic production of glycerol by a recombinant cyanobacterium. *J Biotechnol*, 195:46–51.

Savakis PE, Angermayr SA, and Hellingwerf KJ (2013). Synthesis of 2,3-butanediol by *Synechocystis* sp. PCC6803 via heterologous expression of a catabolic pathway from lactic acid- and enterobacteria. *Metab Eng*, 20:121–130.

Savinell JM and Palsson BØ (1992). Optimal selection of metabolic fluxes for in vivo measurement. I. Development of mathematical methods. *J Theor Biol*, 155(2):201–214.

Schellenberger J, Park J, Conrad T, and Palsson BØ (2010). BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions. *BMC Bioinformatics*, 11(1):213.

Schellenberger J, Que R, Fleming RM, Thiele I, Orth JD, et al. (2011). Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. *Nat Protoc*, 6(9):1290–1307.

Schilling CH, Letscher D, and Palsson BØ (2000). Theory for the systemic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. *J Theor Biol*, 203(3):229–248.

Schlögl F (1972). Chemical reaction models for non-equilibrium phase transitions. *Zeitschrift für Physik*, 253(2):147–161. ISSN 0044-3328.

Schriek S, Aguirre-von Wobeser E, Nodop A, Becker A, Ibelings BW, et al. (2008). Transcript profiling indicates that the absence of PsbO affects the coordination of C and N metabolism in *Synechocystis* sp. PCC 6803. *Physiol Plant*, 133(3):525–543.

Schriek S, Kahmann U, Staiger D, Pistorius EK, and Michel KP (2009). Detection of an L-amino acid dehydrogenase activity in *Synechocystis* sp. PCC 6803. *J Exp Bot*, 60(3):1035–1046.

Schriek S, Ruckert C, Staiger D, Pistorius EK, and Michel KP (2007). Bioinformatic evaluation of L-arginine catabolic pathways in 24 cyanobacteria and transcriptional analysis of genes encoding enzymes of L-arginine catabolism in the cyanobacterium *Synechocystis* sp. PCC 6803. *BMC Genomics*, 8:437.

Schuetz R, Kuepfer L, and Sauer U (2007). Systematic evaluation of objective functions for predicting intracellular fluxes in *Escherichia coli*. *Mol Syst Biol*, 3:119.

Schuetz R, Zamboni N, Zampieri M, Heinemann M, and Sauer U (2012). Multidimensional optimality of microbial metabolism. *Science*, 336(6081):601–604.

Schuster S, Dandekar T, and Fell DA (1999). Detection of elementary flux modes in biochemical networks: a promising tool for pathway analysis and metabolic engineering. *Trends Biotechnol*, 17(2):53–60.

Schwabe A and Bruggeman FJ (2014). Contributions of cell growth and biochemical reactions to nongenetic variability of cells. *Biophys J*, 107(2):301–313.

Schwabe A, Maarleveld TR, and Bruggeman FJ (2013). Exploration of the spontaneous fluctuating activity of single enzyme molecules. *FEBS Lett*, 587(17):2744–2752.

Scott M, Klumpp S, Mateescu EM, and Hwa T (2014). Emergence of robust growth laws from optimal regulation of ribosome synthesis. *Mol Syst Biol*, 10:747.

Segrè D, Vitkup D, and Church GM (2002). Analysis of optimality in natural and perturbed metabolic networks. *Proc Natl Acad Sci USA*, 99(23):15112–15117.

Shahrezaei V and Swain PS (2008). Analytical distributions for stochastic gene expression. *Proc Natl Acad Sci USA*, 105(45):17256–17261.

Shastri AA and Morgan JA (2005). Flux balance analysis of photoautotrophic metabolism. *Biotechnol Prog*, 21(6):1617–1626.

Shaw DE, Maragakis P, Lindorff-Larsen K, Piana S, Dror RO, et al. (2010). Atomic-level characterization of the structural dynamics of proteins. *Science*, 330(6002):341–346.

Shlomi T, Benyamini T, Gottlieb E, Sharan R, and Ruppin E (2011). Genome-scale metabolic modeling elucidates the role of proliferative adaptation in causing the Warburg effect. *PLoS Comput Biol*, 7(3):e1002018.

Siso-Nadal F, Ollivier JF, and Swain PS (2007). Facile: a command-line network compiler for systems biology. *BMC Syst Biol*, 1:36.

Smoot ME, Ono K, Ruscheinski J, Wang PL, and Ideker T (2011). Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, 27(3):431–432.

Snitkin ES, Dudley AM, Janse DM, Wong K, Church GM, et al. (2008). Model-driven analysis of experimentally determined growth phenotypes for 465 yeast gene deletion mutants under 16 different conditions. *Genome Biol*, 9(9):R140.

Stelling J, Klamt S, Bettenbrock K, Schuster S, and Gilles ED (2002). Metabolic network structure determines key aspects of functionality and regulation. *Nature*, 420(6912):190–193.

Stock AM, Robinson VL, and Goudreau PN (2000). Two-component signal transduction. *Annu Rev Biochem*, 69:183–215.

Stolyar S, Van Dien S, Hillesland KL, Pinel N, Lie TJ, et al. (2007). Metabolic modeling of a mutualistic microbial community. *Mol Syst Biol*, 3:92.

Szabo A and Merks RM (2013). Cellular potts modeling of tumor growth, tumor invasion, and tumor evolution. *Front Oncol*, 3:87.

Terzer M (2009). Polco: A Java tool to compute extreme rays of polyhedral cones.

Teusink B and Smid EJ (2006). Modelling strategies for the industrial exploitation of lactic acid bacteria. *Nat Rev Microbiol*, 4(1):46–56.

Teusink B, Wiersma A, Molenaar D, Francke C, de Vos WM, et al. (2006). Analysis of growth of *Lactobacillus plantarum* WCFS1 on a complex medium using a genome-scale metabolic model. *J Biol Chem*, 281(52):40041–40048.

Thiele I, Fleming RM, Que R, Bordbar A, Diep D, et al. (2012). Multiscale modeling of metabolism and macromolecular synthesis in *E. coli* and its application to the evolution of codon usage. *PLoS ONE*, 7(9):e45635.

Thiele I and Palsson BØ (2010). A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nat Protoc*, 5(1):93–121.

Thiele I, Swainston N, Fleming RM, Hoppe A, Sahoo S, et al. (2013). A community-driven global reconstruction of human metabolism. *Nat Biotechnol*, 31(5):419–425.

Trinh CT, Unrean P, and Srienc F (2008). Minimal *Escherichia coli* cell for the most efficient production of ethanol from hexoses and pentoses. *Appl Environ Microbiol*, 74(12):3634–3643.

Tsukanov R, Reshes G, Carmon G, Fischer-Friedrich E, Gov NS, et al. (2011). Timing of Z-ring localization in *Escherichia coli*. *Phys Biol*, 8(6):066003.

Unden G and Bongaerts J (1997). Alternative respiratory pathways of *Escherichia coli*: energetics and transcriptional regulation in response to electron acceptors. *Biochim Biophys Acta*, 1320(3):217–234.

van der Woude AD, Angermayr SA, Puthan Veetil V, Osnato A, and Hellingwerf KJ (2014). Carbon sink removal: Increased photosynthetic production of lactic acid by *Synechocystis* sp. PCC6803 in a glycogen storage mutant. *J Biotechnol*, 184:100–102.

van Heerden JH, Wortel MT, Bruggeman FJ, Heijnen JJ, Bollen YJ, et al. (2014). Lost in transition: start-up of glycolysis yields subpopulations of nongrowing cells. *Science*, 343(6174):1245114.

van Hoek MJ and Merks RM (2012). Redox balance is key to explaining full vs. partial switching to low-yield metabolism. *BMC Syst Biol*, 6:22.

Vellela M and Qian H (2009). Stochastic dynamics and non-equilibrium thermodynamics of a bistable chemical system: the Schlögl model revisited. *J R Soc Interface*, 6(39):925–940.

Wagner C and Urbanczik R (2005). The geometry of the flux cone of a metabolic network. *Biophys J*, 89(6):3837–3845.

Wang Z, Gerstein M, and Snyder M (2009). RNA-Seq: a revolutionary tool for transcriptomics. *Nat Rev Genet*, 10(1):57–63.

Wei K, Moinat M, Maarleveld TR, and Bruggeman FJ (2014). Stochastic simulation of prokaryotic two-component signalling indicates stochasticity-induced active-state locking and growth-rate dependent bistability. *Mol Biosyst*, 10(9):2338–2346.

Werner HM, Mills GB, and Ram PT (2014). Cancer Systems Biology: a peek into the future of patient care? *Nat Rev Clin Oncol*, 11(3):167–176.

Wiback SJ, Famili I, Greenberg HJ, and Palsson BØ (2004). Monte Carlo sampling can be used to determine the size and shape of the steady-state flux space. *J Theor Biol*, 228(4):437–447.

Wiback SJ and Palsson BØ (2002). Extreme pathway analysis of human red blood cell metabolism. *Biophys J*, 83(2):808–818.

Wijffels RH, Kruse O, and Hellingwerf KJ (2013). Potential of industrial biotechnology with cyanobacteria and eukaryotic microalgae. *Curr Opin Biotechnol*, 24(3):405–413.

Wilkinson DJ (2011). *Stochastic modelling for systems biology*. CRC press.

Wortel MT, Peters H, Hulshof J, Teusink B, and Bruggeman FJ (2014). Metabolic states with maximal specific rate carry flux through an elementary flux mode. *FEBS J*, 281(6):1547–1555.

Xu J and Gordon JI (2003). Honor thy symbionts. *Proc Natl Acad Sci USA*, 100(18):10452–10459.

Yim H, Haselbeck R, Niu W, Pujol-Baxley C, Burgard A, et al. (2011). Metabolic engineering of *Escherichia coli* for direct production of 1,4-butanediol. *Nat Chem Biol*, 7(7):445–452.

Zhang S and Bryant DA (2011). The tricarboxylic acid cycle in cyanobacteria. *Science*, 334(6062):1551–1553.

Zomorrodi AR and Maranas CD (2012). OptCom: a multi-level optimization framework for the metabolic modeling and analysis of microbial communities. *PLoS Comput Biol*, 8(2):e1002363.

Zwicker D, Lubensky DK, and ten Wolde PR (2010). Robust circadian clocks from coupled protein-modification and transcription-translation cycles. *Proc Natl Acad Sci USA*, 107(52):22540–22545.

Summary

Biology, computer science, and mathematics, as independent sciences, deal with completely different aspects. Biology studies living organisms. Computer science deals with the construction, programming, operation, and use of computers. Mathematics is the study of number, quantity, and space which can be applied to different disciplines. Computational systems biology is a fast growing interdisciplinary discipline that uses computer science and (applied) mathematics to study complex biological systems.

Building a mathematical model is an attractive and effective way of using computational systems biology to study the emergent properties of a complex biological system. A computer simulation uses such a mathematical model to predict its behavior in a specific condition. These kind of simulations require following a set of mathematical operations—an algorithm. The used models and algorithms are often so complex that appropriate software is required for this purpose.

I underlined mathematical model, computer simulation, algorithm, and software for a reason. In Chapter 1 I explained that they are the key players I used to study *fluxes and fluctuations in biochemical models* (the title of this dissertation). More specifically, I used two distinct simulation approaches to study fluxes and fluctuations in biochemical models: stoichiometric simulations to study fluxes (Chapters 2–5) and stochastic simulations to study fluctuations (Chapters 6–8). Briefly, stoichiometric simulations are deterministic (the outcome is precisely determined without any room of variation). They are called stoichiometric simulations because kinetic information is ignored. In contrast, stochastic simulations take into account this kinetic information and they posses inherent randomness (the outcome of each simulation is different). This means that both simulation approaches are completely different; both approaches require also different models. This dissertation is therefore logically divided into different parts, a part about stoichiometric models and simulations and a part about stochastic models and simulations.

The part about fluxes in biochemical models starts with Chapter 2. Here we gave an overview of the state-of-the-art stoichiometric modeling and simulation approaches. We additionally discussed the biological implications of these approaches. Stoichiometric modeling and simulation is mainly used to study metabolism. This requires genome-scale stoichiometric models that cover the total metabolic potential encoded in the genome. We developed such a genome-scale stoichiometric metabolic model of the cyanobacterium *Synechocystis* (Chapter 3). There is a lot of interest in these cyanobacteria because they have the potential to be used as a“cellular factory“ for the conversion of solar energy and carbon dioxide into various industrial products such as biofuels.

Simulating these genome-scale models often results in big data sets. The interpretation of these big data sets a challenge. To be of any biological value, interpretation of these big data sets is required in terms of biological routes; these routes can be compared with highways. Road maps and navigation systems are widely used to travel from starting point to destination. To simplify the interpretation of the data sets generated by simulating genome-scale models, we developed a hand-drawn metabolic map (Chapter 3). Our metabolic map can be read and modified by machines and interpreted by researchers. Therefore, this metabolic map can be used, for example, for the direct visualization and analysis of the (big) data sets that are generated with stoichiometric simulations.

Genome-scale stoichiometric models are often used to determine the optimal biomass yield. Perhaps surprisingly, there are typically multiple biological routes (and combinations thereof) with an optimal biomass yield. This gives rise to the optimal solution space. In Chapter 4 we extensively studied the characteristics of this space. We are the first that uniquely characterized the entire optimal solution space in terms of a specific set of metabolic flux routes. We can quickly characterize this set of metabolic flux routes with the TimoTimo (this name is derived from the navigation systems manufacturer TomTom). The TimoTimo uses a divide-and-conquer approach (Chapter 5) to quickly characterize the optimal solution space in terms of this specific set of metabolic flux routes.

Stoichiometric simulations are, as mentioned earlier, deterministic. They ignore that biological systems are inherently stochastic. While this stochasticity is often negligible in the macroscopic world because of the law of large numbers, experimental evidence indicates that significant stochastic fluctuations are present and essential. This is especially true when molecules (e.g. mRNA or transcription factors) are present at low copy numbers. There exist mathematical methods that capture these stochastic fluctuations, which are called stochastic simulation algorithms.

The application and development of these algorithms to study fluctuations in biochemical models is the topic of the second part of my dissertation. This part starts with Chapter 6 where we provided an overview of the state-of-the-art stochastic simulation algorithms. We also used several illustrative examples to demonstrate the advantage of stochastic simulations compared to deterministic simulations. For instance, stochastic simulations allow for better estimation of the kinetic model parameters.

To study fluctuations in biochemical models we developed StochPy (Chapter 7). StochPy is a comprehensive and user-friendly software package for stochastic simulations. In Chapter 7 we used various case studies to demonstrate the power of StochPy. The current state-of-the-art stochastic simulation algorithms ignore the, often important, stochastic contributions of both cell growth and division. In Chapter 8 we developed a generic stochastic simulation algorithm that takes into account the stochastic contributions of net molecule synthesis as the stochastic contributions of cell growth and division. This novel algorithm is now implemented in StochPy and is in exact agreement with theory, and in good agreement with time-lapse microscopy data of growing micro-organisms.

In a concluding discussion (Chapter 9), I looked critically at the use and future of the key players (model, simulation, algorithm, and software) in systems biology. I expect that they are going to be a core research component to study complex biological systems. This discussion ends with my philosophy about the requirements for useful scientific software.

Samenvatting

Biologie, informatica en wiskunde zijn wetenschappen die gaan over compleet verschillende aspecten. Biologie bestudeert levende organismen. Informatica is het vakgebied van de digitale informatieverwerking. Wiskunde is de studie van onder andere getallen, hoeveelheden en ruimtes die je kunt toepassen in andere wetenschappen. Computationele systeembiologie is een snel groeiende interdisciplinaire wetenschap die informatica en (toegepaste) wiskunde gebruikt voor het onderzoeken van complexe biologische systemen.

Het ontwikkelen van een wiskundig model is een attractieve en effectieve methode voor het onderzoeken van een complex biologisch systeem. Met behulp van een computersimulatie kun je zo'n model vervolgens gebruiken om het gedrag van een biologisch systeem—in een specifieke situatie—te voorspellen. Dit soort simulaties voeren een set van wiskundige instructies uit, oftewel een algoritme. De gebruikte modellen en algoritmes zijn vaak zo complex, zodat geschikte software noodzakelijk is.

Het is geen willekeur dat ik zowel wiskundig model, computersimulatie, algoritme en software onderstreept heb. In hoofdstuk 1 heb ik uitgelegd dat dit de belangrijkste spelers zijn in mijn proefschrift voor het onderzoeken van *fluxen en fluctuaties in biochemische modellen* (de titel van mijn proefschrift). Ik heb twee verschillende soorten simulaties gebruikt voor het onderzoeken van fluxen en fluctuaties in biochemische modellen: stoichiometrische simulaties voor het onderzoeken van fluxen (hoofdstukken 2–5) en stochastische simulaties voor het onderzoeken van fluctuaties (hoofdstukken 6–8). Een stoichiometrische simulatie is deterministisch (een simulatie zonder random variabelen; de uitkomst van de simulatie is iedere keer hetzelfde). Het wordt een stoichiometrische simulatie genoemd, omdat de kinetiek wordt weglaten. Stochastische simulaties daarentegen nemen deze kinetiek mee en laten een vorm van willekeur toe (een simulatie met random variabelen; de uitkomst van de simulatie is telkens anders). Dit betekent dat beide soorten simulaties compleet verschillend zijn; je hebt daarom verschillende modellen nodig voor stoichiometrische en stochastische simulaties. Dit proefschrift is daarom logischerwijs onderverdeeld in verschillende delen: een deel over stoichiometrische modellen en simulaties en een deel over stochastische modellen en simulaties.

Het deel over fluxen in biochemische modellen start met hoofdstuk 2. Hierin hebben we de huidige stand van zaken voor stoichiometrische modellen en simulatiemethodes besproken. In dit hoofdstuk hebben we ook de biologische implicaties van deze modellen en simulatiemethodes besproken. Stoichiometrische modellen en simulatiemethodes worden met name gebruikt voor het onderzoeken van het metabolisme (stofwisseling). Hiervoor worden stoichiometrische modellen van het metabolisme op genoom-schaal gebruikt; dit betekent dat er rekening gehouden wordt met het totale metabole potentieel dat is gecodeerd in het genoom. Wij hebben zo'n genoom-schaal stoichiometrisch model ontwikkeld over het metabolisme van de cyanobacterie *Synechocystis* (hoofdstuk 3). Er is veel interesse in deze cyanobacteriën, omdat ze de potentie hebben om gebruikt te worden als "cellulaire fabriek" voor het omzetten van zonlicht en koolstofdioxide in diverse industriële producten, zoals biobrandstoffen.

De simulatie van deze genoom-schaal modellen resulteert vaak in grote datasets. Het interpreteren van deze grote datasets is een uitdaging. Om van biologische waarde te zijn, is het belangrijk om deze grote datasets te kunnen interpreteren in de vorm van biologische routes; deze routes kun je vergelijken met snelwegen. Routekaarten en navigatiesystemen worden veel gebruikt om van A naar B te komen. Om het interpreteren van deze datasets te vereenvoudigen hebben we een handgemaakte metabole routekaart ontwikkeld (hoofdstuk 3). Deze metabole routekaart is zowel door de computer te lezen en te bewerken als door wetenschappers te interpreteren. Deze metabole routekaart kan gebruikt worden voor bijvoorbeeld het automatisch visualiseren van (grote) datasets die gegenereerd worden met stoichiometrische simulaties.

Genoom-schaal stoichiometrische modellen worden vaak gebruikt om de optimale biomassa opbrengst uit voedingsstoffen uit te rekenen. Misschien verrassend, maar er zijn doorgaans meerdere biologische routes (en combinaties daarvan) die de biomassa opbrengst optimaliseren; dit geeft de optimale oplossingsruimte. In hoofdstuk 4 hebben we de eigenschappen van deze oplossingsruimte onderzocht. We hebben als eerste de complete oplossingsruimte op een unieke manier omschreven met een specifieke collectie van biologische routes. Deze collectie van biologische routes kunnen we efficiënt uitrekenen met de TimoTimo (de naam is afgeleid van de navigatiesystemenfabrikant TomTom). De TimoTimo gebruikt een verdeel en heers strategie (hoofdstuk 5) om deze specifieke collectie van biologische routes in het optimum efficiënt uit te rekenen.

Stoichiometrische simulaties zijn, zoals eerder gezegd, deterministisch. Ze negeren dat biologische systemen van nature stochastisch zijn. Indien de stochastische fluctuaties verwaarloosbaar zijn, kunnen we deterministische simulaties gebruiken. Dit is niet altijd het geval. Voor complex biologische systemen spelen stochastische fluctuaties met name een rol wanneer specifieke moleculen (bijvoorbeeld mRNA of transcriptiefactoren) aanwezig zijn in lage aantallen. Er bestaan wiskundige methodes die kunnen omgaan met stochastische fluctuaties. Deze methodes worden stochastische simulatie algoritmes genoemd.

Over de toepassing en het ontwikkelen van deze algoritmes voor het onderzoeken van fluctuaties in biochemische modellen gaat het tweede deel van mijn proefschrift. Dit deel begint in hoofdstuk 6 waar we een overzicht hebben gegeven van de huidige stand van zaken over stochastische simulatie algoritmes. Met een aantal illustratieve voorbeelden hebben we ook de voordelen van stochastische simulaties laten zien ten opzichte van deterministische simulaties. Stochastische simulatie algoritmes zijn bijvoorbeeld uitermate geschikt om de parameters van een (kinetisch) model te schatten.

Voor het onderzoeken van fluctuaties in biochemische modellen hebben we StochPy ontwikkeld (hoofdstuk 7). StochPy is een uitgebreid en gebruiksvriendelijk softwarepakket voor stochastische simulaties. De kracht van StochPy hebben we in hoofdstuk 7 geïllustreerd door middel van een aantal voorbeelden. De vaak niet te verwaarlozen stochastische contributies van celgroei en celdeling worden niet meegenomen in de tot nu toe ontwikkelde stochastische simulatie algoritmes. Wij hebben in hoofdstuk 8 een generiek stochastisch simulatie algoritme ontwikkeld dat rekening houdt met zowel de stochastische contributies van netto molecuul synthese als de stochastische contributies van celgroei en celdeling. Dit nieuwe algoritme is toegevoegd aan StochPy en de voorspellingen komen exact overeen met theorie en goed overeen met time-lapse microscoop data van groeiende micro-organismen.

In een afsluitende discussie (hoofdstuk 9) heb ik kritisch gekeken naar het gebruik en de toekomst van de vier hoofdrolspelers (model, simulatie, algoritme en software) in systeembiologie. Ik verwacht dat ze een onmisbaar onderdeel gaan worden van onderzoek naar complex biologische systemen. De discussie eindigt met mijn filosofie over de eisen voor nuttige software in de wetenschap.

Publications

1. **Maarleveld T.R.**, A. Schwabe, M. Kempe, J. van Heerden, N. Nordholt, M. Moinat, Bruggeman F.J. (2015) *Cell Growth and Division in Stochastic Simulation Algorithms Match Theory and Time-lapse Microscopy Data*, in preparation;
2. A.C. Reimers, **T.R. Maarleveld**, F.J. Bruggeman, B.G. Olivier, L. Stougie (2015) *A unified theory on flux modules*, in preparation;
3. **T.R. Maarleveld**, M.T. Wortel, B.G. Olivier, B. Teusink, F.J. Bruggeman (2015) *Interplay between constraints, objectives, and optimality for genome-scale stoichiometric models of metabolism*, PLoS Comput. Biol. 11(4);
4. L.C. Anink-Groenen, **T.R. Maarleveld**, P.J. Verschure, F.J. Bruggeman (2014) *Mechanistic stochastic model of histone modification pattern formation*, Epigenetics & Chromatin 7 (1), 30;
5. K. Wei, M. Moinat, **T.R. Maarleveld**, F.J. Bruggeman (2014) *Stochastic simulation of prokaryotic two-component signalling indicates stochasticity-induced active-state locking and growth-rate dependent bistability*, Molecular Biosystems, 10 (9), 2338-2346;
6. **T.R Maarleveld**, J. Boele, F.J. Bruggeman, B. Teusink (2014) *A Data Integration and Visualization Resource for the Metabolic Network of Synechocystis sp. PCC 6803.* Plant Physiology (164);
7. **T.R. Maarleveld**, B.G. Olivier, and F.J. Bruggeman, (2013). *StochPy: A Comprehensive, User-Friendly Tool for Simulating Stochastic Biological Processes.* PloS ONE 8(11);
8. A. Schwabe, **T.R Maarleveld**, and F.J. Bruggeman, (2013). *Exploration of the spontaneous fluctuating activity of single enzyme molecules.* FEBS letters, 587(17), 2744-2752; and
9. **T.R. Maarleveld**, R. Khandelwal, B.G. Olivier,B. Teusink, and F.J. Bruggeman. (2013). *Basic concepts and principles of stoichiometric modeling of metabolic networks.* Biotechnology journal, 8(9), 997-1008.

Curriculum Vitae

Timo Rian Maarleveld

16 January 1988, Amsterdam

Education

Ph.D. Computational Systems Biology 2011 - 2015
Centrum Wiskunde & Informatica (CWI), Amsterdam, Netherlands
Vrije Universiteit Amsterdam, Netherlands

M.Sc. Bioinformatics *(cum laude)* 2009 - 2011
Vrije Universiteit Amsterdam, Faculty of Exact Sciences, Netherlands

- GPA 9.1 (10) [132 ECTS; 120 ECTS required]

B.Sc. Medical Natural Sciences 2006 - 2009
Vrije Universiteit Amsterdam, Faculty of Exact Sciences, Netherlands

- GPA: 8.0 (10) [186 ECTS; 180 ECTS required]

Pre-University Education 2000 - 2006
Atheneum College Hageveld, Heemstede, Netherlands

- Double specialization degree: *"Nature & Technology", "Nature & Health"*
- Additional course: Economics

Index

τ-leap method, 96
Synechocystis, 35, 83

Active-state locking, 105
Algorithm, 6

Baby cells, 140
Big O notation, 96
Biology, 7

Chemical master equation, 90
Chemical moiety conversation, 16
Cyanobacterium, 9

Direct method, 94

Elementary flux modes, 25, 58
Extant cells, 140
Extreme pathways, 27

F-modules, 77
FAME, 49
Flux balance analysis, 21, 49
Flux modes, 18
Flux variability analysis, 22, 49

Genome-scale stoichiometric models, 15, 34, 48, 73

Interdivision time, 137

Kernel matrix, 19

Leap condition, 97
Lineality, 28, 57, 76
Linear programming, 21, 56

Markov process, 90
Metabolism, 15, 33, 53
Michaelis-Menten kinetics, 118
Micro-organism, 8
Minkowski sum, 28, 71
Model, 3
Monte Carlo method, 95
Mother cells, 140

Open source software, 171
Optimal solution space, 21, 28, 56, 83

Photorespiration, 39
Poisson process, 92
Propensity function, 90, 137
Python, 11, 171

Ray, 28, 57, 76
Reduced cost, 24

Scalable Vector Graphics, 37
Shadow price, 24
Simulation, 5
Software, 6
Stochastic simulation algorithm, 91, 136
StochPy, 110
Stoichiometric matrix, 16
Systems biology, 8

TimoTimo, 76

Vertex, 28, 57, 76
VoNDA, 35

Waiting times, 113, 130