©Copyright 2019
J Kyle Medley

المنارة للاستشارات
www.manaraa.com

Towards a Scalable, Future–Proof Platform for Dynamical Modeling in Biology

J Kyle Medley

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Herbert M Sauro, Chair

Joseph Hellerstein

Gianluca Interlandi

Program Authorized to Offer Degree:
Bioengineering

المنارة للاستشارات

www.manaraa.com

University of Washington

Abstract

Towards a Scalable, Future–Proof Platform for Dynamical Modeling in Biology

J Kyle Medley

Chair of the Supervisory Committee:
Herbert M Sauro
Bioengineering

Mathematical models are used widely throughout the sciences, and often influence not only research, but our daily lives. For example, weather prediction is made possible by numerical weather models that have advanced steadily since computers became widely available in the 1970's. The improvements in weather models have been so consistent that some researchers have coined a "Moore's law" for weather models. In its original formulation, Moore's law states that the number of transistors in an integrated circuit such as a CPU approximately doubles every year. The weather forecasting equivalent states that the accuracy of numerical weather models improves by ten percent every ten years [291, 181].

Systems biology models do not advance to a steady drumbeat as weather models do. It can hardly be claimed that a ten year period yields a ten percent universal improvement in systems biology. In fact, current systems biology models are in many ways more primative than weather models [1]. Systems biology models (especially detailed, mechanistically–accurate models) are underutilized in synthetic biology and are almost completely absent from the clinic. This is unfortunate, because systems biology models have the potential to aid in drug discovery [57], cancer treatment [28, 159, 117], disease biology [227, 307], and

[1] This can be seen, for example, by looking at the usefulness of multiscale "subgrid" methods in weather modeling. By comparison, systems biology models struggle to recapitulate biology at a single scale — to use an analogy, systems biology is stuck at the "grid" phase of modeling. A considerable amount of work is required to catch up to the equivalent "subgrid" milestone.

المنارة للاستشارات
www.manaraa.com

production of biofuels [56, 170, 46].

Why are systems biology models so underutilized, despite considerable advances in computing power and experimental data? There are at least three major factors: difficulty in model reuse, lack of scalability, and lack of technological advances for simulation. Specifically, these three factors are due to the following problems:

1. Perhaps more so than any other field, models in systems biology need to be reusable and reproducible [2]. Due to the complexity of biology, no single research team can specialize in every subsystem of the cell. Therefore, models of cellular subsystems must be developed, validated, and analyzed by different research teams, and combined into a single, larger model of the cell. This can only happen if researchers use uniform standards to store their work, and provide a means for others to reuse and incorporate their models.

2. As the size of a model increases, so do the resources required to simulate it. This is not simply an issue of convenience, since fitting a model requires simulating it many times. The strain on computational resources has caused many groups to use simpler "constraint–based" modeling approaches. However, this approach trades detail for performance. Continued advancement of systems biology requires the development of mechanistically accurate kinetic models, which is currently hampered by scalability constraints.

3. Finally, despite decades of innovation in computer simulation, kinetic models are still commonly simulated using 40–year old solvers such as LSODA [231, 124]. It is natural to ask whether advancements in computer technology could be used to provide a better approach to simulating models. Indeed, this thesis considers state–of–the–art specialized silicon hardware specifically designed to simulate kinetic models.

[2] Chapter 2 clarifies the difference between the terms "reusable" and "reproducible".

المنارة للاستشارات
www.manaraa.com

This thesis seeks to address these factors through technological innovations which enable the construction of larger, more accurate, and more robust models. This is accomplished through providing better solutions for encoding and reusing models, providing a scalable solution for optimizing large, challenging kinetic models, and providing a way to simulate models on special–purpose hardware. Taken together, these foundational advances in modeling technology provide a pathway toward building larger, more complex, and more accurate models.

المنارة للاستشارات
www.manaraa.com

TABLE OF CONTENTS

Page

List of Figures iii

Glossary v

Chapter 1: Introduction 1

1.1 A Survey of Modeling Formalisms 11

1.2 Applications: Models in Medicine / Systems Pharmacology 28

1.3 Applications: Models in Synthetic Biology 28

1.4 Parameterizing Models 29

1.5 Summary of Chapters and Scientific Contributions 36

Chapter 2: A Platform for Reproducible, Dynamical Modeling in Systems Biology 38

2.1 Survey of Standards 39

2.2 Background 46

2.3 Design and Implementation of Tellurium 49

2.4 Using Tellurium to Accomplish Advanced Tasks with SED–ML 52

2.5 Enabling Web–Based SBML Tools 65

2.6 Web Compilation of libSBML 66

2.7 Discussion 69

2.8 Review of Contributions 74

Chapter 3: Parameter Estimation via the Island Method to Enable Scalable Modeling 75

3.1 Parallelization Approaches 77

3.2 Survey of Population–based Algorithms 80

3.3 Description of Benchmarks 85

3.4 Details of Each Benchmark 89

3.5 Scaling Behavior 90

المنارة للاستشارات

www.manaraa.com

3.6 Comparison with Distributed Algorithms 91
3.7 Hardware Specification 92
3.8 Conclusions 93

Chapter 4: Accelerated Simulations via Special–Purpose Hardware 106
4.1 Introduction 106
4.2 The Divergence Problem 108
4.3 Chip Layout 109
4.4 Methods 111
4.5 Shift Registers & Block Parameters 116
4.6 SRAM & Network Motifs 116
4.7 Lumped Kinetics 118
4.8 Compiling Gene Regulatory Kinetics 127
4.9 Higher–Order Compilation 132

Chapter 5: Conclusion 136
5.1 Contributions to Standards Integration 136
5.2 Contributions to Web–based Support for Standards 137
5.3 Contributions to Scalable Kinetic Modeling 137
5.4 Contributions to Simulation Technology 138

Bibliography 140

Appendix A: List of Software Projects 170

المنارة للاستشارات
www.manaraa.com

LIST OF FIGURES

Figure Number Page

1.1 Glycolysis — An example of a metabolic network. 3

1.2 Diagram of the EGFR pathway. 7

1.3 Examples of natural and synthetic gene regulatory networks. 9

1.4 Commonly occurring feed–forward loop motifs. 10

1.5 Conserved moieties example. 17

1.6 Multiple simulation traces of a stochastic model. 19

1.7 Rule–based model simulation. 26

1.8 Bootstrapping pseudocode. 34

2.1 Using SBO terms in Tellurium. 53

2.2 Enhanced readability of standards using Tellurium's shorthand representation. 55

2.3 Advanced features of SBML/SED–ML COMBINE archives: multiple parameter sets and plots 57

2.4 Encoding model variants using COMBINE archives 60

2.5 Linear and logarithmic PhraSEDML plotting. 62

2.6 Testing the shift in regulatory mechanism of mitotic oscillations. 64

2.7 The libsbmljs compilation process and demo. 70

3.1 A visual depiction of the island method. 76

3.2 Island method pseudocode. 77

3.3 Problems used in the elimination benchmark. 78

3.4 Convergence curves for problems B1 (A) and B3 (B) from the BioPreDyn–Bench suite for various numbers of islands. 94

3.5 Convergence curves for problem B2 from the BioPreDyn suite. 95

3.6 Convergence curves for problem B4 from the BioPreDyn suite. 96

3.7 B1 problem best solution. 97

3.8 B1 problem best solution. 98

3.9 B2 problem best solution. 99

المنارة للاستشارات

www.manaraa.com

3.10 B2 problem normal probability plots. 100
3.11 B3 problem best solution. 101
3.12 B3 problem normal probability plots. 102
3.13 B4 problem best solution. 103
3.14 B4 problem normal probability plots. 104
3.15 Quality–of–fit benefit for a fixed computation time. 105

4.1 Demonstration of the divergence problem. 110
4.2 Layout of the cytomorphic chip. 111
4.3 Block diagram of a reaction unit. 112
4.4 A flow diagram for the cytomorphic compiler. 114
4.5 Input and output ports for a single block in the cytomorphic chip. 117
4.6 Testing a three–step feed–forward network with the compiler. 119
4.7 Testing a “fan–out” reaction with the compiler. 120
4.8 Testing a dissociation reaction with the compiler. 121
4.9 Testing a “fan–in” reaction with the compiler. 122
4.10 Negative feedback around the circuit for A. 124
4.11 Different margin values and their respective simulations. 125
4.12 Pseudocode for the Martin matching algorithm. 128
4.13 Wiring diagram of the repressilator model. 130
4.14 Comparison of repressilator model simulations. 131
4.15 Compilation of a rule–based MAP kinase model [176, 64]. 133

المنارة للاستشارات
www.manaraa.com

GLOSSARY

CHAPTER 1:

SYSTEMS BIOLOGY: The branch of biology that uses mathematical modeling and quantitative data to describe the behavior of large, complex biological systems.

SYNTHETIC BIOLOGY: A field that focuses on combining biology and engineering to produce novel phenotypes for use in industrial chemical production, medicine, and other applications. Synthetic biology heavily employs recent advances in molecular cloning, genome annotation, and occasionally modeling to achieve the desired cellular phenotype.

ODE: Ordinary differential equation. An equation or set of equations describing the rate of change of a set of quantities as a function of the current state of the system. Also used to refer to biochemical models based on such a formalism. In this work, differential equations are always first–order.

PDE: Partial differential equation. In systems biology, PDEs are usually used to represent spatial dependence of processes (such as processes occurring only at the cellular membrane, in the nucleus, or in the vicinity of various organelles or structures). However, such spatial dependence is rare due to the difficulty of parameterizing and experimentally validating the precise spatial dependency of a process.

STOCHASTIC MODEL: A systems biology model that uses Gillespie's algorithm [108] to predict the timing of individual reaction events that involve bond formation / breaking or binding at the single molecule level, such as elongation of an mRNA transcript or a single enzyme turnover.

CONSTRAINT–BASED MODEL: A model that specifies reaction fluxes in terms of constraints. Solving the set of constraints yields a set of fluxes for the network. Unlike an ODE or stochastic model, a constraint–based model does not change in time. It is based on the tacit assumption that the fluxes of all reactions do not change significantly over time (although variants do exist that explicitly address time dependence [259]).

للاستشارات المنارة
www.manaraa.com

BIOMASS REACTION: A reaction in a constraint–based model that includes all essential metabolites with coefficients that are assumed to mirror the biomass composition of the physical cell.

CHEMICAL SPECIES: A chemical compound in a biological model. Examples include metabolites, mRNA molecules of different genes, and proteins in various states (such as different phosphoforms, or different states of protein complexes).

BIOCHEMICAL REACTION NETWORK: (or simply “network”). The set M of all chemical species in a biological system, together with the set of all reactions N, forms a directed graph with M vertices and N edges. If a particular species is a reactant in a reaction, then the reaction represents an outgoing edge. If the species is a product, the reaction is an incoming edge.

SENSITIVITY: The rate of change of an output of a model with respect to an input, which is usually a parameter or species abundance.

CHAPTER 2:

XML: Extensible markup Language. A hierarchically organized language that evolved from HTML. Used on the World Wide Web to exchange data in a structured format. Also used by many of the standard formats described in this thesis.

COMMUNITY STANDARD: A community standard is developed by the community that uses it (as opposed to a recognized standard body). In other words, it is a self–imposed standard voluntarily agreed to by members of the community due to the benefits it provides.

SBML: The Systems Biology Markup Language. An XML–based format that specifies an ODE or stochastic systems biology model in terms of its chemical species, reactions, parameters (e.g. on/off constants or Michaelis constants), and possible discrete events (such as adding a bolus of ligand to a system).

LANGUAGE BINDING: A layer of wrapper code that allows a C++ library to be used by many scripting languages. It translates function calls in the target scripting language (such as Python) to C++ function calls.

LIBSBML: A C++ library for reading and writing SBML. libBSML has bindings for many languages such as Python, Java, and others.

www.manaraa.com

JSBML: A native Java library for reading and writing SBML. It is independent of libSBML (i.e. not a language binding).

CELLML: Another XML standard for describing systems biology models. Whereas SBML is primarily used for molecular–scale mechanistic models, CellML is used to encode physiological models (such as action potential models of neurons like the Hodgkin-Huxley model [126]). Unlike SBML, CellML encodes ODEs for quantities instead of processes.

URI: Uniform Resource Identifier. An identifier that unambiguously points to a specific resource, usually under a given domain name in the World Wide Web. For example, the URI `http://identifiers.org/biomodels.db/BIOMD0000000012` unambiguously identifies the BioModels repressilator [88] model. This URI can be entered into a web browser to retrieve information about the resource, but this is not a general requirement for URIs.

ONTOLOGY: A database (usually online) that describes relationships between entities identified by URIs. The purpose of ontologies is to allow systematic interpretation classification of the biological processes and entities used in models. For example, an SBML model may contain a reference to adenosine triphosphate (ATP). In this case, the SBML model should use the URI `http://identifiers.org/CHEBI:15422`, which unambiguously points to the term in the ChEBI ontology that represents the substance “ATP”.

SEMANTIC ANNOTATION: A way of linking elements of a model (SBML or CellML) to resources in an ontology.

SED–ML: An XML standard that describes how to simulate SBML and CellML models. This standard is needed in order to reproduce previously published results. For example, to reproduce a simulation, it is necessary to know the values of all model parameters, the simulation options, and also what algorithm was used to simulate the model. SED–ML makes it possible to reproduce previously published results in a fully automated fashion.

COMBINE ARCHIVE: The proliferation of XML standards such as SBML, CellML, and SED–ML has created a need to package related files together. SED–ML is particularly problematic because it references other files using relative paths. A COMBINE archive is a single file that contains other files, and is used to distribute related files together in order to avoid problems caused by SED–ML references, mismatches between copies of files, and generally improve usability of projects using multiple related standards.

المنارة للاستشارات
www.manaraa.com

CHAPTER 3:

OBJECTIVE FUNCTION: The function that is to be minimized in an optimization problem. In this work, it is always a multivariate real–valued function of real variables. A typical choice for the objective function is the relative root–mean–square deviation (RMSD) between the predicted values of an ODE model and experimentally measured values.

RELATIVE DEVIATION: (or relative RMSD). The RMSD value of a quantity scaled by the quantity's average value. For example, if a model has an RMSD of 10 μM for pyruvate and 50 μM ATP, whereas the average intracellular concentrations of pyruvate and ATP are 100 μM and 1 mM respectively, then the respective relative RMSD values for pyruvate and ATP would be $10/100 = 0.1$ and $0.05/1 = 0.05$.

DECISION VECTOR: The set of arguments to the objective function. For example, if the objective function is a function of 12 real variables, the decision vector will have length 12. A solution to an optimization problem is a decision vector that yields a minimum value when evaluated with the objective function.

ISLAND METHOD: Also known as the "island model." A means of parallelizing population–based optimization algorithms by running different algorithms on different nodes in a computing cluster, and occasionally allowing these algorithms to exchange candidate solutions.

CHAPTER 4:

INTEGRATED CIRCUIT: A silicon chip designed and fabricated to carry out a specific task.

CYTOMORPHIC CHIP: A special–purpose integrated circuit that uses analog computation based on current values to simulate a chemical reaction network.

REACTION BLOCK: Each cytomorphic chip contains 20 reaction blocks, which each simulate a single bimolecular reaction with mass–action kinetics. Multiple reaction blocks are required to simulate larger reactions, as described in Chapter 4.

CYTOMORPHIC COMPILER: A compiler that translates biochemical models (described e.g. by an SBML model) into a hardware configuration that can be used to simulate models on the cytomorphic chip or validate the hardware model via ODE integration.

RATE LAW REDUCTION: The process by which the cytomorphic compiler breaks down rate laws of lumped processes, such as Michaelis–Menten kinetics, into constituent mechanistic components, such as separate binding and catalysis steps.

المنارة للاستشارات
www.manaraa.com

WIRING DIAGRAM: As part of the compilation process, the cytomorphic compiler generates a wiring configuration describing the connectivity of input and output ports for each reaction block. This wiring can be visualized as a diagram. Example configurations are shown in Chapter 4.

المنارة للاستشارات

www.manaraa.com

ACKNOWLEDGMENTS

I am deeply grateful to my advisor, Herbert Sauro, for the many opportunities he has given me through international conferences, collaborations, internships, and workshops. All of these opportunities allowed me to connect with a large community of researchers and gain a wealth of knowledge of cutting–edge research in systems biology, none of which would have been possible without Herbert's help.

I would also like to thank Joseph Hellerstein for taking time to meet with me on a monthly basis to give me research advice. Proficiency at research and articulating one's findings are quite difficult skills to develop, and I am grateful to Joe for helping me to hone those abilities through in–depth research discussions and manuscript editing.

I would like to express my gratitude to the rest of my thesis committee: Gianluca Interlandi, who was always supportive and met with me several times to give me advice on research and navigating the labyrinth of graduate school; James Carothers, who gave me the encouragement to submit my Tellurium manuscript to PLoS Computational Biology (which was a major success); and Eric Shea–Brown, who opened my eyes to techniques unknown within my field such as High Dimensional Model Representation [169].

Academic writing is a highly coveted skill which is difficult to nurture. During the preparation of my first manuscript, I was very fortunate to have the help of Jonathan Karr, who also co–authored the manuscript and provided a great deal of wisdom and expertise. I do not think the manuscript could have been accepted without Jonathan, and I am also very grateful to him for helping me to hone my scientific writing.

للاستشارات المنارة
www.manaraa.com

During my graduate studies, I had the privilege of working with Steve Wiley at Pacific Northwest National Lab (PNNL) on a signaling pathway modeling project. I am very grateful to Steve for sharing his knowledge with me and helping me to gain a better understanding of this important field.

I would also like to thank my academic counselor, Erin Kirschner, for helping me many times throughout the years

Finally, I would like to thank my graduate student companions, Kiri Choi and Bryan Bartley, both for their help in tackling problems and for being travel partners in our many conference trips, many of which turned out to be international adventures.

المنارة للاستشارات
www.manaraa.com

DEDICATION

To my parents, to whom I am deeply indebted. I am at a loss for words to express my gratitude for their love and support.

المنارة للاستشارات

www.manaraa.com

Chapter 1
INTRODUCTION

Mathematical models are used ubiquitously throughout biology. At the atomistic level, forces due to electrostatic charges and Van der Waals forces (often modeled using the Lennard–Jones potential) can be used to predict the conformation of biomolecules, drive molecular dynamics simulations [97], and simulate small molecule docking for drug screening and design [65]. Still more detailed formalisms, such as density functional theory, incorporate quantum–mechanical effects into molecular models [319].

At the next level of granularity are network models. Interactions between biomolecules, such as binding, catalysis, and degradation, give rise to a set of interactions. The interactions form the edges of the network and the components themselves form the nodes. This network formalism can be applied to many different types of cellular systems. Networks process nutrients, generate energy and carbon building blocks, transmit information, and allow the cell to adapt to different environmental conditions. Cellular networks can be broadly divided into three categories: metabolic networks, signaling / protein interaction networks, and gene regulatory networks. Conveniently, all three of these types of networks can be modeled using the same mathematical formalism, which allows network modeling software to be used for a wide range of scientific questions.

At still a higher level of granularity are models of multicellular structures and tissues. Multi–cell models have been used to study processes such as angiogenesis and tumor growth [293] and even to predict the behavior of entire organs such as the heart [212].

This chapter aims to serve as a brief, high–level introduction to cellular networks, systems biology, and various methods of modeling. It provides the contextual background for subsequent chapters, which build on this material to provide new insights and advances. It

المنارة للاستشارات

www.manaraa.com

introduces biological network modeling through a number of examples, and examines the significance, potential impact, and drawbacks of current methods for modeling.

1.0.1 Metabolic Networks

Metabolic networks are one of the most well–studied examples of biochemical networks. Their investigation was spurred by the field of biochemistry long before the advent of computers. The glycolysis pathway is a well–known and well–studied example. It processes glucose in order to produce two net molecules of ATP, two molecules of the cofactor NADH, and a host of carbon skeleton molecules required by the cell. Figure 1.1 shows the key steps in glycolysis based on a previously published model [136]. Negative feedback plays a major role in the glycolysis pathway. This is due to the fact that many glycolytic enzymes can be allosterically regulated, meaning that effector molecules can bind to these enzymes outside of the active site to alter the activity of the enzyme. Six regulatory signals converge on fructose 1,6–bisphosphate (of which one is shown in the figure). Fructose 6–phosphate represents a pool of intracellular sugar molecules, and phosphofructokinase commits these molecules to the remainder of glycolysis via the transfer of ATP [1]. This step is one of four key steps which largely control glycolytic flux in mammalian cells [295].

Partly as a result of negative feedback, metabolic networks have a rapid response time (microseconds to seconds [261]). Metabolic networks need to respond rapidly to changes in supply and demand of nutrients, an important property for the organism's survival.

Glycolysis is one of the most well–studied biochemical pathways. Yet, modeling even this archetypical pathway poses significant challenges. In one study, Teusink *et al.* painstakingly determined all enzymatic parameters of the pathway experimentally [298] and used these values to parameterize a kinetic model. This model failed to reach a steady state unless branches were included to other pathways (trehalose, glycogen, glycerol and succinate), and

[1] the molecules may be converted back to fructose 6–phosphate via the reverse reaction catalyzed by fructose bisphosphatase, but this is a futile cycle that needlessly consumes ATP and is thus assumed to be downregulated

المنارة للاستشارات
www.manaraa.com

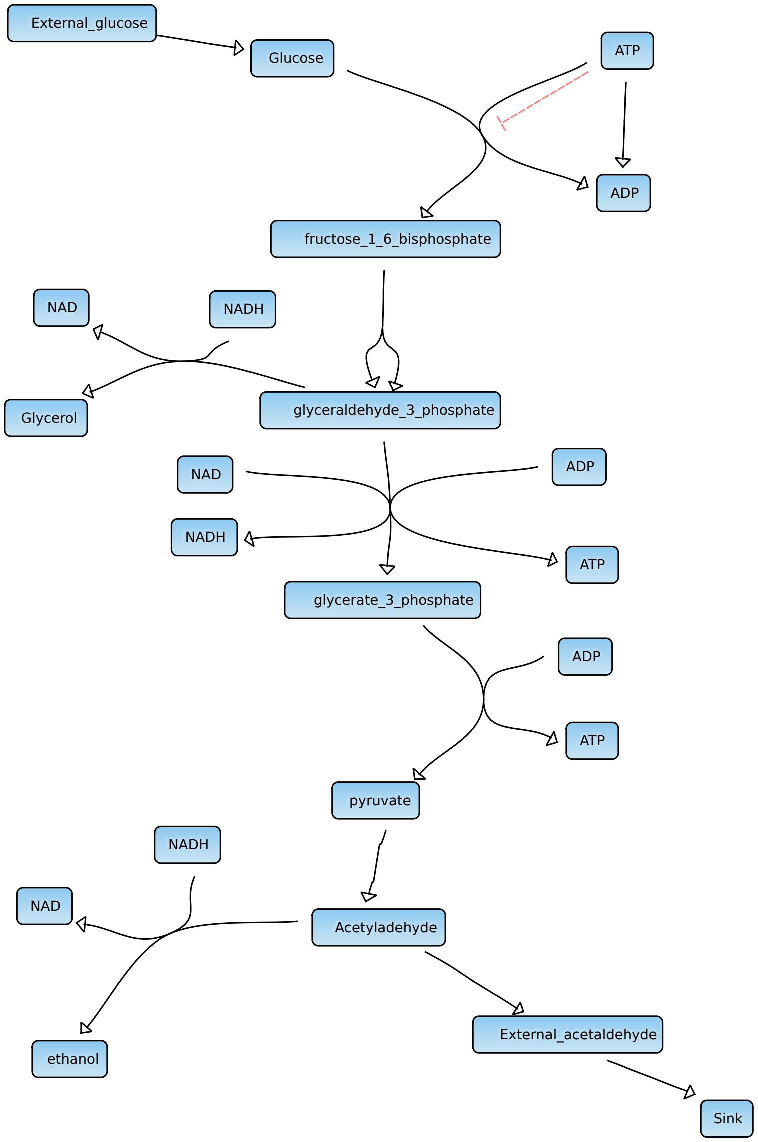


Figure 1.1: A canonical example of a metabolic pathway in the form of a condensed version of glycolysis (the conversion of fructose 1,6–bisphosphate to pyruvate is a 6–enzyme process — not all steps are shown). Glycolysis consumes six–carbon sugars and produces energy in the form of ATP and the reducing agent NADH. The diagram shown here depicts negative regulation (in red) of the fructose 1,6–bisphosphate committal step via ATP. Thus, an excess of ATP will tend to reduce glycolytic flux. Glyclysis is also negatively regulated by the downstream metabolite phosphoenolpyruvate (not shown) and possibly positively regulated by citrate. This diagram was generated from an SBML model [136], which also considers fermentation (hence the presence of reactions downstream from pyruvate in the diagram).

المنارة للاستشارات

www.manaraa.com

even with these branches, significant deviations were observed compared to *in vivo* results. Teusink *et al.* posited that these deviations were possibly due to different regulation of the enzymes under *in vivo* conditions. Thus, even the most well–studied metabolic pathways are difficult to model mathematically.

Nevertheless, improved understanding of metabolic pathways like glycolysis is critical in industry and medicine. The fermentation of sugar to ethanol represents one of the first instances of human exploitation of a biochemical pathway [306]. Today, this process is widely used by the food industry for the production of alcohol and cheese. Current efforts are underway to use glycolysis for biofuel production [56, 170, 46]. In medicine, the upregulation of glycolysis to produce lactate from glucose is one of the hallmarks of cancer (the Warburg effect [316]), an important phenomenon that could be used to detect and treat cancer [256, 224, 185, 34, 295, 331]. A set of key enzymes in upper gylcolysis is consistently upregulated in cancer by the Ras oncogene [295], whereas other studies based on computational modeling have pointed to a different set of key mediators of flux in the pathway [273]. This suggests that a holistic approach based on modeling and combined data from different conditions is needed to fully understand the range of responses to different nutrient conditions, even for a single pathway. Indeed, it has been suggested that glycolysis itself can act as a flux sensor and drive various cellular functions based on nutrient availability and intracellular conditions [158, 160, 130, 306].

Glucose metabolism may play an important role in the activation of T lymphocytes [223]. Whereas naïve and memory T cells circulate in the bloodstream in a catabolic resting state, activated T cells rapidly transition to a state of growth and proliferation. This requires a significant shift in metabolic programming and nutrient allocation. It is believed that newly activated T cells prefer glycolysis over oxidative phosphorylation to drive the accumulation of biomass and prepare the cell to enter infected areas or tumors, where nutrients may be depleted [223]. Altered glycolysis is also a major factor in other disorders such as diabetes [270, 116], where it may play a role in rendering muscle tissue insulin resistant. Finally, glycolysis is the main ATP production pathway in the malaria parasite *Plasmodium falciparum*,

المنارة للاستشارات
www.manaraa.com

and could potentially be used to treat malaria infections [210].

Metabolic networks, in addition to providing the cell with nutrients and energy, also play an important role in disease, potential treatments for cancer and infection, and the production of biofuels and other important industrial chemicals in synthetic biology. The advent of enzyme kinetics more than a century ago [200] has allowed for a great deal of mathematical modeling of metabolic pathways. Yet, despite this, much work remains in order to fully understand the regulation of these pathways.

1.0.2 Signaling / Protein Interaction Networks

Signaling networks help the cell to respond to external stimuli. Extracellular signals are frequently transduced based on interactions between proteins, such as protein binding, phosphorylation, or degradation, and for this reason signaling and protein interactions are grouped together here. Whereas the other categories of cellular networks (metabolic and gene regulatory networks) are well defined, signaling and protein interaction networks comprise the remainder of networks that do not fall into these neatly defined categories. As a consequence, they are very diverse in their mechanism of action, involving enzymatic catalysis, protein binding, and sometimes ion flow across membranes.

One major signaling pathway in eukaryotes is the G protein–coupled receptor (GPCR) pathway. This pathway is notable because its modular nature allows it to connect a wide variety of stimuli to different cellular responses. GPCRs are known that respond to light, sugars, peptides, and lipids [220]. A GPCR is a membrane–bound receptor that, upon stimulation via its cognate ligand or absorption of light energy, releases an associated G protein via replacing the G protein's bound GDP nucleotide with GTP. The G protein may then activate secondary messenger molecules. GPCR pathways are highly important in medicine and are the target of approximately a third of all currently used drugs [96], and have been the subject of numerous modeling studies [123].

Another important signaling pathway is the epidermal growth factor receptor (EGFR) pathway. The EGF receptor consists of a family of four closly related membrane proteins

المنارة للاستشارات
www.manaraa.com

that respond to their cognate autocrine signaling peptides (EGF and TGFα). The signal is transduced slightly differently by different members of the EGFR family, but it usually involves receptor dimerization, internalization (i.e. engulfment of active receptors by the cell to form endosomes), and phosphorylation of the cytoplasmic tyrosine residue on the receptor. The EGFR pathway is mutated in the majority of cancers, making it a target in cancer treatment as well as an important part of understanding tumor biology. Many attempts have been made to model the EGFR pathway, with the earliest models accounting for short–term kinetics using Western blotting for quantitative data to parameterize the model [150, 156]. More modern methods use mass spectrometry to obtain quantitative data with better accuracy and reproducibility [274].

Signaling networks are usually formed by sets of interacting proteins. Even in the case of ionic signaling such as calcium ion flow in muscles and neurons, membrane transporter proteins invariably play an important role. A major part of elucidating the behavior of signaling networks is therefore determining which proteins in the cell interact by binding or some other mechanism.

Binding interactions can be detected using a variety of approaches. One high–throughput method is the yeast two–hybrid assay [261], which is based on fusing the putative interacting proteins of interest to different domains of a transcription factor. If the proteins are binding partners, these transcription factor domains are brought close together, the transcription factor becomes functional, and transcription of the gene can be used as a readout of activity. However, the two–hybrid system is also known to have reliability problems and frequently causes false positives and negatives, as evidenced by the inconsistency of results [77].

An alternative to the two–hybrid assay is Förster resonance energy trasnfer (FRET), which relies on labeling the putative interacting proteins with fluorophores that have separate excitation and emission spectra. The donor fluorophore emission spectrum is selected to match the acceptor fluorophore absorption spectrum. When the two fluorophores are brought close together, the donor may transfer some of its fluoroescent energy to the acceptor, which then emits in turn, and this signal can be used to determine the proximity of the two proteins

المنارة للاستشارات
www.manaraa.com

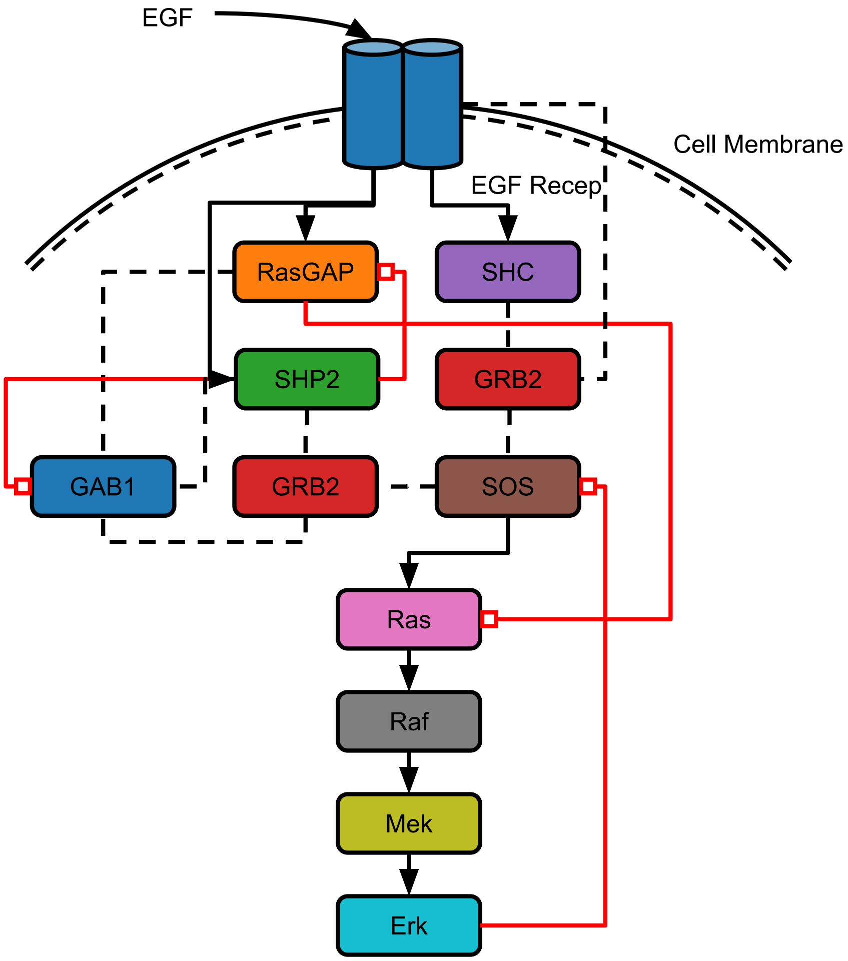


Figure 1.2: A diagram of the EGFR pathway based on the model by Borisov *et al.* [47]. Dashed lines are putative interactions, solid black lines are verified interactions, and red lines are inhibitory interactions. After binding to its cognate ligand, an EGF receptor is internalized and phosphorylates a typrosine residue on its own cytoplasmic tail. The cytoplasmic side of the receptor can then bind to the Src–Homology–2–containing SHC adapter protein, as well as other potential adapter protein complexes such as Grb2–Sos [44]. These adapter proteins then eventually activate Ras, a small GTPase, that in turn activates the mitogen activated protein kinase (MAPK) pathway, which regulates cellular processes related to growth and survival.

المنارة للاستشارات

www.manaraa.com

and hence whether an interaction occurs. Other techniques for detecting protein interactions include pulldown and coimmunoprecipitation assays [285].

1.0.3 Gene Regulatory Networks

Gene regulatory networks control the expression of different genes at the transcriptional and translation level. A classic example is the *lac* operon in *E. coli* and many other bacteria. *E. coli* has a constitutively expressed repressor protein (LacI) that binds to the *lac* operon and prevents its expression. When lactose is present in the cytoplasm, it is metabolized to allolactose, which binds to the LacI repressor and causes it to change conformation, preveting it from binding to DNA at the *lac* operon.

LacI along with two other repressors, TetR from a transposon encoding tetracycline resistance, and the cI repressor from λ phage, have been combined in a synthetic genetic oscillator known as the *repressilator* [88]. Modeling played an important part of this feat of synthetic biological engineering. As can be shown from the model, the system only oscillates under certain conditions. Hence, it is important to ensure these conditions can be met prior to building the circuit. The repressilator model is available in the BioModels repository [89].

"*What I cannot create, I do not understsand.*" Physicist Richard Feynman famously left this message on a blackboard shortly before his death in 1988. This observation certainly holds true for biology. In the same year as the repressilator (2000), another synthetic genetic circuit was also published in *Nature*: the bistable switch (also known as a "toggle switch") by Gardner *et al.* [103]. This switch is formed from two genetic repressors: LacI and TetR (other variants use cI). When LacI protein is at high levels, it prevents transcription of TetR and vice versa. The system can be flipped from one state to the other via induction of LacI and TetR using isopropyl–β–D–thiogalactopyranoside (IPTG) or anhydrotetracycline respectively. The *lac* operon, the repressilator, and the toggle switch are shown in Figure 1.3.

المنارة للاستشارات
www.manaraa.com

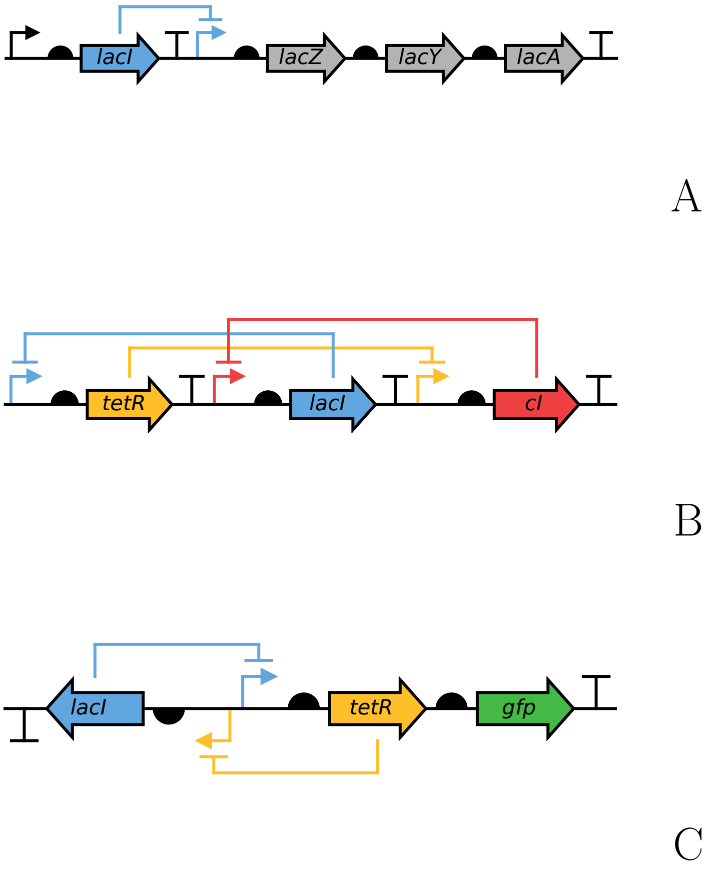


Figure 1.3: Examples of natural and synthetic gene regulatory networks. All diagrams use SBOL visual [241] symbols: promoter (arrow), ribosome binding site (RBS, half–circle), coding sequence (thick arrow), and terminator ("T"). The *lac* operon consists of a collection of genes for metabolizing lactose (A) along with the LacI repressor under a separate constitutive promoter. (B) The repressilator: A synthetic network of three transcriptional repressors that generates oscillations under certain conditions [88]. (C) A genetic toggle switch [103]. These images were generated with DNAplotlib [79]. (B) is based on an example included with [79].

المنارة للاستشارات

www.manaraa.com

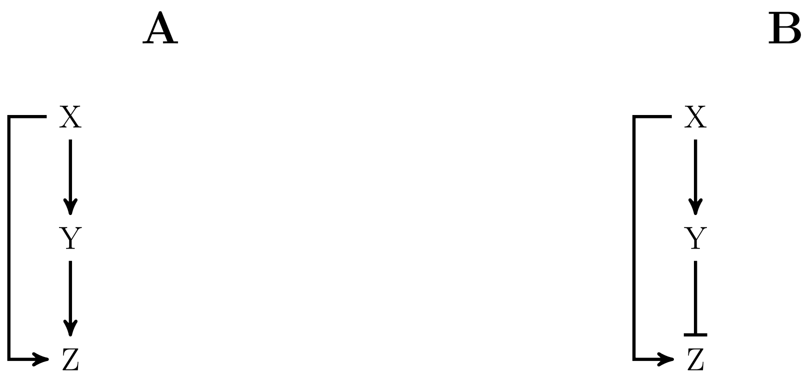


Figure 1.4: Coherent (A) and incoherent (B) feed–forward loops. Arrow heads are positive interactions and flat connectors are negative. Many other types of combinations of positive and negative connections are possible, but the specific configurations shown here are the most common in natural yeast and *E. coli* gene networks. See [261] for a more comprehensive list of different types of motifs.

1.0.4 Network Motifs & Design Principles

An interesting feature of many types of networks is the appearance of *motifs* — subgraphs with certain structures that occur much more frequently than would be expected given a random graph. Motifs were first introduced to systems biology in 2002 [202] but were studied in other fields under different nomenclature as early as 1979 [71, 287, 289, 288]. To enable a fair computation of the frequency of motifs, the random networks should be generated with a degree distribution that matches the input graph [261] or, more generally, sampled from the "universe" of graphs with row and column sums matching the input [288]. Although the role and significance of most motifs is still largely unknown, feed–forward loops have been suggested to serve as persistence detectors (i.e. they respond to sustained input but not short bursts of input), to speed up overall transcription response times in genetic networks [182], and to process pulsatile dynamics [329]. Figure 1.4 shows the two most common types of feed–forward loops in natural yeast and *E. coli* gene networks.

المنارة للاستشارات
www.manaraa.com

The usefulness of motifs as design principles is dubious due to the fact that ubiquity alone does not yield any conclusions regarding the essentiality of the function of motifs.

1.1 A Survey of Modeling Formalisms

Cells are ultimately chemical systems, so it is only natural to describe cells in terms of the chemical reactions that occur within them [2]. Perhaps the two most important reactions in living cells are transcription and translation. In prokaryotes, transcription can be modeled as the binding of sigma factor and RNA polymerase (RNAP) to a gene promoter, open–complex formation, and elongation of the mRNA transcript:

$$\begin{aligned} P + \sigma &\rightleftharpoons P_\sigma \text{ Binding of } \sigma \text{ factor} \\ P_\sigma + \text{RNAP} &\rightleftharpoons T_1 \text{ Binding of RNAP, transcript initiation} \\ T_N + \text{tRNA} &\longrightarrow T_{N+1} \text{ Transcript elongation} \end{aligned}$$

Depending on the level of detail of the model, these processes can be condensed into a single process per gene that represents simply “transcription” with no intermediate states.

Translation can be represented similarly at varying levels of detail (ribosome binding and elongation, or simply a lumped process without any intermediate states). There are many different methods for modeling biological processes. These methods mainly differ by the level of detail that they use to represent underlying processes. Table 1.1 shows a summary of the methods discussed in this thesis from least detailed to most detailed. It is possible to combine multiple different approaches at different levels of detail to create a multi–scale model. Ideally, such a model would be accurate at every level of detail, but in practice tradeoffs must be made in order to parameterize and efficiently simulate multi–scale models, necessitating case–by–case compromises. Hence, multi–scale models are not included in Table

[2] Reactions need not occur in a living system to exhibit complex behavior. An interesting counterexample is the Belousov–Zhabotinsky reaction, which produces complex, localized dynamics (see `https://www.youtube.com/watch?v=IBa4kgXI4Cg` for a demonstration).

المنارة للاستشارات
www.manaraa.com

1.1 because they may have different accuracies at different levels of detail, or they may be inaccurate at all levels due to an inaccuracy at one level propagating to other levels.

The choice of modeling formalism for a particular problem depends on the scientific questions one wishes to ask and the level of detail required to address them, the available data, and computational feasibility. If the goal is to understand knockout lethality, a constraint–based model may be sufficient, since these models are often able to predict which reactions are required for a pathway. However, for modeling transient processes such as calcium signaling or circadian rhythms encoded by genetic circuits, a kinetic model may be necessary.

1.1.1 Kinetic Models

Kinetic models have a long history in biology, with perhaps one of the first examples being Michaelis–Menten kinetics for enzyme catalysis, derived over a century ago [200]. At the mechanistic level, kinetic models are based on the idea of chemical potential acting as a driving force for chemical reactions, but they can also be used anywhere the law of mass action is applicable (for example, clearance of a virus from the body — see below). Kinetic models describe the rate of change of different quantities over time. This change can occur in bulk, as it does for most cases where the number of molecules is large (abundant cellular metabolites and proteins), but the change can also occur in discrete events (stochastic models, below). Kinetic models thus span a wide range of levels of granularity, from bulk reactions to individual molecules.

A Motivating Example

An important instance of kinetic modeling was in the development of a treatment regimen for human immunodeficiency virus (HIV). Following initial flu–like symptoms that result from infection by the virus, patients typically exhibit a very gradual depletion of T lymphocytes over many years, which initially led many researchers to suspect that the virus emerged very slowly and gradually, eventually reaching high enough numbers to overwhelm the immune system. However, a study incorporating a kinetic model by Alan Perelson shows that this was

المنارة للاستشارات
www.manaraa.com

	Formalism	
Static ↑	Logical / boolean models (can be dynamic)	Least detailed
	Constraint–based models	
	Hybrid constraint / kinetic models, cybernetic models [326]	
	Kinetic ODE models	
	rule–based ODE models	
	Stochastic kinetic / rule–based models	
	Spatial diffusion / PDE models	
Dynamic ↓	Molecular dynamics simulations	↓ Most detailed

Table 1.1: Modeling formalisms operate at varying levels of detail. "Detail" here is taken to refer to the number of reactions (mechanistic detail), inclusion of dynamics (temporal detail), and the presence of discrete molecules (as in stochastic simulations). In the most coarse–grained representation, cellular behavior is represented phenomenologically without a connection to the underlying mechanism. For example, logical models describe the on/off action of genes but not the molecular mechanisms that lead to these states. Additionally, logical and constraint–based models cannot be used to predict the timescale of changes in cellular state. Hence, coarse–grained models can be used to predict some aspects of biological behavior but not necessarily *why* or *how* the behavior occurs on a molecular level. On the other hand, molecular dynamics (MD) simulations can, in principle, account for all molecular interactions in the cell. However, cellular–scale MD simulations are difficult to parameterize (the initial states of all molecules must be known) and are computationally infeasible. Formalisms in between these two extremes provide mechanistic insight by balancing detail against varying levels of accuracy versus computational efficiency. Further, some models exclude temporal information (static formalisms, top part of table) whereas more detailed models incorporate temporal information and can be used to predict the change in system state over time (bottom part of table).

المنارة للاستشارات
www.manaraa.com

not the case [125]. In the study, patients were treated with an HIV protease inhibitor, which caused a rapid fall in plasma virion count and a rapid rise in lymphocyte count. By fitting a model of virus production and clearance to this transient response, Perelson *et al.* calculated that the mean half–life of virions in the blood was about 2.1 days. Hence, production of the virus was not a gradual process as was previously assumed. Instead, an equilibrium state with rapid production and clearance of the virus is maintained during infection. Using this information, Perelson *et al.* calculated the mean production of the virus to be on the order of 10^9. The mutation rate of HIV is approximately one in three virions. This rapid mutation is enough to quickly develop resistance to a single anti–retroviral drug (ARV). However, for three drugs, the expected length of time required for the virus to develop resistance is:

$$3 \cdot \binom{9800}{3} \Big/ 10^9 = 12700 \text{ days} = 35 \text{ years}$$

The number 9800 is the size of the HIV genome in base pairs. The main conclusion — that three–drug therapy is required to overcome resistance, was successfully employed in anti–retroviral combination drug therapy that saved many lives. This example illustrates the importance of kinetic modeling. Despite the simplicity of the model of virus production and clearance used here, it was able to provide counterintuitive results (virions are in equilibrium between rapid production and clearance) that led to clinical strategies.

ODE Models

Ordinary differential equation (ODE) models are a very common type of kinetic modeling formalism. They are based on the bulk movement of large numbers of molecules and/or cells, so that the effects of single molecules become negligible. The rate of a given process (such as a chemical reaction or protein binding) can be described from the law of mass–action. For example, a process with n sources (usually referred to as reactants, but may be interacting proteins or some other source state) are converted into products at the rate ν given by:

المنارة للاستشارات

www.manaraa.com

$$\alpha_1 R_1 + \alpha_2 R_2 + \ldots + \alpha_m R_m \rightleftharpoons \beta_1 P_1 + \beta_2 P_2 + \ldots + \beta_n P_n, \quad \nu = k_f \cdot R_1^{\alpha_1} \cdot R_2^{\alpha_2} \cdot \ldots \cdot R_m^{\alpha_m} - k_r \cdot P_1^{\beta_1} \cdot P_2^{\beta_2} \cdot \ldots \cdot P_n^{\beta_n}$$

where R and P are the reactant and product vectors and α and β are the reactant and product stoichiometries respectively. This equation can be similarly applied to every reaction in the network, thus yielding a set of first order, ordinary differential equations. In order to write a more general form, let $\boldsymbol{s}$ be a length–n vector of species (nodes) in the network, $\boldsymbol{\nu}$ be a vector of rates, and the stoichiometry matrix $\mathbf{N}$ contain the stoichiometry coefficient of $\boldsymbol{s}_j$ in $\boldsymbol{\nu}_k$ in row j and column k. Positive elements of $\mathbf{N}$ are products and negative elements are reactants. The rate of change of the species vector is given by:

$$\dot{\boldsymbol{s}} = \mathbf{N}\boldsymbol{\nu}$$

Conveniently, this same formalism can also be used for lumped processes such as Michaelis–Menten kinetics.

Numerical methods for solving these types of differential equations have been extensively studied and are readily available. One particular notable method is the LSODA solver developed by Petzold and Hindmarsh in 1983 [231, 124], which automatically selects between stiff and non–stiff solvers. Stiffness is a property of a system of differential equations that causes the step size of the solver to become extremely small unless the solver is specifically designed for stiff systems, e.g. using backward differentiation formulas (a more mathematically rigorous definition of stiffness is provided in [231]). Dispite its age, LSODA is still widely used today (see [4] for a modern implementation and [118] for a GPU version), which is a testimate to its reliability for solving systems of differential equations.

Stoichiometry reduction

In order to find the steady state of the network, if one exists, Newton's method can be used. Newton's method locates states where the time derivative of the species vector $\dot{\boldsymbol{s}}$ is zero. Newton's method requires inverting the Jacobian:

المنارة للاستشارات

www.manaraa.com

$$J = \begin{bmatrix} \frac{\partial \dot{s_1}}{\partial s_1} & \frac{\partial \dot{s_1}}{\partial s_2} & \cdots \\ \vdots & \ddots & \\ \frac{\partial \dot{s_n}}{\partial s_1} & & \frac{\partial \dot{s_n}}{\partial s_n} \end{bmatrix}$$

However, this matrix can be singular if any species in the network are not independent of the others. Consider the kinase cascade depicted in Figure 1.5. The kinase A can be phosporylated to A^P, which phosphorylates B to B^P in turn. The total quantities $A + A^P$ and $B + B^P$ are constant in time. The stoichiometry matrix for this system is:

$$\begin{array}{cc} & \begin{array}{cccc} \boldsymbol{\nu_1} & \boldsymbol{\nu_2} & \boldsymbol{\nu_3} & \boldsymbol{\nu_4} \end{array} \\ N = & \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{array}{l} A \\ A^P \\ B \\ B^P \end{array} \end{array}$$

This matrix clearly has rank 2 since the second and fourth rows are inversions of the first and third respectively. We can use the substitutions $A = A_{tot} - A^P$ and $B = B_{tot} - B^P$ to refactor the system as follows [134]:

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

The matrix $\mathbf{N_R}$ is called the reduced stoichiometry matrix and contains only the independent rows of $\mathbf{N}$. In this example, it has rank 2. The matrix $\mathbf{L}$ is called the link matrix and can be computed in general from the reduced stoichiometry matrix:

$$\mathbf{L} = \mathbf{N} \cdot \mathbf{N_R^T} \left(\mathbf{N_R} \cdot \mathbf{N_R^T}\right)^{-1}$$

المنارة للاستشارات

www.manaraa.com

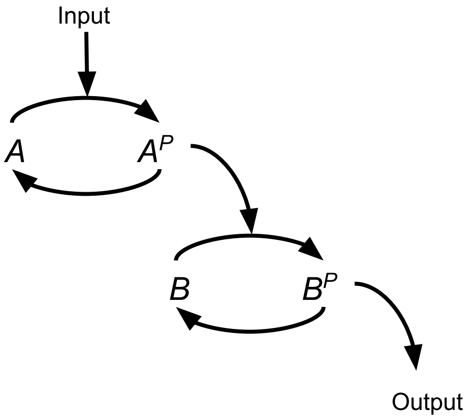


Figure 1.5: The kinase A is phosphorylated to A^P, which then phosphorylates B to B^P. In this kinase cascade, the total quantities $A + A^P$ and $B + B^P$ are constant in time. Hence, $\mathrm{d}A/\mathrm{d}t = -\mathrm{d}A^P/\mathrm{d}t$ and $\mathrm{d}B/\mathrm{d}t = -\mathrm{d}B^P/\mathrm{d}t$. This creates problems for Newton's method, which requires that the Jacobian be invertible.

Further, the independent species $\boldsymbol{s_i} = (A\ B)$ are related to the original species vector $\boldsymbol{s}$ by the link matrix and vector of total quantities $\boldsymbol{t} = (0\ A_{tot}\ 0\ B_{tot})$:

$$\boldsymbol{s} = \mathbf{L} \cdot \boldsymbol{s_i} + \boldsymbol{t}$$

The Jacobian can then be calculated based solely on the two–element vector $\boldsymbol{s_i}$, yielding a 2×2 full–rank matrix that can be inverted to calculate the steady state using Newton's method.

Stochastic Models

Stochastic models represent chemical reactions or binding / unbinding of proteins as discrete events involving individual molecules. They are applicable when the bulk movement of mass represented by ODE models becomes inaccurate. For example, the number of LacI repressors in a single *E. coli* cell has been estimated to be just 10–50 molecules [152]. Hence, when

المنارة للاستشارات
www.manaraa.com

one of these repressors is created or destroyed, the total repressor count changes by as much as 10%. This distinction can have far–reaching counterintuitive effects. For example, it is possible to build a linear amplifier for gene expression without negative feedback simply by relying on noise due to having a low copy number of molecules [152].

Stochastic models are usually simulated using Gillespie's algorithm [108], developed in 1977. The Gillespie algorithm is inherently stochastic and generates a different trajectory each time it is run, but the set of trajectories is statistically correct (having the correct mean and variance for the given model's dynamics) as the number of runs tends toward infinity. The original Gillespie algorithm scaled linearly with the number of reactions in the model. A more recent version [106] (the "Next Reaction Method") scales with the logarithm of the number of reactions but generates the same statistically accurate results under all conditions. Other methods such as tau–leaping [109, 110, 245, 300, 303] trade statistical accuracy for simulation speed.

Figure 1.6 shows examples of multiple stochastic simulations of a system with the same initial state. Despite being behaviorally different from ODE simulations, stochastic simulations require the same set of information: a list of kinetic parameters and processes. The only required change is that initial quantities should be specified in numbers of molecules instead of concentrations. In theory, this conversion can be performed automatically if all species concentrations and the volume of all compartments in the model are both known. As a result of this overlap, stochastic and ODE models can often be used interchangeably. This is useful for testing whether discretization appreciably affects the model's dynamcs, as is the case in Figure 1.6.

Limitations of Kinetic Models

Kinetic models are one of the oldest modeling techniques and their utility for constructing accurate mechanistic models of biological processes is well–established from both a theoretical perspective (using arguments based on underlying physical mechanisms) and an empirical basis (through the development of many instances of accurate, predictive models, at least in

المنارة للاستشارات
www.manaraa.com

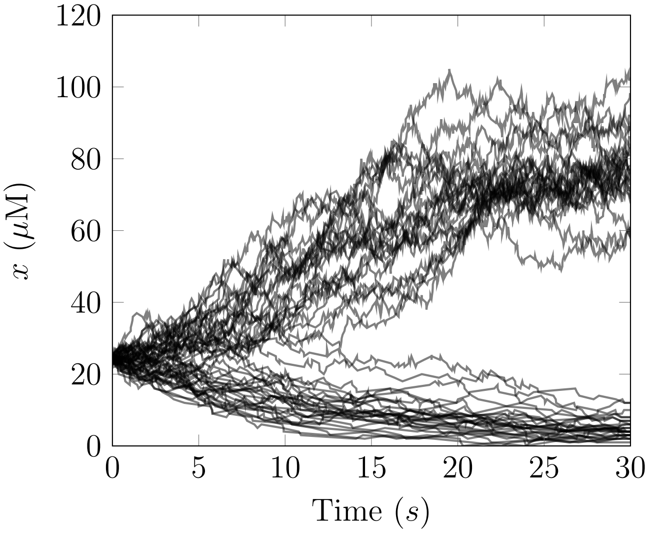


Figure 1.6: A stochastic model generates different output each time it is simulated. This figure shows multiple stochastic simulations of a bistable switch starting at the unstable equilibrium of the system. The simulation randomly converges to either of the two stable equilibria.

www.manaraa.com

cases where parameters can be accurately measured [125, 277]).

However, in larger, more complex models, the parameters are not known and must be determined indirectly using a fitting method. Many models that rely on fitting to determine parameter values are not quantitatively accurate. For example, the repressilator and toggle switch models in Section 1.0.3 both predict the qualitative behavior (oscillations and bistability) of the genetic circuits they are designed to model, but do not accurately predict the exact conditions (i.e. the concentrations of all species in the model) under which these behaviors occur[3]. Terence Hwa, who studies cellular biology from a top–down perspective, once remarked of mechanistic modeling that the predictive capabilities of such models is limited by the fact that the reference state is unknown [132].

The data for parameterizing kinetic models may improve with the advancement of single–cell technologies. However, other bottlenecks exist for kinetic models. Large kinetic models are very expensive to simulate and calibrate compared to their constraint–based counterparts (described below). Fitting parameters to a large kinetic model may take several days or weeks (see Chapter 3 of this thesis). In such cases, determining error bounds on the fitted parameters, which usually requires at least one round of fitting per parameter (usually many hundreds of rounds), is computationally prohibitive. Thus, kinetic models can exhibit scalability problems that make them difficult to apply to whole cells.

1.1.2 Constraint–based Models

Constraint–based models, also frequently referred to as flux–balance models, compute the flux through all reactions in the model subject to a set of constraints in the form of upper and lower limits of the flux of each reaction, as well as an overall objective function to optimize (often based on maximizing the growth rate, but may also be miximizing the production of ATP or some other by–product [248]).

For example, consider a cell that metabolizes nutrient A to produce carbon skeleton

[3] The models can be made to predict the exact conditions through parameter fitting on said conditions, but this would amount to a circular argument.

المنارة للاستشارات
www.manaraa.com

molecules B and C according to Equation 1.2.

$$A \longrightarrow 2B \tag{1.1}$$

$$A \longrightarrow C \tag{1.2}$$

Assuming the reaction is at steady state, we must have:

$$J_A = 2J_B + J_C,$$

where J_A, J_B, and J_C are the fluxes leading into the pathways for A, B, and C respectively (J_A is inverted for clarity). Assume that J_A is limited by nutrient uptake and therefore fixed. Further assume the cell maximizes its growth rate given by the *biomass reaction* $J_B \cdot J_C$.

$$\max\left(J_B \cdot J_C\right) = \max\left(J_C \frac{J_A - J_C}{2}\right)$$

This maximum occurs when $J_A = 4J_B = 2J_C$ (the flux through J_B is half that through J_C, which is expected since one molecule of B requires fewer nutrients to produce than a molecule of C).

Constraint–based modeling takes its name from several eponymous constraints used to define the solution space of the model.

- **Stoichiometric constraints:** The stoichiometry coefficients of reactants and products in all reactions in the model can be used to constrain flux values in different branches as we did in the example model above.

- **Steady–state constraints:** The concentrations of all intermediate metabolites in the pathway can be measured and used to constrain the fluxes to values which yield these concentrations in steady state.

المنارة للاستشارات

www.manaraa.com

- **Abundance constraints:** The transcript or (better yet) protein levels and kinetic parameters (if available) of all enzymes can be used to place relative constraints on the flux catalyzed by the enzyme (the enzyme's V_{max}).

- **Other types of constraints:** Arguments based on the free energy of a reaction can be used to suggest reasonable ranges for the flux under different conditions. This allows constraint–based models to incorporate temperature and structural dependence [327, 122].

The overall effect of these constraints is that the space of acceptable solutions to the flux vector forms a cone in $\mathbb{R}^N$. The fluxes of the N reactions in the network can then be determined (albiet non–uniquely) from an objective function on this flux cone. It is assumed that the cell is optimized by evolution to maximize this objective function, which is usually taken to be the biomass composition of the cell (i.e. the combination of all essential biomolecules in the ratios required by the cell). Cells with optimal production of all essential biomolecules in the correct ratios will exhibit the least waste of energy and thus have the highest growth rate. In a colony of rapidly growing cells in exponential phase, this may well be true, since cells with the highest growth rate will dominate the population and hence be selected by evolution. By assuming that cells optimize their growth in this way, linear programming can be used to find a solution to the flux vector which maximizes this function within the flux cone. However, when cellular crowding occurs, cells often switch into an energy conservation mode characterized by a reduction in both energy consumption and expenditure. In these cases, the assumption of maximum growth rate may be a poor approximation of the cell's state and lead to erroneous results. What assumptions should be used instead under such a situation? The field of constraint–based modeling has yet to provide a general answer to this question, but the inverse problem of determining an objective function given a flux vector has been explored [332] and may eventually reveal strategies for improving the accuracy and fidelity of constraint–based models.

Since no kinetic data is required to construct constraint–based models, they are often

المنارة للاستشارات
www.manaraa.com

used for poorly characterized organisms where data is sparse or non–existent. Another major advantage of constraint–based models is that the linear programming methods used to optimize them are much more computationally efficient than the generalized, global, non–linear optimization required by kinetic models. This allows scaling of constraint–based models to larger sizes and they are frequently used in genome–scale models [257]. Constraint–based models are useful in various "bottom–up" studies and have been used to predict kinetic parameters for all enzymes in *E. coli* [75]. A very large continual community–based effort has yielded a highly–complete genome–scale model of yeast [29].

Limitations of Constraint–Based Models

Many of the advantages of constraint–based models can be equally viewed as disadvantages. For example, the fact that constraint–based models do not require kinetic data makes them easy to construct, but also limits their predictive power, since they ignore changes in the cell over time. Diurnal models have been constructed for photosynthetic organisms by using separate day and night–cycles to approximate the organism's internal circadian clock [259]. However, this relies on *a priori* demarkation of the boundaries of the cycle and bears no connection to the underlying biological mechanisms that drive this change.

Another limitation of constraint–based models is that the assumption of maximum growth is difficult to verify or refute. If the model is constrained using metabolite, flux, and abundance data, the flux cone of acceptable solutions becomes more narrow and the effect of the objective function is diminished, but any variability that remains creates a potential source of error that is difficult to address. Additionally, it is possible that multiple maxima with the same value exist within the flux cone. In this case, the fluxes predicted by the model are non–unique, which further complicates analysis and makes drawing conclusions difficult.

Constraint–based models thus have excellent scalability and are easy to construct. However, they are difficult to validate, they are only predictive of the state of the cell at a single snapshot in time, and their solutions may not be unique. Despite these drawbacks, constraint–based models are routinely used in constructing genome scale models for their util-

المنارة للاستشارات

www.manaraa.com

ity in incorporating large numbers of reactions without requiring kinetic data. Constraint–based models are also used for organisms that have many uncharacterized genes, such as cyanobacteria [259] and other types of exotic organisms that might be useful for biofuel production.

1.1.3 Rule–based Models

Rule–based models are an approach of managing complexity when modeling biological networks. Fundamentally, they are no different from kinetic models. Both approaches describe a chemical reaction network along with associated kinetics, and allow the network to be simulated using either an ODE solver or a stochastic solver such as the Gillespie method. The difference is that rule–based models do not enumerate all possible states of molecules in the network. Proteins can be modified in numerous ways — phosphorylation, glycosylation, and glypiation. Further, there are often numerous sites for the modification. The tumor suppressor protein p53 has between 17 and 20 phosphorylation sites [261, 301]. Proteins can also bind to form multimeric complexes. Some proteins, such as small heat shock proteins, form polydisperse oligomers (i.e. the exact number of subunits fluctuates) [119]. All of these modifications can theoretically alter the network behavior by changing the binding equilibria between proteins / small molecules or enzyme activity. It is unlikely that all states that arise as a result of modification and complexation are necessary to explain network dynamics (the number of phosphorylation states for p53 alone is over 100,000 [261]), but this myriad of states nevertheless creates a significant bookkeeping problem for modeling.

Rule–based models describe the molecules in a system and the ways these molecules can be modified. Modifications occur at user–defined *sites*, such as threonine or tyrosine residues (phosphorylation), or protein binding sites (complexation). Each site can have a number of discrete states, indicating whether phosphorylation or binding has occurred at the specific site. Rule–based modeling software then enumerates all possible molecular states. This formalism can be used to manage complexity. For example, consider the following epidermal growth factor receptor (EGFR) model from the BioNetGen / RuleBender rule–

www.manaraa.com

based modeling platform [43, 280].

```
begin molecule types
    egf(r) # egf, the receptor ligand
    egfr(l,r,Y1068~Y~pY,Y1148~Y~pY) # the egf receptor
    Shc(PTB,Y317~Y~pY) # Shc, Grb2, and Sos are adapter proteins
    Grb2(SH2,SH3)
    Sos(dom)
end molecule types
```

The code above describes all molecules in the system and the associated sites for each molecule. For example, the EGF receptor (`egfr`) has four sites: a ligand binding site (`l`), a dimerization binding site (`r` — the receptor forms autodimers upon ligand binding), and two tyrosine phosphorylation sites. After phosphorylation, the tyrosine residues can then bind to various adapter proteins, which leads to a number of possible complexes. In the fully enumerated network, there are 356 possible combinations of complexes with various states of modification. In contrast, the rule–based model contains just multi–state species, which clearly helps to manage complexity.

1.1.4 Other Types of Models — Logical, Boolean, Hybrid

In certain cases, the behavioral properties of a biological network can be captured by a very simple model where genes exist in on/off states. In a boolean network, the state of each node is a function that maps a subset of nodes (the input nodes, e.g. transcriptional regulators of a given gene) to a true/false value. The model is dynamic in the sense that repeated evaluations of all nodes in the network generate a time series, and it is possible for the network to have a stable steady state or exhibit oscillations. However, the extreme simplicity of these models makes drawing scientific conclusions difficult. Boolean networks have been used in the past to reconstruct plausible transcription factor regulatory networks

المنارة للاستشارات
www.manaraa.com

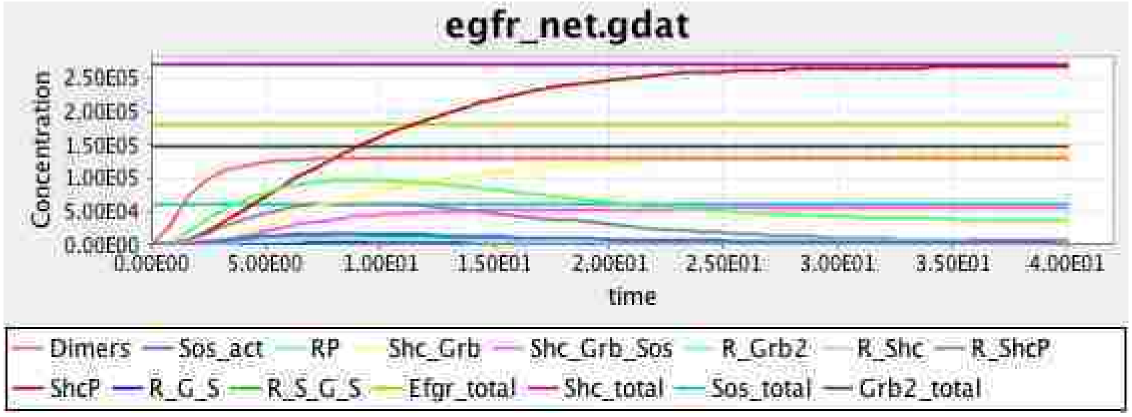


Figure 1.7: The transient response of the rule–based EGFR model from BioNetGen [43, 280]. The plotted quantities are defined by *observables*, which collect related states according to user–defined criteria (for example, the total concentration of all receptor dimers regardless of phosphorylation state or bound adapter proteins).

in yeast in cases where limited data prevented the construction of a more detailed model [147].

The two–state approach of boolean models leaves out important dynamical information. More recent studies employing so–called "constrained fuzzy logic" models [179] have attempted to provide a compromise between the temporal resolution of kinetic models and the simplicity of boolean models. Fuzzy logic is a well–established technique from machine learning that uses values between 0 and 1 to represent uncertainty in classification. It can be thought of as an extension of boolean logic to a continuum of values (not just true or false). Constrained fuzzy logic is a subset of fuzzy logic wherein the transfer functions between nodes are constrained to be Hill functions (in normal fuzzy logic, arbitrary transfer functions can be used). The advantage of this approach is a slight reduction in parameters compared to a kinetic model, and boolean logic can be used in place of complex regulation machinery. Constrained fuzzy logic models only require two parameters per node (species), whereas kinetic models require at least two parameters per reaction, or more for lumped kinetics or

المنارة للاستشارات
www.manaraa.com

allostery. This formalism has been used to model the dynamic response of the EGFR/ERK signaling pathway [179]. However, constrained fuzzy logic models are not widely adopted (being used only by a few groups), are difficult to relate to experimental measurements in a quantitative way, and it is unknown whether they provide significant benefits over boolean models.

The last type of modeling formalism considered here is hybrid models. This not really a formalism, but rather a method of combining different formalisms. Having reviewed the advantages and disadvantages of a number of formalisms up to this point, it is natural to ask whether different formalisms can be combined to yield a more versatile appraoch. Indeed, hybrid modeling was used in the construction of a *M. genitalium* whole–cell model in one high–profile study [145]. This model employs constraint–based modeling for the metabolic pathways of the cell, but uses kinetic models for transcription and translation. At pre–determined time intervals, the constraint–based metabolic model is updated with the abundances of various metabolites, and a new flux vector is calculated. This approach adds flexibility and a better performance over a single kinetic model. However, there are significant drawbacks:

- The constraint–based model may have multiple solutions for a given state, introducing ambiguity.

- Whereas the numerical stability of ODE models is well–studied [69, 231, 124], the numerical stability of hybrid methods has not been investigated. For example, numeric ODE solutions converge as the time axis is subdivided into finer intervals, but for hybrid models this is not necessarily the case.

These issues must be dealt with in order to lend credibility to the output of such models. Other types of hybrid models seek to combine agent–based and kinetic models to create spatial models of colonies and tissues of cells [293]. Recent advances in various modeling formalisms continue to make hybrid models an active and largely unexplored area of research.

المنارة للاستشارات

www.manaraa.com

1.2 Applications: Models in Medicine / Systems Pharmacology

Systems pharmacology seeks to use biological models as predictive tools in guiding drug discovery and patient treatment [305, 304]. This field is closely related to two very important disciplines in pharmacology. Pharmacokinetics (PK) is the study of how drugs are metabolized by the body, and pharmacodynamics (PD) is the study of how these drugs affect the body. These two approaches can be combined to create quantitative PK/PD models, i.e. dynamical models of the action of a drug or combination of drugs and the body's response using an integrated approach. Kinetic PK/PD models have been shown to be very useful in guiding treatments based on detailed predictions of drug levels in the blood, and play an important role in drug discovery. ODE–based kinetic models are a common formalism used to construct PK/PD models (e.g. [222]) owing to their versatility for expressing time–dependent phenomena at various levels of detail. Modeling efforts are likely to be highly central to pharmacology and drug discovery for the foreseeable future.

1.3 Applications: Models in Synthetic Biology

The field of synthetic biology has always made extensive use of models in the development of genetic circuits. Two seminal studies, both published in the year 2000, used kinetic ODE models in the construction of the repressilator [88] and a genetic toggle switch [103]. Modern synthetic biology continues to make use of modeling, although perhaps not to the same degree of rigor. For example, the concept of negative feedback has been used to engineer scale–invariant patterns in bacterial colonies [62]. A PDE model was used to guide engineering of this circuit.

Software tools such as TinkerCell [63] can help alleviate cost by designing and simulating genetic circuits *in silico* before the circuits are built in a lab. As with all aspects of modeling, accurate predictions are hampered by lack of quantitative data. Much work has been done to remedy this by quantifying promoter and ribosome binding site (RBS) activities to allow quantitative modeling of expression [49, 51, 50, 254], which has enabled extremely versatile,

المنارة للاستشارات
www.manaraa.com

scalable sequential logic to be implemented using genetic circuits [24]. Modeling will likely play an important role in scaling up genetic circuits in the future.

1.4 Parameterizing Models

The parameter values of systems biology models generally do not come exclusively from experimental measurements (although see [298, 278] for interesting counterexamples) due to the laborious nature of *in vitro* characterization methods, difficulty in purifying components, and potential disagreement between *in vitro* and *in vivo* component behavior. Thus, most models use a more integrative formalism by measuring bulk properties of the system, such as the transient response of key components, and fit the model parameters to this data using optimization methods. Parameter fitting is accomplished by using an *objective function* that quantifies the discrepancy between model predictions and measured data. The type of data required depends on the model formalism.

- Kinetic models require timecourse data of key nodes in the network (e.g. metabolite levels or protein phosphoforms). The model can be further constrained by measuring initial conditions, a subset of parameter values (e.g. using *in vitro* reconstitution), and possibly inequality constraints [203]. The objective function in such cases is usually a variant of the sum of squared errors (SSE) between the model timecourse predictions and the data.

- Constraint–based models can make use of steady–state levels of metabolites and abundances of proteins for a given cellular state.

- Mappings in boolean models can be inferred from gene network states under many different conditions.

This thesis focuses primarily on kinetic models, so the following parameter fitting considerations apply chiefly to these types of models. Fitting ODE dynamical models is a

المنارة للاستشارات
www.manaraa.com

challenging non–convex, non–linear optimization problem. The presence of local minima prevents the use of algorithms such as gradient descent, which is popular in other fields such as machine learning. An ideal solution to this problem would be an algorithm that robustly locates the global minimum of the objective function. However, in reality, this cannot be used as a convergence criterion because the global minimum is not known *a priori*. Instead, convergence is usually measured using some arbitrary stopping criterion, such as stagnation of the algorithm for a certain number of iterations. This work in this thesis uses a threshold objective function value as a stopping criterion to ensure consistent comparison.

Biologically inspired population–based algorithms have emerged as an effective means of solving the non–convex problem of parameter fitting for kinetic models. These algorithms mimic biological evolution [76] or flocking behavior as observed in swarms of insects or flocks of birds [149, 144]. The algorithms avoid becoming trapped in local minima by maintaining a diverse population of candidate solutions at every iteration, and usually feature the ability to combine two or more solutions to generate a potentially better solution. A non–exhaustive summary of these methods is given in Table 1.2. This thesis presents a highly scalable approach to fitting large models in Chapter 3.

1.4.1 Software for Parameter Estimation

Parameter fitting is an essential step in the development of most kinetic models and hence a large number of software tools exist for solving this problem. One very well–known project is the Data2Dynamics (D2D) engine implemented in Matlab [247, 246, 284]. D2D uses a variety of mostly local optimization algorithms with a multi–start approach to locate global minima [246]. The principal feature of D2D is that it provides convenience features for calculating sensitivities and various statistics for fitted parameters. However, the use of local algorithms (i.e. algorithms that converge to a local minimum) limits the scalability of this approach and it is unsuitable for the large, challenging models considered here (see Chapter 3).

A recently published solution that uses global algorithms is PyBNF [203]. PyBNF uses a handful of local and global algorithms running in a distributed fashion on different nodes to

المنارة للاستشارات
www.manaraa.com

Table 1.2: A summary of population–based optimization algorithms used to fit biochemical models.

Algorithm
Genetic algorihms (GAs) [76]
Differential evolution (DE) [290]
Particle swarm [149]
Artificial bee colony [144]
Harmony search [105]
Simulated annealing [72]

Interested readers are referred to a more complete summary at `https://esa.github.io/pagmo2/docs/algorithm_list.html`.

achieve a speedup. This is similar to the work presented in Chapter 3 due to the overlap in some algorithms (e.g. differential evolution is used by both methods), but PyBNF includes fewer algorithms and has not been tested on models as large as the benchmark described in Chapter 3.

1.4.2 Identifiability, Uncertainty and Sloppy Models

In many cases, it is not possible to uniquely infer the values of certain parameters from data. Consider a simple production / degradation reaction:

$$\varnothing \xrightarrow{k_p} \mathrm{SS} \xrightarrow{k_d} \varnothing$$

The rate of change of S is given by:

$$\frac{\mathrm{d}S}{\mathrm{d}t} = k_p - k_d S$$

المنارة للاستشارات

www.manaraa.com

If the only data points are measured at equilibrium, the values of k_p and k_d can only be determined up to a ratio (the equilibrium constant for this reaction). The parameters k_p and k_d are then said to be structurally non–identifiable. This obviously poses a problem for parameter fitting because k_p and k_d can, in general, have vastly different values for each parameter fitting attempt. This also limits the model's predictivity outside of its fitting data. In order to be useful, mechanistic models should ideally be able to predict states that are not directly observed.

A closely related concept is *sloppiness* [302], which states that the output of a model depends on certain combinations of parameter values, and that (within constraints) several different combinations may exist that give the same output. Finally, uncertainty is a term that refers to the range of values for a given parameter that satisfy a given data set. Uncertainty, identifiability, and sloppiness are all closely related concepts. A famous quote attributed to George Box is: “All models are wrong, but some models are useful”. Uncertainty quantification tells us how “useful” the model is on a parameter–by–parameter basis. A parameter with a large uncertainty is usually not identifiable and thus is not “useful” for understanding the underlying mechanism. Conversely, a parameter with a low uncertainty is strong evidence that the mechanism it is attached to is an accurate representation of the underlying biological process. The starting point in analyzing these properties is to use one or more of the methods described below.

1.4.3 Multi–start Optimization

While not a true uncertainty quantification technique, multi–start optimization is occasionally used as a stand–in due to its ease of use. For multiple runs of an optimization algorithm with random initial conditions, parameters that converge to different values are taken to be uncertain. However, the converse is false except in the absence of measurement noise. For example, consider a model fitted to a noisy data set. The fitting algorithm should ideally find the same minimum of the objective function in each run. Thus, any identifiable parameters should always converge to the same value provided the algorithm completely converges.

www.manaraa.com

Thus, multi–start optimization would imply no uncertainty in these parameters, which is not correct when measurement noise is present.

1.4.4 Fisher Information Matrix

The Fisher information matrix is a well-known and frequently used way to account for identifiability. Let y refer to the output of a model and $\boldsymbol{\theta}$ be a vector of the model's real–valued parameters. Then, the Fisher matrix is defined as [302] (some sources use the logarithm of y):

$$\mathcal{L}_{ij} = \frac{\partial y}{\partial \theta_i}\frac{\partial y}{\partial \theta_j}$$

The Fisher information is defined by the variance / covariance of different parameters. A parameter with a large variance has poor identifiability, while one with a small variance has good identifiability. This is closely related to the concept of *sensitivity analysis*, which is a systems biology term used to measure the impact of a parameter or species [255]. Sensitivity analysis typically uses total derivatives instead of the partial derivatives used in the FIM, and often does not consider cross–correlations (the non–diagonal terms in the FIM). However, in practice, the measured quantities will be system–level observables, such as oscillation timings, peak height, or steady state values, in which case the reported FIM must also necessarily be based on total derivatives, meaning that the FIM and sensitivity analysis are computed in the same way.

1.4.5 Bootstrapping

Bootstrapping is a powerful technique from statistics that can be used for uncertainty analysis for a given data set. First, the model is fit to the entire data set and the residuals between the fitted model and the data are calculated. Then, subsequent fitting runs are performed using a data set constructed by simulating the model and corrupting the simulation results with residuals sampled from the original run with replacement. The procedure is shown in

المنارة للاستشارات
www.manaraa.com

pseudocode in Figure 1.8.

```
# calculate residuals from initial data set
model,residuals = fit(initial_dataset)
# run the bootstrapping method to generate new residuals
repeat N times:
  # create a new dataset based on the simulation results corrupted with noise
  new_dataset = model.simulate() + sample_with_replacement(residuals)
```

Figure 1.8: Pseudocode for the bootstrapping method.

The subsequent optimizations after the initial optimization can be performed *ab initio* or they can use the initial fit as a starting point (called local bootstrapping [99]). The uncertainty in parameter values is derived from the collection of parameter estimates produced by bootstrapping. A limitation of bootstrapping is that it requires a large number of optimizations, usually at least 1000–10,000, in order to produce reliable estimates.

1.4.6 Profile Likelihood

The profile likelihood is yet another method for measuring uncertainty. In order to understand the profile likelihood, it helps to think of the optimization process as a maximum likelihood estimator [4]. Using this terminology, maximum likelihood can be equated with the optimum parameter values. Now consider a constrained optimization obtained by fixing a single parameter value, say $\boldsymbol{\theta}_i$, while allowing the others to converge. The optimum parameter values $\boldsymbol{\theta}_{i \neq j}$ are used to compute the objective function minimum:

$$\min \mathcal{J}(\boldsymbol{\theta}_{i \neq j})$$

[4] the likelihood is defined as the probability of observing the given data from a certain set of parameters, so this is equivalent to optimizing the parameter values so that the probability of observing the given data is maximized

المنارة للاستشارات
www.manaraa.com

The profile likelihood is then defined by the ratio [99]:

$$\ln\left(\mathrm{PL}\left(\boldsymbol{\theta}_{i\neq j}, c\right)\right) = \frac{\min \mathcal{J}(\boldsymbol{\theta})}{\min \mathcal{J}(\boldsymbol{\theta}_{i\neq j})}$$

In other words, the profile likelihood is the ratio of the likelihood of the constrained optimization problem to the unconstrained problem. A limitation of the profile likelihood is that it cannot be used to compute parameter cross–correlations.

Fröhlich *et al.* compared the results of the FIM, bootstrapping, and profile likelihood, and found that only profile likelihood yielded accurate estimates of parameter uncertainties, even for simple models [99].

1.4.7 Bayesian Methods

Bayesian statistics are widely used throughout many fields. Bayesian methods differ from the methods discussed thus far in that they accept as input a *prior distribution* for each parameter (instead of a range of values) and output a *posterior distribution* (instead of a single value). The prior distribution represents parameter estimates based on evidence prior to the current fitting data. For example, if there are several conflicting literature reports of a parameter, the prior can be chosen to encompass all of them. Many studies simply base the prior on measurement noise or (worse yet) use a fabricated distribution based on intuition, but this is abuse of the Bayesian method. The Bayesian method should only be used when the prior distributions are founded on credible evidence, such as literature reports. The posterior distribution can be thought of as a modified version of the prior that takes into account the current fitting data. For example, the mean or standard deviation may be shifted if a given parameter needs adjustment to fit the data.

The Bayesian method fills dual roles of parameter estimation and uncertainty quantification. Bayesian methods have several useful properties: they avoid overfitting, and they allow more flexibility than other methods by allowing the user to specify a prior. Given these advantages, it may seem strange that Bayesian methods are not used more often. The reason has to do with performance and the difficulty in obtaining reliable priors. In systems biology,

المنارة للاستشارات
www.manaraa.com

the Bayesian method is often implemented using sequential Monte Carlo algorithms, which are much more computationally intensive than the general non–linear fitting algorithms discussed above. Furthermore, as already remarked, choosing vacuous priors eliminates the most significant benefit of the Bayesian method. If the input to a method is faulty, then the output will be unreliable. It is unfortunate that many studies nevertheless employ the Bayesian method even in the absence of credible priors. Bayesian methods thus represent a useful approach when there is sufficient evidence for the priors and the complexity of the system being analyzed does not exceed the available computing resources. A Python framework for Bayesian computation in systems biology models is described in [172].

1.5 Summary of Chapters and Scientific Contributions

This chapter serves as a general introduction to systems biology and common modeling formalisms and methods for building and calibrating models. The remainder of this thesis is organized into three specific aims:

- Aim 1: Enable better reproducibility via the development of a standards–centric modeling platform.

- Aim 2: Provide a scalable method for fitting large, complex kinetic models.

- Aim 3: Create a compiler for converting between models and special–purpose, high–performance simulation hardware.

Taken together, these aims represent foundational advances in enabling larger, more robust, and more scalable systems biology models. Contributions to each of these aims are described in the respective chapters below.

1.5.1 Contributions to Scientific Reproducibility

Reproducibility is a cornerstone of the scientific method. In systems biology, this is especially true since researchers often need to build on or directly integrate prior models in order to

المنارة للاستشارات
www.manaraa.com

build larger, more comprehensive, and more accurate models. To achieve reproducibility, models need to be stored in a form that is exchangeable, transparent, and usable long into the future. A major contribution of this thesis is the development of a platform that provides facilities for designing models using systems biology standards, which achieves the requirements of future–proofing and transparency. Chapter 2 describes the contribution, provides a technical definition of reproducibility, and shows how standards–centric modeling is crucial for scientific progress in systems biology.

1.5.2 Aim 2: Contributions to Scalability of Kinetic Modeling

Parameter fitting is necessary for the development of accurate models. However, as the complexity of a model increases, the computational cost of fitting the model becomes intractable. Chapter 3 describes a contribution to model fitting that uses an innovative form of parallelism to enable fitting of very large kinetic models. The method is rigorously tested on an extremely large, challenging benchmark and is able to achieve improvements to both performance and quality of fit. This contribution is a key step in enabling the construction of larger, more comprehensive models in systems biology.

1.5.3 Aim 3: Contributions to Foundational Advances in Modeling Technologies

It is well–known that special–purpose hardware can often outperform general–purpose CPUs for a specific task. Chapter 4 presents a contribution that allows kinetic models to be simulated on special–purpose hardware specifically designed for systems biology. This is a key step in enabling broader adoption of special–purpose hardware, which will in turn allow the simulation of larger, more complex models with better performance than general CPUs. This is also a significant contribution to the field of biomimicry, as it provides an important bridge from model descriptions to hardware. The lessons learned from this project will be useful in the design of future biomimetic hardware and software.

المنارة للاستشارات
www.manaraa.com

Chapter 2

A PLATFORM FOR REPRODUCIBLE, DYNAMICAL MODELING IN SYSTEMS BIOLOGY

The contents of this chapter are largely based on a manuscript published in *PLoS Computational Biology* [196]. The curriculum vitae at the end of this thesis lists all manuscripts and resp. statuses.

This chapter relies heavily on the role of standards in systems biology. The central contribution is a modeling platform that achieves better reproducibility through integration with standards. Thus, it is appropriate to first discuss why standards are necessary.

Standards are widely used throughout scientific research. In systems biology, standards are especially important because model reuse is so common. In fact, the BioModels database [164, 168] (a large database of systems biology models with over 1676 entries) exists solely for the purpose of curating and hosting published models. Central to the existence of such repositories is the availability of a common, standard format for entires. Assume, for argument's sake, that BioModels instead hosted models written in Matlab, C, and various other languages. If a research wishes to run a published model written in Matlab 2007, the user may need to download an older version of Matlab to run the model [1]. However, older versions may not work on newer operating systems. Furthermore, models written in C may only run on certain platforms and depend on libraries that are no longer available. Clearly, this would create problems for future–proofing. Scientific reproducibility generally requires

[1] https://www.mathworks.com/matlabcentral/answers/99265-how-do-i-download-an-older-release-of-matlab

المنارة للاستشارات

www.manaraa.com

the ability to reproduce a result long into the future [2]. It would also make curation much more difficult because BioModels staff would need to be familiar with a variety of languages. Thus, a standard encoding is necessary.

There are additional benefits to a standard encoding besides enabling large repositories. Standard–encoded models are also *transparent*, in that they allow the number of reactions, species, and other elements to be interrogated automatically. This information would otherwise be buried in lines of Matlab code (requiring manual intervention by a human to extract). Additionally, standard–encoded models can use *annotations* (which are essentially links from model elements to resources) to unambiguously describe the biological molecules, enzymes, and cellular components contained in the model. For example, the ChEBI database [78, 120, 121] contains an entry that unambiguously represents adenosine triphosphate (ATP) [3]. By linking entries such as this, a standard–encoded model can provide a wealth of information that can be extracted by automatic data mining tools.

2.1 Survey of Standards

This section introduces various systems biology standards. Before describing the contributions of this chapter, it is useful to review each standard and its intended use–cases.

2.1.1 SBML and CellML

The Systems Biology Markup Language (SBML) [131] is a standard for describing chemical reaction networks. Originally designed for kinetic ODE and stochastic models, SBML has since expanded to cover constraint–based, logical (boolean), rule–based, and spatial models. SBML's flexibility comes from *extensions*, which are optional additions to the standard for

[2] On closer examination, this is not completely true. As an aside, it is interesting to note that recent landmark results from physics, such as the confirmation of the Higgs boson and detection of gravitational waves, suffer from much worse reproducibility woes than systems biology. However, the sheer number of researchers involved in rigorously carrying out these experiments means the results are unlikely to be contended.

[3] http://identifiers.org/CHEBI:15422

www.manaraa.com

support various modeling formalisms (e.g. the aforementioned constraint–based models etc.). Since extensions are optional, software that supports SBML is not required to implement support for them. This can also be seen as a disadvantage, since it fragments the standard and the supporting software base.

SBML is encoded in XML (see glossary). Consider a simple model consisting of the reversible binding of allolactose to the LacI repressor in bacteria. This binding process is called a *reaction* in SBML, even though no covalent bonds are broken or formed. The reaction is:

$$\mathrm{a} + \mathrm{L} \underset{k_r}{\overset{k_f}{\rightleftharpoons}} \mathrm{C}$$

where a is allolactose, L is the LacI repressor, and C is the bound complex. The kinetic law is:

$$\nu = k_f a \cdot L - k_r C$$

Finally, let the initial concentrations of a and L be 10 and 1 respectively (in order to keep this example simple, arbitrary units are used). The listing below shows how this model would be represented in SBML.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1">
  <model id="mymodel" name="mymodel">
    <listOfCompartments>
      <compartment sboTerm="SBO:0000410" id="default_compartment" spatialDimensions="3"
        size="1" constant="true"/>
    </listOfCompartments>
    <listOfSpecies>
      <species id="a" compartment="default_compartment" initialConcentration="10"
        hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
      <species id="L" compartment="default_compartment" initialConcentration="1"
        hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
      <species id="C" compartment="default_compartment" hasOnlySubstanceUnits="false"
        boundaryCondition="false" constant="false"/>
    </listOfSpecies>
    <listOfParameters>
      <parameter id="kf" constant="true"/>
      <parameter id="kr" constant="true"/>
    </listOfParameters>
    <listOfReactions>
      <reaction id="_J0" reversible="true" fast="false">
        <listOfReactants>
          <speciesReference species="a" stoichiometry="1" constant="true"/>
          <speciesReference species="L" stoichiometry="1" constant="true"/>
```

المنارة للاستشارات

```
        </listOfReactants>
        <listOfProducts>
          <speciesReference species="C" stoichiometry="1" constant="true"/>
        </listOfProducts>
        <kineticLaw>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <minus/>
              <apply>
                <times/>
                <ci> kf </ci>
                <ci> a </ci>
                <ci> L </ci>
              </apply>
              <apply>
                <times/>
                <ci> kr </ci>
                <ci> C </ci>
              </apply>
            </apply>
          </math>
        </kineticLaw>
      </reaction>
    </listOfReactions>
  </model>
</sbml>
```

Even for this simple model, the SBML appears quite complicated. Let us examine each component. First, SBML requires at least one *compartment*, which specifies the physical place where the reaction occurs (e.g. cytoplasm, nucleus). Upon exporting to SBML, the required compartment is automatically inserted and given the id `default_compartment`. SBML organizes all top–level elements into lists. In this case, `listOfCompartments` only contains one element:

```
<listOfCompartments>
  <compartment sboTerm="SBO:0000410" id="default_compartment" spatialDimensions="3"
    size="1" constant="true"/>
</listOfCompartments>
```

The list of species contains the a, L, and C variables we defined earlier. Note that `initialConcentration` is 10 for a and 1 for L as required. There is no `initialConcentration` for C so it is assumed to be zero.

```
<listOfSpecies>
  <species id="a" compartment="default_compartment" initialConcentration="10"
    hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
  <species id="L" compartment="default_compartment" initialConcentration="1"
    hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
  <species id="C" compartment="default_compartment" hasOnlySubstanceUnits="false"
```

www.manaraa.com

```
    boundaryCondition="false" constant="false"/>
</listOfSpecies>
```

The `listOfParameters` contains the forward and reverse rate constants k_f and k_r:

```
<listOfParameters>
  <parameter id="kf" constant="true"/>
  <parameter id="kr" constant="true"/>
</listOfParameters>
```

Finally, the list of reactions contains a single reaction with id `_J0` (generated automatically). This reaction is where much of the complexity in the SBML encoding comes from. The kinetic law is described using yet another standard, MathML (`https://www.w3.org/Math/`), which is an XML standard designed for representing mathematical expressions. It is useful to reuse other standards in order to build on prior work (i.e. avoid "reinventing the wheel"). However, MathML is itself a complex standard, and this thesis will not endeavor to cover it here. It is sufficient to assume that mathematical expressions *can* be expressed, somehow, in this standard without concerning ourselves with the details of the representation.

Ignoring the large chunk of MathML, it is clear that SBML consists of *compartments*, *species*, *reactions*, and *parameters*, and that these elements are organized into lists in the document. Many other types of elements are possible (particularly if extensions are used), but the aforementioned four element types are the most common in SBML. When broken down and analyzed in this way, SBML is really a very simple and transparent standard. We can already begin to see the advantages of SBML over Matlab. For example, the organization of elements into lists makes it easy to write a script to read the number of reactions in a model and print them out one–by–one, whereas otherwise this information would be buried in Matlab source files and would require a great deal of effort to parse with a script (possibly by matching certain patterns, which is a fragile approach).

CellML [74] is also an XML standard for storing biological models. Whereas SBML is designed to represent chemical reaction networks, CellML is designed to represent physiological models. Accordingly, CellML does not use "reaction" elements, since the higher–level physiological processes encoded in CellML rarely contain a reaction–by–reaction description of the process's biochemistry. Instead, all rates of change are encoded using ODEs. Unlike

المنارة للاستشارات
www.manaraa.com

SBML, CellML is designed in tandem with its own normative simulator, OpenCOR [104] (SBML does not have a normative simulator — instead it has a comprehensive test suite that tests simulator compliance with the standard). Since CellML is not heavily used in this work, it is only mentioned in passing here.

2.1.2 Antimony

Though technically not a standard, Antimony is a human–readable abstraction of SBML that is used used extensively in this chapter. A major problem with the above SBML example is that, even for a trivial model, the SBML encoding is too cumbersome to write by hand. There are libraries to assist with reading and writing SBML [48, 165, 82], but another approach is to create a scripting language that can be interconverted with SBML. Antimony is such a language [4]. The previous LacI binding model can be expressed in Antimony as follows:

```
model mymodel
  a + L -> C; kf*a*L - kr*C
  a = 10
  L = 1
end
```

Clearly, this is considerably more parsimonious than the SBML encoding shown previously (5 lines for Antimony vs. 47 for SBML). However, this code can be interconverted with SBML using the Antimony library / Python package. Thus, any edits made to the Antimony code will be incorporated into the SBML model.

2.1.3 SED–ML

The SED–ML standard is closely tied to SBML and CellML. Whereas SBML and CellML describe models, SED–ML describes how to simulate models. This can include specifying

[4] Technically, Antimony is not a scripting language. It is a human–readable abstraction of SBML. However, Antimony fulfills the same role as a scripting language: it provides simple syntax to mask underlying complexity.

المنارة للاستشارات
www.manaraa.com

what algorithm should be used for simulation, whether the model should be simulated over time or simply used to calculate the steady state, and specifying all parameters used by the algorithm(s). The main motivation for storing this information is to enable automated reproduction of a previous result without human intervention. For example, if a researcher wishes to recreate a previously published figure, the researcher can download the SBML model from the BioModels data. However, the researcher must manually guess what simulator parameters (e.g. integrator step size) are required to reproduce the figure. Additionally, different figures may use different parameter values. Ideally, this would be noted in the original paper, but this is not always the case. SED–ML captures these algorithmic details to enable fully automatic reproduction of a previous result, and also provides a few additional features discussed in the examples in this chapter.

2.1.4 SBOL

Though not used extensively in this chapter, the Synthetic Biology Open Language (SBOL) [102, 30] is an important standard for biological parts. SBOL stores sequence data along with information describing the biological parts it belongs to. For example, the LacI repressor described previously could be stored as a sequence with the promoter region, ribosome binding site, and terminator delineated and linked to corresponding standard sequences for each respective component. This is in contrast to older formats such as GenBank (https://www.ncbi.nlm.nih.gov/genbank/) that only store sequence. The iGEM Parts Registry (`http://parts.igem.org/Main_Page`) contains over 20,000 parts. SBOL allows users to better utilize these parts by providing hierarchical semantic information that can be used by computer aided design (CAD) tools to help design plasmids to perform a certain function. In a way, SBOL can be thought of as an advanced system for sequence annotation, albiet one that uses a powerful linked–data approach to enable better informatics and design approaches.

المنارة للاستشارات

www.manaraa.com

2.1.5 COMBINE Archives

It is clear from the preceding few paragraphs that there appears to be a proliferation of standards. In order to allow researchers to share their work with others, it is useful to have a way of organizing and packaging together related work. This is especially true for SED–ML, which relies on having an SBML or CellML file in tandem within the same directory in order to reproduce a result. This is a case where it would be better from a usability standpoint to have a single file, but this is problematic because SBML, CellML, and SED–ML are described by different standards. COMBINE archives provide a solution to this problem by allowing different standards to be packaged together in the same file. A COMBINE archive is essentially a compressed file format (similar to a `.tar` or `.zip` archive) that contains metadata describing the contained files and pointing to the respective standards they represent. For example, if a COMBINE archive is created from a single SBML file and a single SED–ML file, the metadata will have one entry for each with a pointer to each respective standard. This metadata information can then be used to decide what to do with each entry when opening a COMBINE archive.

2.1.6 COMBINE Metadata

A recent proposal has suggested using COMBINE archive metadata to also store annotations [208]. This would allow storing annotation information in a uniform way across different model types (SBML and CellML). Recalling that semantic annotations allow linking of models to biology (by unambiguously identifying the exact biological molecules and cellular structures represented by variables), this would be very beneficial for improving the utility of models. By storing the annotation information in the COMBINE archive metadata, this would serve to separate annotations from standards and allow more freedom for the evolution of semantic annotations.

Having reviewed these important systems biology standards, we next proceed to examine the role of standards in reproducibility.

المنارة للاستشارات
www.manaraa.com

2.2 Background

One goal of systems biology is to construct and simulate large, multiscale models. Examples include the *Mycoplasma genitalium* whole–cell model [145] and the central metabolism of *E. coli* [201]. These models are often composed of many submodels. However, in order to be useful, these submodels must be extensively validated to ensure good agreement with experimental results. Typically, development and validation of a submodel is a very time consuming process and thus necessarily is performed by respective experts of each biological subsystem. Thus, an imminent goal of systems biology is to allow these different submodels to be reused and combined. Indeed, without the ability to reuse existing models, constructing larger models becomes impractical.

Being able to reuse tools and techniques developed by others is a hallmark of science. Poor reproducibility of biomedical experimental studies has been recognized as a major impediment to scientific progress [238, 204]. Much of the focus on poor reproducibility has been on wet lab experiments. However, barriers to reproducibility is also a significant problem in computational studies [225, 167, 197, 187, 314, 286]. In recognition of this problem, **reproducibility** has become a central focus of scientific software [226, 258]. A similar problem in poor reproducibility also exists for systems biology models. Difficulty in model reproducibility can result from a published model not being deposited in a public repository or from differences in the deposited model and the actual model used for published simulations. In addition, it is difficult for researchers to utilize and modify public models because the standards are not human–readable. This state of affairs creates barries to the continued development of biological models.

The first aim of this thesis is to address reproducibility through the use of standards. Reproducible computational studies must satisfy two requirements. First, they must be *transparent*; that is, researchers must be able to inspect and understand the details of the model and the computational experiments. With transparency, researchers can check assumptions and explore variations in computational studies. Second, computational studies

www.manaraa.com

must be *exchangeable*; that is, it must be possible for a study done in one computational environment to be done in another computational environment and produce comparable results. For a study to be exchangeable means that other researchers can make use of and build on the published results in their computational environment.

In order to be transparent and exchangeable, a computational model and any simulation experiments must be encoded in a standard format that separates the reusable part of a model and its simulations (i.e., parameters, processes, and kinetics) from the implementation used to simulate it (i.e., the numerical methods and algorithms used to generate results). Models can be described using the Systems Biology Markup Language (SBML) [131] or CellML [74] standards. These standards support models based on ordinary differential equations (ODEs), stochastic master equations, and constraint-based modeling [216, 217], partial differential equations (PDEs, using the proposed geometry extension [11, 12]), etc. Simulations can be described using the Simulation Experiment Description Markup Language (SED–ML) [313], which encodes the types of simulations, either time-course simulations or steady state computations, that should be run on a model. SED–ML allows specifying the exact numerical algorithms needed to run a simulation using the Kinetic Simulation Algorithm Ontology (KiSAO) [73], which includes widely used ODE (e.g., LSODA [231, 124], CVODE [69]) and stochastic solvers (e.g., Gillespie direct method [108], Gibson algorithm [106]).

In order to facilitate exchanging models and simulations between software tools, SED–ML simulations and SBML/CellML models can be packaged together using COMBINE archives [36]. However, few authoring tools exist for SED–ML and COMBINE archives [38, 268]. Furthermore, existing resources require technical knowledge of standards, restricting use of these standards by the modeling community at large. Therefore, an authoring tool is needed that allows a wider range of users to create and edit COMBINE archives containing both models and simulations. An authoring tool should satisfy five requirements:

1. It should represent the models or simulation specifications in a human–readable form.

2. It should allow the user to easily edit this human–readable representation.

المنارة للاستشارات

www.manaraa.com

3. It should allow the user to provide narrative, annotations, or comments in order to improve transparency.

4. It should translate the specifications into an implementation that can be used to run simulations.

5. It must be capable of repackaging the model and/or simulation in a standard form that is usable by other tools.

Tellurium Notebook environment was designed to satisfy these requirements, and it extends the literate notebook concept used by tools like Jupyter [243] and Mathematica [322] to support community standards in systems biology. Whereas Jupyter notebooks contain code and narrative cells, Tellurium adds a third cell type for representing models and simulations encoded as standards. Our tool allows modeling studies to be constructed in a notebook environment and exported using community standards. This workflow provides both transparency, through a literate notebook, and exchangeability, through seamless, fluid support for standards.

Tellurium supports embedding human–readable representations of SBML [281] and SED–ML [67] directly in cells. These cells can be exported as COMBINE archives which other tools can read. This human–readable representation is referred to as *inline OMEX* (after Open Modeling and EXchange, the encoding standard used by COMBINE archives). Inline OMEX cells operate in much the same way as code cells, i.e., they have syntax highlighting and are executable. Executing an inline OMEX cell runs all SED–ML simulations in the cell, producing any plots or reports declared in the SED–ML. A major advantage of this approach is that it offers a means of authoring transparent, exchangeable modeling studies without requiring technical knowledge of file format standards.

المنارة للاستشارات

www.manaraa.com

2.3 Design and Implementation of Tellurium

Computational tools in systems biology typically focus on one of three major areas: authoring models, simulating models, or visualizing network–based depictions of models. Tellurium is primarily a tool for authoring and simulating models. As such, it must satisfy certain requirements in order to be useful for model construction. To construct a dynamical model, it is necessary to translate biological measurements and observations into a mathematical language that can be used to derive a set of differential equations. One method of constructing dynamical models in systems biology is to survey the literature for known interactions, rate constants, and parameters (the so–called "bottom-up" approach [155]). Thus, known information about a system can be used to construct a list of reactions, concentrations, and kinetic parameters, which, in turn, can be used to derive a set of differential equations. However, the list of known kinetic parameters is usually incomplete. Depending on the available data, some parameters may have to be inferred indirectly, while others may be completely unknown. A common method of addressing this problem is to perform parameter fitting, i.e., use a numerical optimization algorithm to minimize the discrepancy between the model's predictions and available data by tuning parameters. To be useful, fitting parameters requires prudent selection of which parameters to optimize, as well as the numerical algorithm(s) to use and the upper and lower bounds for the selected parameters. As a result, parameter fitting can require considerable expertise and forethought, although attempts have been made to systemitize the process [151].

Tellurium's principal feature is that it allows integrating standards such as SBML, CellML, and SED–ML in a single, unified representation called inline OMEX cells. This approach has the advantage of allowing users to encode models in these standards without necessarily being technical experts in said standards. While Tellurium could in principle have been designed to provide separate cell types for Antimony and PhraSEDML, a single, unified editable representation is a better solution for three reasons: 1) With a unified representation, users are not required to know the technical boundaries between standards, 2) exchanging

المنارة للاستشارات
www.manaraa.com

Table 2.1: **Standards and terminology at a glance.**

Standard / Technology	**Description**
SBML and CellML	Standards and respective file formats for systems biology models. Originally designed to support ODE models, but have since expanded to cover other types of models.
SED–ML	Standard and file format for specifying a simulation to be run over an SBML/CellML model. Can specify simulation algorithm and parameters.
libSBML and libSEDML	Import/export of respective standard–encoded files.
COMBINE	COmputational Modeling in BIology NEtwork (COMBINE) is an initiative to coordinate the development of standards (such as SBML, CellML, and SED–ML) and formats for modeling in biology.
COMBINE Archive / OMEX	A zip file–like container for standards covered by COMBINE. Also known by the moniker OMEX (Open Modeling and EXchange format).
Antimony / PhraSEDML	Human–readable representation of standards (Antimony for SBML/CellML and PhraSEDML for SED–ML). Provided as software libraries that can interconvert between human–readable form and standard encoding.
nteract	Notebook viewer featuring code and narrative.
libRoadRunner	Library / Python package for simulation of SBML ODE and stochastic models.
Jupyter	An umbrella project for technologies for authoring, rendering, and supporting literate coding notebooks. Provides a message protocol which can be used by any language to interface with Jupyter notebooks.
Plotly [276]	A set of software packages for rendering interactive plots using web technologies (HTML/-Javascript). Used by Tellurium to provide high–quality plots in the notebook environment.
Tellurium	Integrated Python environment that includes or supports all of the above technologies and standards.

Integration is a major design goal of Tellurium. Tellurium brings together many different standards and technologies. These include not only standards such as SBML and SED–ML, but also has human–readable representations of these standards which have their own unique names (Antimony and PhraSEDML respectively). This table summarizes the principal standards, terms, and technology used in Tellurium.

multiple related files between software tools is cumbersome and error–prone, hence a single cell should be exportable as a COMBINE archive, and 3) spreading out model and simulation specifications across different cells could lead to synchronization issues if a cell is updated but not re–executed. Therefore, having a single cell type which is interconvertible with a COMBINE archive provides the best way to author and exchange modeling studies.

In order to provide a unified representation of a COMBINE archive, it is necessary to integrate many standards and software technologies. Table 2.1 shows the roles of the various standards and technologies included in Tellurium. While the components in Table 2.1 can, in principle, be used independently from Tellurium, they do not carry the same benefits for reproducibility when used in isolation. Without integration, a user would need to be familiar with the application programming interface (API) of each of the libraries in the workflow

المنارة للاستشارات
www.manaraa.com

above.

Two very important technologies indicated in Table 2.1 are Antimony and PhraSEDML. These are human–readable shorthand representations which can be converted to/from SBML and SED–ML respectively (additionally, Antimony also supports converting to/from CellML models, but elements may not necessarily have a one–to-one correspondence due to the fact that Antimony was built specifically to represent SBML). These technologies form the basis of Tellurium's *inline OMEX* representation of COMBINE archives, which allows converting an entire COMBINE archive, including all contained models and simulations, into a human–readable form. An inline OMEX cell is comprised of Antimony models enclosed in `model...end` blocks and PhraSEDML instructions in the global scope. Since Antimony models are delimited, each `model...end` block is automatically converted into an SBML model. All PhraSEDML instructions at global scope are then converted into a SED–ML file, which can contain multiple simulations and tasks. In the case where Tellurium is used to import a COMBINE archive containing multiple SED–ML files, Tellurium will add the headers `%antimony` and `%phrasedml` before each Antimony/PhraSEDML block. The headers are followed by a pathname locating the file inside the COMBINE archive. While it is not necessary to embed multiple SED–ML files in a COMBINE archive due to the fact that SED–ML can specify multiple simulations and tasks, the scheme outlined here allows Tellurium to gracefully interoperate in cases where multiple SED–ML files or a complex directory structure within the archive is desired.

An often overlooked aspect of encoding models and simulations is connecting the mathematical entities represented by the processes in the model to physical, biological entities. This can be remedied by encoding semantic content in models using the Systems Biology Ontology (SBO) [143]. SBO is a vocabulary that contains a number of identifiers for linking mathematical entities in models to specific biological entities such as enzymes and substrates. SBO can also be used to describe mathematical concepts such as Michaelis–Menten kinetics [139]. Antimony was developed before the widespread use of SBO, and so lacks native support for SBO annotations. However, support for SBO annotations is implemented as part of

www.manaraa.com

Tellurium's inline OMEX syntax. Figure 2.1 shows an example of Tellurium's SBO syntax. These SBO terms are preserved when the SBML or COMBINE archive is exported.

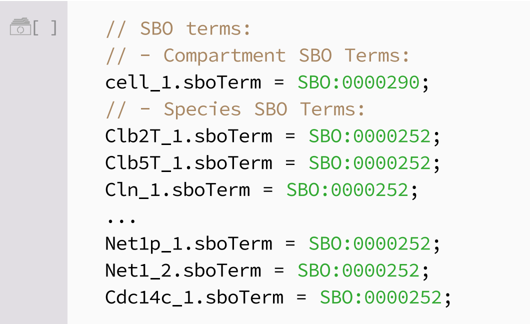
Tellurium's design is defined by three major features: providing software libraries necessary for supporting reproducibility, integration of the steps in the workflow such that the steps can be performed automatically, and providing an interface for authoring models and visualizing the output of simulations. Table 2.1 shows the respective software libraries that are used to perform these tasks.

Tellurium's notebook interface, based on the *nteract* app [148], allows authoring and editing of human–readable standards in a human–readable representation. The nteract environment is similar to Jupyter and supports Jupyter notebooks, but nteract is a complete redesign of the notebook viewer front–end using the Electron framework [2], which allows Web technologies to be used for desktop app development. Whereas Jupyter is a browser–based front–end, nteract is a desktop app, which carries several benefits for interaction with the host operating system [8]. Unlike Jupyter, nteract features: an installer and native file menus, 2) a terminal-free installation procedure, 3) integrated support for publishing notebooks to GitHub, and 4) additional user interface (UI) features, such as sticky cells and hidable cell input/output. nteract also aims for a more minimalistic user experience.

2.4 Using Tellurium to Accomplish Advanced Tasks with SED–ML

The benefits for reproducibility provided by Tellurium Notebook are demonstrated with the following case studies. the first case study shows how to encode multiple parameter sets and plot phase portraits. Explorations of parameter space are frequently done to determine if a model is applicable to conditions beyond those in the original model, an important consideration for testing model validity. The second case study evaluates if a model implementation produces results that are comparable to those in the original study via a series of tests which cover important dynamical properties of the model.

المنارة للاستشارات
www.manaraa.com



```
// SBO terms:
// - Compartment SBO Terms:
cell_1.sboTerm = SBO:0000290;
// - Species SBO Terms:
Clb2T_1.sboTerm = SBO:0000252;
Clb5T_1.sboTerm = SBO:0000252;
Cln_1.sboTerm = SBO:0000252;
...
Net1p_1.sboTerm = SBO:0000252;
Net1_2.sboTerm = SBO:0000252;
Cdc14c_1.sboTerm = SBO:0000252;
```

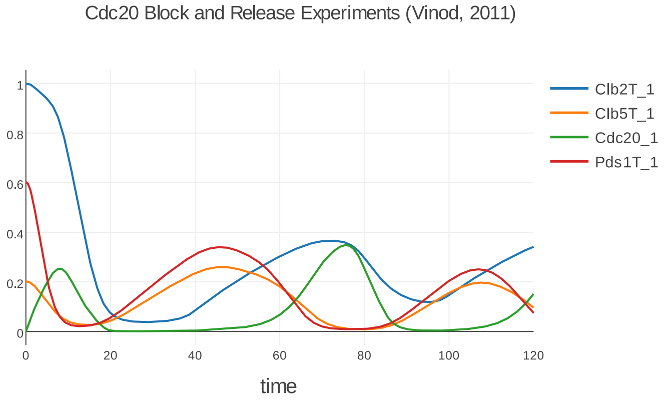


Figure 2.1: A demonstration of Tellurium's SBO syntax. This figure shows a model of mitotic exit in budding yeast [312] available via the Biomodels repository entry `BIOMD0000000370` [213]. When the SBML for this model is imported, Tellurium automatically extracts SBO identifiers for species, reactions, compartments, etc. and embeds the identifiers in the Antimony code. These identifiers point to specific physical, biological, or mathematical entities recorded in the ontology. For example, the first identifier, `SBO:0000290`, refers to a compartment in physical space [163]. Other identifiers refer to polypeptide chains (`SBO:0000252`) and protein complexes (`SBO:0000297`). These identifiers appear inline in the notebook cell and can be edited by the user. The right panel shows the transient response for this model from 0 to 120 minutes. This example is included with the Tellurium notebook viewer version 2.0.14 and later (*File*→*Open Example Notebook*→*Mitotic Exit (Vinod)*).

www.manaraa.com

2.4.1 Case study 1: Encoding post–processing and multiple parameter sets in combine archives

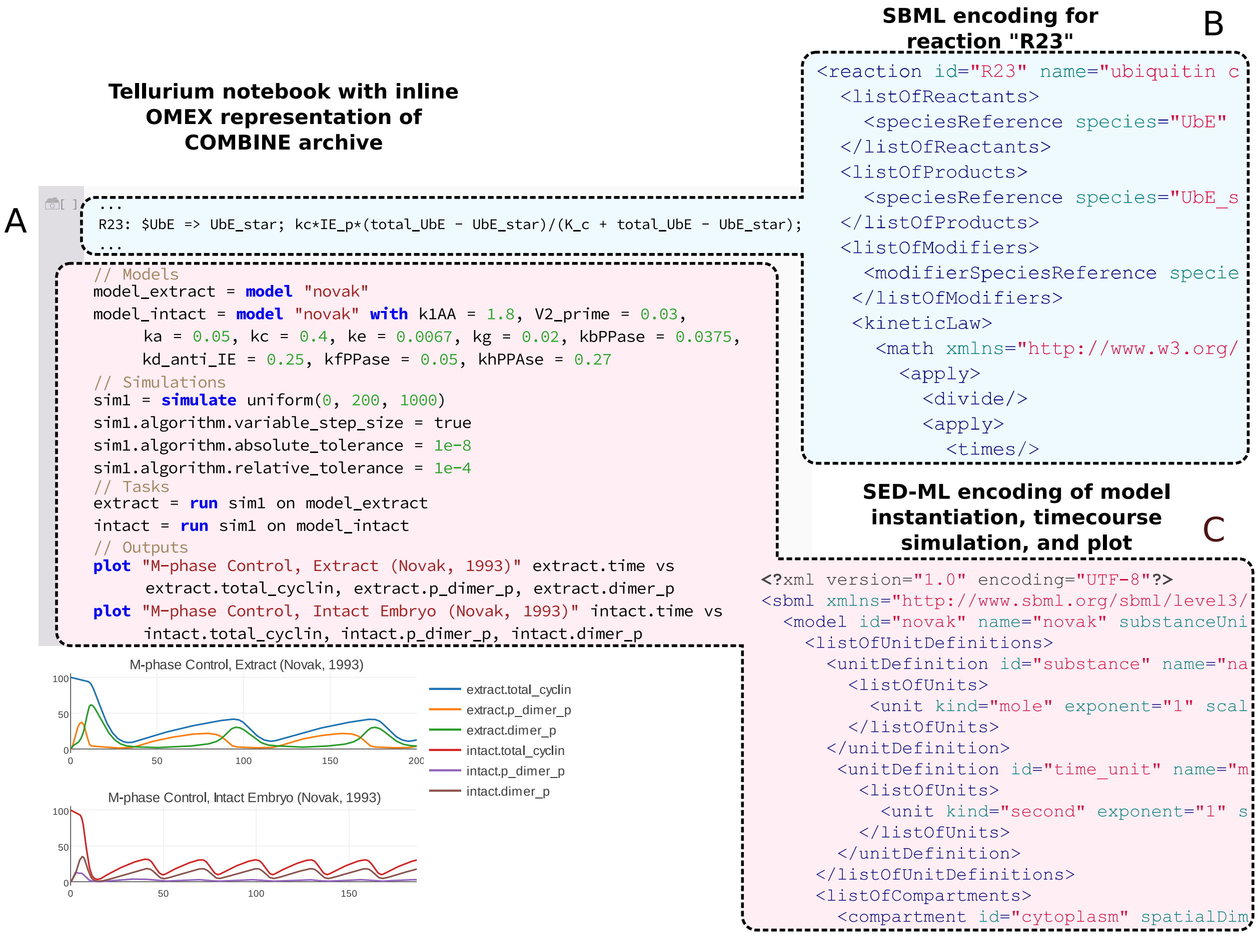
In order to meet our requirements for reproducibility, it is necessary to visualize different types of data, such as phase portraits. Furthermore, it is not sufficient to merely recreate a simulation. Rigorous reproducibility requires the ability to test existing models under a variety of circumstances (encoded as parameter sets). This first case study shows how Tellurium can be used to encode multiple parameter sets in a COMBINE archive. It also shows how COMBINE archives can be used to create phase portraits, which plot the transient value of one system variable against another.

This case study uses a model of M phase control [214]. M phase is triggered by the heterodimerization of cyclin (specifically the cyclinB in this model) and a cyclin–dependent kinase (Cdk) to form M phase promoting factor (MPF). MPF has an activating threonine phosphorylation site and two inhibitory phosphorylation sites (in this model, the two inhibitory sites are represented as a single inhibitory site) on the Cdk subunit. MPF is also regulated by the phosphatase Cdc25 (which activates MPF) and the kinase Wee1 (which inactivates MPF). MPF itself inhibits Wee1 and activaes Cdc25. Hence, it forms a pair of positive feedback loops with its own regulators. The model contains 21 reactions and is available via the BioModels repository (`BIOMD0000000107` [80]).

Figure 2.2 shows a comparison of Tellurium's human–readable representation of the M phase control model and simulation versus the standard–encoded representations. Clearly, readability is essential for model transparency. However, readability is essential for model reuse as well. To demonstrate this this SBML–only model is converted into a COMBINE archive containing both SBML portions describing the model and SED–ML portions describing the simulation. Tellurium's human–readable format permits easy modification of the published model and simulations contained in the COMBINE archive.

In order to create a SED–ML specification for this model, four steps in the workflow (corresponding to distinct elements in SED–ML) must be defined: (1) model definition, (2)

www.manaraa.com



Tellurium notebook with inline OMEX representation of COMBINE archive

```
...
R23: $UbE => UbE_star; kc*IE_p*(total_UbE - UbE_star)/(K_c + total_UbE - UbE_star);
...
```

```
// Models
model_extract = model "novak"
model_intact = model "novak" with k1AA = 1.8, V2_prime = 0.03,
     ka = 0.05, kc = 0.4, ke = 0.0067, kg = 0.02, kbPPase = 0.0375,
     kd_anti_IE = 0.25, kfPPase = 0.05, khPPAse = 0.27
// Simulations
sim1 = simulate uniform(0, 200, 1000)
sim1.algorithm.variable_step_size = true
sim1.algorithm.absolute_tolerance = 1e-8
sim1.algorithm.relative_tolerance = 1e-4
// Tasks
extract = run sim1 on model_extract
intact = run sim1 on model_intact
// Outputs
plot "M-phase Control, Extract (Novak, 1993)" extract.time vs
     extract.total_cyclin, extract.p_dimer_p, extract.dimer_p
plot "M-phase Control, Intact Embryo (Novak, 1993)" intact.time vs
     intact.total_cyclin, intact.p_dimer_p, intact.dimer_p
```

SBML encoding for reaction "R23"

```
<reaction id="R23" name="ubiquitin c
  <listOfReactants>
    <speciesReference species="UbE"
  </listOfReactants>
  <listOfProducts>
    <speciesReference species="UbE_s
  </listOfProducts>
  <listOfModifiers>
    <modifierSpeciesReference specie
   </listOfModifiers>
   <kineticLaw>
     <math xmlns="http://www.w3.org/
       <apply>
         <divide/>
         <apply>
           <times/>
```

SED-ML encoding of model instantiation, timecourse simulation, and plot

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/
  <model id="novak" name="novak" substanceUni
    <listOfUnitDefinitions>
      <unitDefinition id="substance" name="na
        <listOfUnits>
          <unit kind="mole" exponent="1" scal
        </listOfUnits>
      </unitDefinition>
       <unitDefinition id="time_unit" name="m
         <listOfUnits>
           <unit kind="second" exponent="1" s
         </listOfUnits>
       </unitDefinition>
     </listOfUnitDefinitions>
     <listOfCompartments>
       <compartment id="cytoplasm" spatialDim
```

Figure 2.2: A comparison of Tellurium's human–readable representation of a COMBINE archive shown in a Tellurium notebook (A) and excerpts from the equivalent SBML (B) and SED–ML (C) encodings. Tellurium's in-line OMEX format contains human–readable representations of both SBML and SED–ML (A). Here, SBML is represented by Antimony code (with the definition of a single reaction in blue) and PhraSEDML (in red). (B) shows the SBML encoding for a single reaction. The single–line human–readable form of this reaction is highlighted in part (A) for comparison. The components of the Antimony syntax are as follows: `R23` is the reaction label, the reactant `$UbE`, with a dollar sign indicating a boundary species, a `=>` symbol, which indicates an irreversible reaction (reversible reactions can be indicated with `->`), the product `UbE_star`, and the kinetic law comprised of everything following the semicolon. Using the SBML encoding, it is difficult to modify the reaction stoichiometry or kinetic law, whereas this task is easy in Tellurium. Finally, (C) shows the SED–ML encoding corresponding to the human–readable simulation portion of this COMBINE archive. The simulation portion performs the following functions: first, two SBML models are instantiated with different sets of parameters (in the original publicatuion [214], the authors provided one set of parameters for oocyte extract and a different set for intact embryos). Second, a timecourse simulation with an adaptive step size is attained by setting the `variable_step_size` property of the simulation as well as appropriate tolerance values. Finally, the simulation is run with the two different model instantiations and plotted with representative state variables (active MPF, doubly phosphorylated/inactive MPF, and total cyclin) to show the behavior of the two parameter sets.

المنارة للاستشارات
www.manaraa.com

simulation, (3) task specification, and (4) output generation. For (1), models can be defined in Tellurium's human–readable format by referencing SBML or CellML files in the same COMBINE archive, with the option of including parameter replacements. For (2), SED–ML simulations can be either timecourse simulations or steady state computation, and can reference a specific algorithm (e.g., LSODA), or a generic implementation using KiSAO [73]. Tellurium uses predefined keywords such as `lsoda` (an ODE solver implementation [232]) to refer to popular implementations. In SED–ML, simulations are specified independently from models. This allows model and simulation elements to be reused in different combinations. For (3), SED–ML uses task elements to describe these combinations. Finally, the output elements of (4) can be plots or reports and allow users to access the output of tasks. Tellurium's human–readable format allows defining a SED–ML model by instantiating the same SBML model with different parameter values (m in this example) using the syntax:

```
mymodel = model "novak" with param1=value2, param2=value2 ...
```

with the param/value pairs being replaced by the corresponding parameter ids and values respectively. This syntax is used to instantiate two copies of the model, one with parameter values for extract and another with parameter values for intact embryos [214]. Finally, Tellurium can be used to encode integrator tolerances and encode an adaptive step–size simulation in SED–ML. Figure 2.3 shows simulation results for both parameter sets.

This case study shows that Tellurium provides an efficient means of converting SBML models into exchangeable COMBINE archives containing simulation components. Furthermore, COMBINE archives can contain important dynamical information about the model, such as the behavior under the different parameter sets that were explored in this study.

2.4.2 Case study 2: Reproducibility through in–depth variational studies

Reproducibility requires that a model implementation produces results consistent with the original study, especially if a different authoring tool is used. In order to provide criteria

المنارة للاستشارات
www.manaraa.com

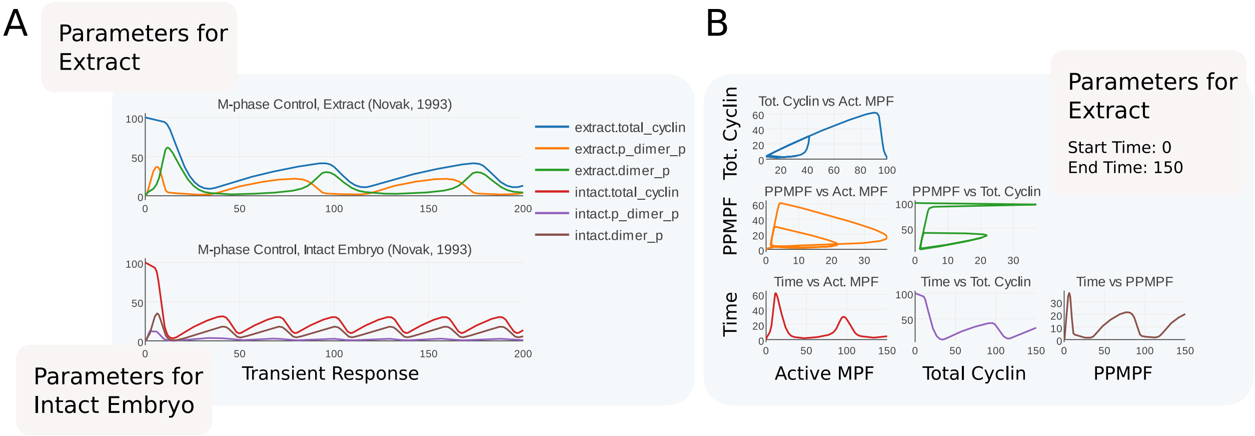


Figure 2.3: A demo of an SBML/SED–ML encoding contained in a COMBINE archive showing two useful features of the encoding: Multiple parameter sets (A) and post–processing (B). (A) Transient responses of M phase control [214]. This model was published with two parameter sets. One set is based on measurements from *Xenopus* oocyte extracts (top) whereas the other is based on measuremetns from intact embryos (bottom). (B) Phase portraits of representative state variables in the model. These variables are chosen after [214] and are as follows: total cyclin, doubly–phosphorylated MPF (PPMPF, the predominant inactive form of MPF [214]), active MPF, and time. Each pair of variables is plotted in this matrix. Y–axis variables are indicated in the rows of the plot and x–axis variables are indicated in the columns. The title of each subplot is given in terms of x vs y, e.g., the top left subplot shows total cyclin on the x–axis vs active MPF on the y–axis. Phase portraits can show transients (such as the initial response of total cyclin in the upper left corner in blue, which starts at 100 and decreases to its normal range) as well as limit cycles (exhibited by all three phase portraits in the upper part of (B)). The slope of a given region of the phase portrait is useful for showing the relative rate of change of two quantities. The green and orange curves show regions where one quantity changes rapidly with respect to another. These regions correspond to the rapid rise in active MPF due to positive feedback from MPF to its own self–activation, and the subsequent falloff of total cyclin due to cyclin degradation via a ubiquitin pathway activated by MPF. The plot in part (B) is derived from the parameter set for oocyte extract, corresponding to the top plot in part (A).

المنارة للاستشارات
www.manaraa.com

for judging whether a model reproduction is consistent with the original, a set of testing criteria is required, similar to the concept of unit testing in software. However, researchers seldomly perform extensive checks on the dynamics of models before using them. This is due in part to the lack of tool support for easily modifying and producing variants of models and simulations encoded in exchangeable formats. Tellurium's authoring features enable modelers to encode dynamical unit tests in COMBINE archives, thereby providing a way to verify that a model has been correctly reproduced.

This case study reproduces a highly–detailed model of syncytial nuclear divisions in the *Drosophila* embryo [60] through testing the model's dynamics under different conditions. This model is biologically similar to the model in Case Study 1 [214] but is more detailed (54 vs. 23 reactions) and is available as a pre–encoded COMBINE archive [266]. In many insect species, the embryo enters a period of rapid mitotic division without cytokinesis [107] immediately following fertilization. In *Drosophila*, 13 of these divisions occur within 3 hours of fertilization [60]. MPF is again the main regulator of these divisions. Recalling that MPF is a heterodimer of cyclin and Cdk, cyclin subunits tend to be the limiting factor in complex formation, and are thought to regulate mitotic division. Cyclin availability is controlled by the anaphase promoting complex (APC), which targets CycB for degradation. However, in *Drosophila*, the levels of CycB appear to remain high during the first 8 mitotic divisions [87]. This observation can be reconciled with known mechanisms by assuming that CycB degradation only occurs in the vicinity of the mitotic spindle [60, 129, 242], despite the absence of a nuclear envelope during the mitotic divisions. To account for this hypothetical local degradation of CycB, the model artificially separates the cytoplasm into two "compartments," with a cytoplasmic compartment representing the cell and a nuclear compartment representing the volume in the vicinity of the mitotic spindle.

The COMBINE archive encoding of this model by Scharm and Tourè [266] is used as a starting point. This archive contains SBML derived from biomodel `BIOMD0000000144`[5],

[5] http://identifiers.org/biomodels.db/BIOMD0000000144

المنارة للاستشارات
www.manaraa.com

which is intended to reproduce Figure 1 of [60]. However, the archive does not contain more extensive tests of the model's dynamics, such as whether the model can be used to reproduce several other simulations described in the paper. The initial variant encoded by the COMBINE archive and shown in Figure 2.4B and C is based on a model with a constant level of the phosphatase String (which corresponds to a Cdc25 homolog in Drosophila), whereas in reality String levels change over the course of the mitotic cycles. String regulates MPF via a positive feedback loop, and has been shown to peak at the seventh or eighth cycle of the mitotic divisions [60]. To account for this, Calzone et al. [60] posited that String mRNA is degraded by a hypothetical factor "X," causing the synthesis rate of String to drop over time. Therefore, the SED–ML of the original COMBINE archive [266] was modified as follows to include the synthesis and degradation of String. These modifications to the original COMBINE archive allow the reproduction of Figure 3 of [60] by making the following changes:

- Enable synthesis and degradation of String by setting the parameters `ksstg=0.02` and `kdstg=0.015` respectively.

- Set the initial concentration of total String to zero by setting `StgPc=0`.

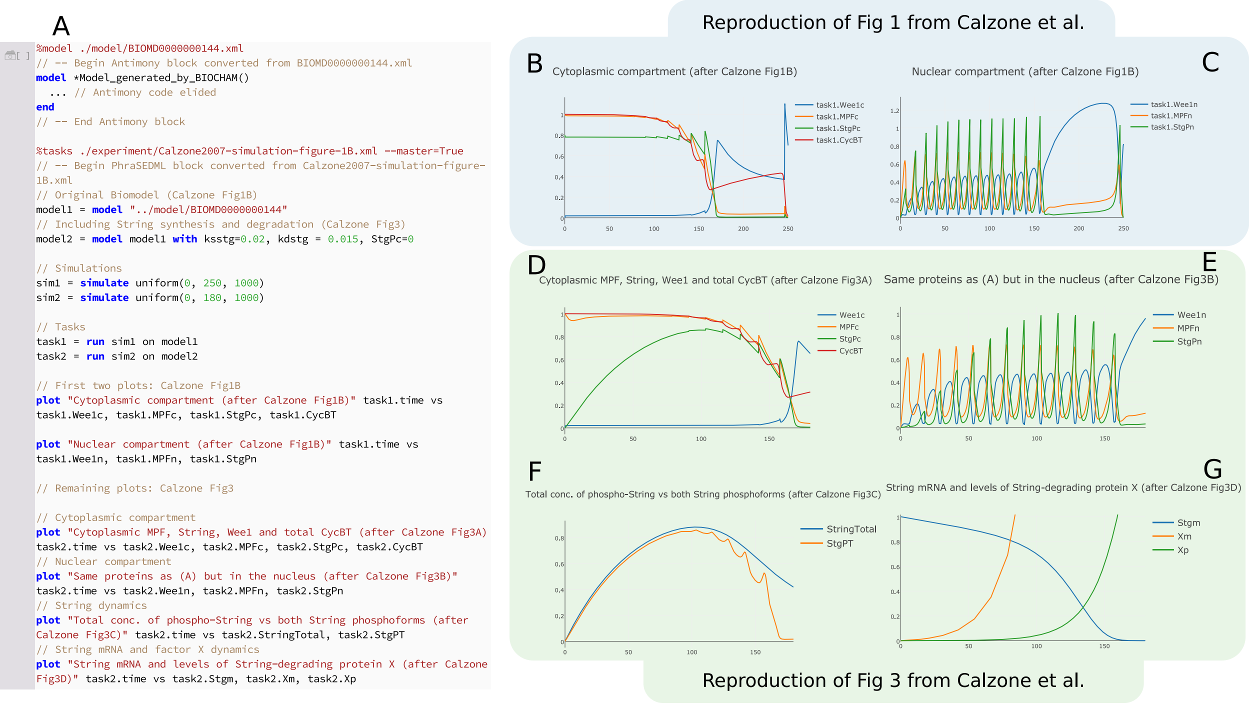
- Compute the total amount of unphosphorylated String by adding the rule `StgT := (1 - N*E_1)*Stgc + N*E_1*Stgn`.

- Compute the total amount of String in the cell by adding the rule `StringTotal := StgPT + StgT`.

Tellurium makes it easy to encode both the original variant, without String synthesis and degradation, and the variant including these terms in a COMBINE archive [17]. Figure 2.4 shows the results of executing this COMBINE archive in Tellurium, and Figure 2.5 shows logarithmic plots of the transient response. The dynamical test cases for this model have thus

المنارة للاستشارات

www.manaraa.com



```
%model ./model/BIOMD0000000144.xml
// -- Begin Antimony block converted from BIOMD0000000144.xml
model *Model_generated_by_BIOCHAM()
  ... // Antimony code elided
end
// -- End Antimony block

%tasks ./experiment/Calzone2007-simulation-figure-1B.xml --master=True
// -- Begin PhraSEDML block converted from Calzone2007-simulation-figure-
1B.xml
// Original Biomodel (Calzone Fig1B)
model1 = model "../model/BIOMD0000000144"
// Including String synthesis and degradation (Calzone Fig3)
model2 = model model1 with ksstg=0.02, kdstg = 0.015, StgPc=0

// Simulations
sim1 = simulate uniform(0, 250, 1000)
sim2 = simulate uniform(0, 180, 1000)

// Tasks
task1 = run sim1 on model1
task2 = run sim2 on model2

// First two plots: Calzone Fig1B
plot "Cytoplasmic compartment (after Calzone Fig1B)" task1.time vs
task1.Wee1c, task1.MPFc, task1.StgPc, task1.CycBT

plot "Nuclear compartment (after Calzone Fig1B)" task1.time vs
task1.Wee1n, task1.MPFn, task1.StgPn

// Remaining plots: Calzone Fig3

// Cytoplasmic compartment
plot "Cytoplasmic MPF, String, Wee1 and total CycBT (after Calzone Fig3A)
task2.time vs task2.Wee1c, task2.MPFc, task2.StgPc, task2.CycBT
// Nuclear compartment
plot "Same proteins as (A) but in the nucleus (after Calzone Fig3B)"
task2.time vs task2.Wee1n, task2.MPFn, task2.StgPn
// String dynamics
plot "Total conc. of phospho-String vs both String phosphoforms (after
Calzone Fig3C)" task2.time vs task2.StringTotal, task2.StgPT
// String mRNA and factor X dynamics
plot "String mRNA and levels of String-degrading protein X (after Calzone
Fig3D)" task2.time vs task2.Stgm, task2.Xm, task2.Xp
```

Figure 2.4: Using Tellurium to reproduce model variants in [60] and repackage as a COMBINE archive. To demonstrate the use of COMBINE archives for encoding model variants, we began with a COMBINE archive describing a single variant of this model without String synthesis or degradation [266], which reproduces Figure 1B of [60] (plots B and C here). A variant was then created in Tellurium describing String degradation, which reproduces Figure 3 of [60] (plots D through G here). Panel (A) shows the inline OMEX cell with the Antimony code elided (it would belong in the `model Model_generated_by_BIOCHAM...end` block, where the ellipsis is currently shown). Everything after the `end` instruction is thus PhraSEDML. Plots B and D show the transient response of the cytoplasmic compartment of the model. Plots C and E show the nuclear compartment (defined as the spatial region around the mitotic spindle). Plot F shows the levels of total String and its phosphorylated state. Plot G shows the level of String mRNA and protein factor X, which degrades String mRNA. Note the y–axis scale on plot G was manually adjusted to show the mRNA dynamics. The subplots in this figure intentionally have different durations, after Calzone *et al.* [60]. The model in [60] was authored using BIOCHAM [59]. Our model reproductions that reproduce these plots are available as a COMBINE archive [17].

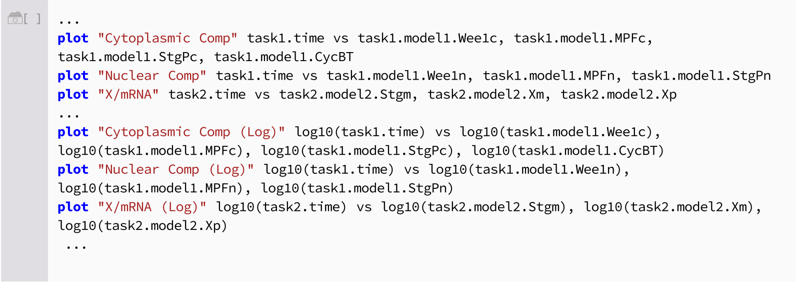
المنارة للاستشارات
www.manaraa.com

been expanded to reproduce two simulations from two different variants described by the original authors (Figure 1 and 3 of [60]), enabling better coverage of the model's dynamics.

In order to gain insight into the regulatory mechanism controlling the mitotic divisions, and understand the transitions that control the exact number of these divisions, Calzone et al. performed a one–parameter bifurcation analysis [60]. Bifurcation analysis probes the number and position of steady states and other types of attractors as a function of a parameter. The oscillations shown in Figure 2.4 are the result of discrete division events, and the behavior shown does not represent a limit cycle. However, the model can be shown to exhibit limit cycle behavior by 1) removing all discrete events and 2) fixing the number of divisions by introducing the variable C as a cycle counter. The number of nuclear compartments is then given by $N = 1.95^C$ (1.95 is a scaling factor described in [60]). For a given cycle number C, MPF exhibits limit cycle oscillations, although the amplitude and period of these oscillations changes with the cycle number. At low cycle number, Calzone et al. observed that these oscillations were dominated by the negative feedback effect of cyclin degradation, whereas for large cycle number ($C \geq 12$), positive feedback via control of phosphorylated MPF by the kinase Wee1 and phosphatase String contributes to the oscillations.

SED–ML does not support bifurcation analysis, precluding us from reproducing that part of the study in an exchangeable format. However, it is still possible to test the change in regulatory shift from negative to positive feedback. Instead of a bifurcation diagram, the limit cycle behavior of the original model is compared with a model variant with reduced Wee1 and String activation and deactivation rates. This slows the timescale of the positive feedback component of the model. Figure 2.6 compares the behavior of the original model at early and late cycle numbers with the variant containing attenuated positive feedback. Whereas the normal model exhibits stable limit cycle oscillations at both $C = 1$ and $C = 12$, the oscillations in the attenuated model are transient at late cycles ($C = 12$) but not at early cycles ($C = 1$). This observation suggests that String and Wee1 dynamics are indeed crucially important for late cycle oscillations, but not for early cycle oscillations, confirming the shift in regulatory mechanism. These simulations thus form a third set of unit tests for

www.manaraa.com



```
...
plot "Cytoplasmic Comp" task1.time vs task1.model1.Wee1c, task1.model1.MPFc,
task1.model1.StgPc, task1.model1.CycBT
plot "Nuclear Comp" task1.time vs task1.model1.Wee1n, task1.model1.MPFn, task1.model1.StgPn
plot "X/mRNA" task2.time vs task2.model2.Stgm, task2.model2.Xm, task2.model2.Xp
...
plot "Cytoplasmic Comp (Log)" log10(task1.time) vs log10(task1.model1.Wee1c),
log10(task1.model1.MPFc), log10(task1.model1.StgPc), log10(task1.model1.CycBT)
plot "Nuclear Comp (Log)" log10(task1.time) vs log10(task1.model1.Wee1n),
log10(task1.model1.MPFn), log10(task1.model1.StgPn)
plot "X/mRNA (Log)" log10(task2.time) vs log10(task2.model2.Stgm), log10(task2.model2.Xm),
log10(task2.model2.Xp)
 ...
```

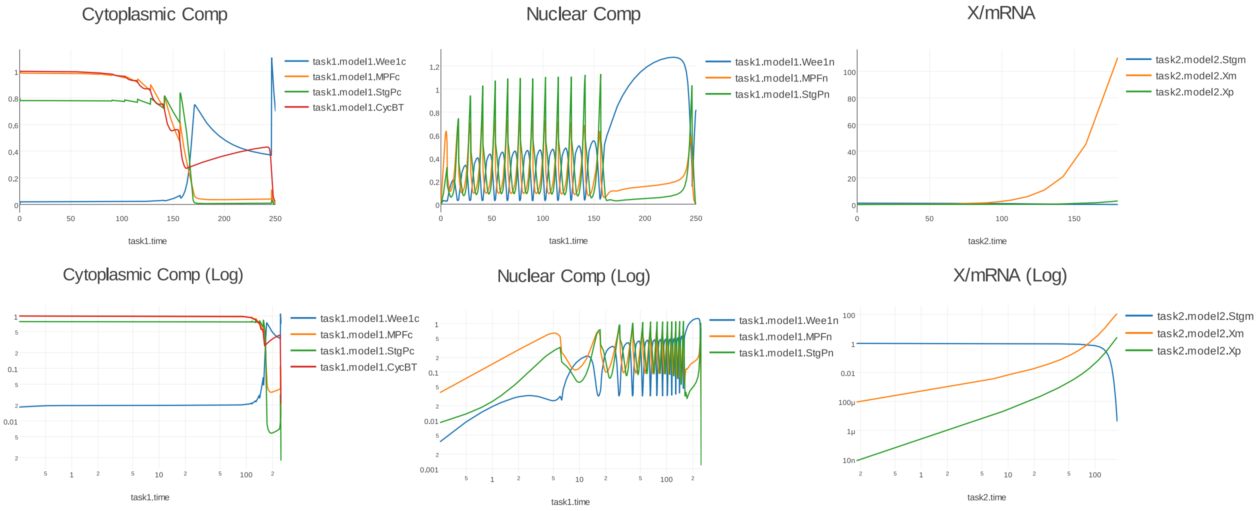


Figure 2.5: Comparison of logarithmic (bottom row of plots) vs. linear (top row of plots) plotting of the transient response in Figure 2.4B, C, and G, along with an excerpt of the relevant PhraSEDML code for plotting on logarithmic axes. Figure 2.4G, which is a plot of String mRNA and hypothetical factor X, which degrades String mRNA, exhibits a large dynamic range. Logarithmic plotting helps visualize the dynamic range of these quantities. This is achieved in PhraSEDML by wrapping the quantities for x and y axes inside a `log10` operation.

www.manaraa.com

the model, encoded as a COMBINE archive [19].

This case study shows that, using Tellurium's editing capabilities, it is possible to create an extensive set of unit tests for dynamical behavior of this model, which can be exported as a COMBINE archive and imported into another tool as shown in Figure S1. Creating these tests required a means of quickly editing and expanding upon both the SBML and SED–ML embedded in the COMBINE archive. Tellurium's notebook approach allows us to satisfy these requirements, and provides an integrated workflow for testing the dynamical behavior of the model.

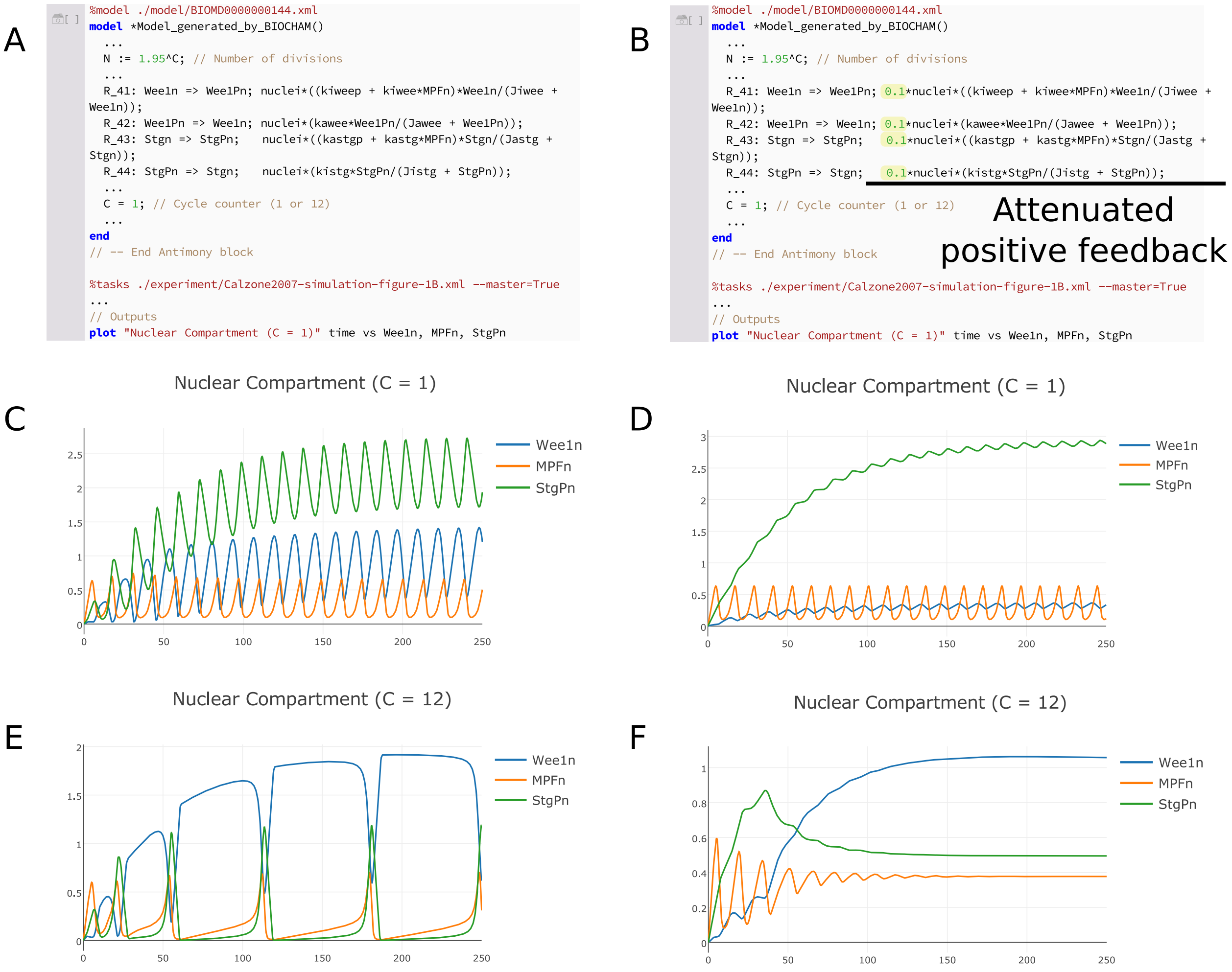
2.4.3 Interoperability & Test Cases

In order to achieve the exchangeability requirement of reproducibility, broad standards compliance is necessary. A small number of test cases, such as the first two case studies, is not sufficient to ensure interoperability with other software. A number of COMBINE archive exemplars from literature, other software tools, and our own archives were used during Tellurium's development. These archives are provided as a resource to other developers and they are publicly available online. The test archives are structured to separate examples with advanced SED–ML features from those with basic SED–ML usage, enabling tool developers to implement incremental support for the standard. S1 Table lists all test archives and how to obtain them.

2.4.4 Advanced SED–ML Support

In order to address the requirement of broad standards compliance, Tellurium was tested against a set of tests provided by the SED–ML Web Tools [38]. These tests utilize advanced features of the SED–ML standard, and are designed to demonstrate the standard's coverage of different types of analysis. S2 Table lists all files used in this test set, and S4 Figure shows the results of exporting these files to Tellurium and back again.

المنارة للاستشارات
www.manaraa.com



```
%model ./model/BIOMD0000000144.xml
model *Model_generated_by_BIOCHAM()
  ...
  N := 1.95^C; // Number of divisions
  ...
  R_41: Wee1n => Wee1Pn; nuclei*((kiweep + kiwee*MPFn)*Wee1n/(Jiwee +
Wee1n));
  R_42: Wee1Pn => Wee1n; nuclei*(kawee*Wee1Pn/(Jawee + Wee1Pn));
  R_43: Stgn => StgPn;   nuclei*((kastgp + kastg*MPFn)*Stgn/(Jastg +
Stgn));
  R_44: StgPn => Stgn;   nuclei*(kistg*StgPn/(Jistg + StgPn));
  ...
  C = 1; // Cycle counter (1 or 12)
  ...
end
// -- End Antimony block

%tasks ./experiment/Calzone2007-simulation-figure-1B.xml --master=True
...
// Outputs
plot "Nuclear Compartment (C = 1)" time vs Wee1n, MPFn, StgPn
```

```
%model ./model/BIOMD0000000144.xml
model *Model_generated_by_BIOCHAM()
  ...
  N := 1.95^C; // Number of divisions
  ...
  R_41: Wee1n => Wee1Pn; 0.1*nuclei*((kiweep + kiwee*MPFn)*Wee1n/(Jiwee +
Wee1n));
  R_42: Wee1Pn => Wee1n; 0.1*nuclei*(kawee*Wee1Pn/(Jawee + Wee1Pn));
  R_43: Stgn => StgPn;    0.1*nuclei*((kastgp + kastg*MPFn)*Stgn/(Jastg +
Stgn));
  R_44: StgPn => Stgn;    0.1*nuclei*(kistg*StgPn/(Jistg + StgPn));
  ...
  C = 1; // Cycle counter (1 or 12)
  ...
end
// -- End Antimony block

%tasks ./experiment/Calzone2007-simulation-figure-1B.xml --master=True
...
// Outputs
plot "Nuclear Compartment (C = 1)" time vs Wee1n, MPFn, StgPn
```

Attenuated positive feedback

Figure 2.6: Testing the shift in regulatory mechanism of mitotic oscillations. To verify the observation [60] that the number of mitotic divisions in the *Drosophila* embryo is governed by a shift from negative to positive feedback, all discrete events were removed and the variable C was introduced such that $N = 1.95^C$. The limit cycles produced by this eventless model (left) were compared with those produced by a variant with attenuated positive feedback from the regulators Wee1 and String (right). Attenuation was achieved by decreasing the rates of the phosphorylation and dephosphorylation of Wee1 and String. The original model exhibits stable limit cycle oscillations for both early cycles (C), which are putatively dominated by negative feedback, and late cycles (E), which are putatively dominated by positive feedback. The attenuated model only exhibits stable oscillations at early cycles (D), suggesting that positive feedback does indeed play a role in late cycle oscillations (F). Our model reuse and modification study is available as a COMBINE archive that reproduces the figure shown and facilitates further modification and reuse [19].

المنارة للاستشارات
www.manaraa.com

2.4.5 SBML Test Suite COMBINE Archives

The SBML Test Suite [18] is a collection of dynamical models along with expected trajectories designed to test software tools for compliance with the SBML standard. Each test case contains a SBML model, simulation parameters encoded in SED–ML, expected trajectories encoded as a comma–separated values (CSV) file, and graphical plots for reference. Each of these 1196 test cases was converted into COMBINE archives containing the SBML models, SED–ML simulations, and CSV expected results and used these COMBINE archives as a benchmark for Tellurium's support for standards. The results of this benchmark are shown in S3 Table.

2.5 Enabling Web–Based SBML Tools

> The contents of this section are largely based on a manuscript in preparation for submission to *Biosystems*. The curriculum vitae at the end of this thesis lists all manuscripts and resp. statuses.

The SBML [131] standard is used for encoding reaction network models in systems biology research in a reusable, exchangeable, and future–proof manner. One of the factors behind SBML's wide adoption is the SBML standard's process for introducing extension modules, which allow incremental incorporation of new capabilities. While the core components of the standard are designed for describing kinetic chemical reaction network models, SBML extensions exist for encoding constraint–based models (the "flux–balance constraints" extension, employed by the widely used COBRA framework for constraint–based modeling [33, 269]), and rule–based models (the SBML "multi" extension [330]). SBML is used in several online model repositories including BioModels [164, 168] and JWS Online [219, 229], which host primarily kinetic reaction network models, and BiGG Models [154], which hosts primarily genome–scale constraint–based models.

Despite this wide–spread adoption and inclusion in several online repositories, no feature–complete JavaScript library currently exists that can run in a web browser (a native Node.js

المنارة للاستشارات
www.manaraa.com

module exists, but cannot run in the browser). Thus, these online repositories must rely on server–side processing of all SBML–related requests. A JavaScript library would allow these services to offload some of their processing to the client, and would also allow for more interactive features on the Web. Furthermore, the Web is becoming a major platform for systems biology tools. With the advent of Web applications for pathway visualization (Escher [153]), gene interaction network visualization (Cytoscape.js [98]), expression analysis (ZBIT [250]) and integrated design systems (Caffeine [175]), the need for a JavaScript library which can read and write SBML becomes imperative.

The first aim of this thesis is to enable reproducible modeling through standards. Providing standards support via a web–based library helps fill a critical gap in current approaches to online model development. This section presents libsbmljs, a feature–complete JavaScript library for reading and writing SBML in the browser and Node.js. libsbmljs uses the full codebase of the libSBML C++ library compiled to the web using Emscripten, a toolset for compiling C++ projects to the web. Emscripten emits WebAssembly [15], a W3C standard for running platform–independent binary code on the web that is supported on all major browsers. libsbmljs provides a JavaScript wrapper around this binary format that allows the library to be used like a normal JavaScript package. The wrapper supports *all* SBML Level 3 extensions, meaning it can read and write any type of SBML content. Since our library runs in the browser, it does not require a dedicated web server. This is an important consideration for academic software, where long–term maintenance cost is a concern.

2.6 Web Compilation of libSBML

Prior work on implementing the SBML standard has resulted in two libraries: libSBML [48], a C++ library with interfaces for many languages, and JSBML [165, 82], a platform–independent pure Java library. While the existence of these separate implementations is certainly a convenience for C++ and Java developers respectively, it necessitates the maintenance of two independent libraries. Rather than attempt to create a third implementation in pure JavaScript, libsbmljs is a web–capable interface for the libSBML C++ library created

المنارة للاستشارات
www.manaraa.com

using Emscripten [328], a C++–to–JavaScript compiler. Despite its C++ origins, libsbmljs is completely platform independent and runs on modern browsers on any device which supports web standards.

Compiling a C++ library with Emscripten does not produce a ready–to–use JavaScript library automatically. Instead, Emscripten compiles to WebAssembly [252], a low–level binary format similar to x86 machine code but with additional features for security and platform–independence. Since WebAssembly is very low level, it is difficult to use to design JavaScript web applications. Instead, Emscripten can be used to also compile a JavaScript interface that abstracts the low–level details of calling into WebAssembly and instead allows developers to use familiar JavaScript objects and methods. However, this interface is not generated automatically by Emscripten. Instead, it must be manually specified using WebIDL.

WebIDL is a World Wide Web Consortium (W3C®) standard that specifies *interfaces* to EMCAScript (i.e. JavaScript) objects. For example, the libSBML C++ class `SBase` has the method `getId()`, which returns a string. In WebIDL, this would be specified as:

```
/**
 * SBase: the base class of
 * most SBML elements
 */
[Prefix="libsbml::"]
interface SBase {
  /**
   * Returns this element's
   * id attribute
   */
  DOMString getId();
};
```

In the example above, the body of the `getId` method is intentionally left blank because it

www.manaraa.com

will delegate to the corresponding WebAssembly routine. Using syntax similar to the above, WebIDL interface files were manually designed for every libSBML class and method. However, one issue remains with this approach. The comments entered into the IDL definition above will not appear in the JavaScript interface generated by Emscripten. Thus, there is no way of adding documentation to the generated JavaScript code, which defeats any attempt to generate API documentation. To remedy this issue, a script is used to automatically extract documentation strings from IDL files and insert them into the generated JavaScript code. This allowed us to generate extensive API documentation using `documentationjs`, a documentation generator for JavaScript.

2.6.1 Special Considerations for Usage

Emscripten–generated WebAssembly/JavaScript libraries are supported on a wide variety of browsers and devices (https://github.com/libsbmljs/libsbmljs lists the browsers that have been tested). However, there are minor differences between these libraries and regular JavaScript libraries, which are described below.

Asynchronous Loading

Emscripten–generated libraries load asynchronously. In other words, the library cannot be used immediately as soon as the web page has loaded. This is due to the fact that Emscripten–generated libraries consist of both a JavaScript source file (.js) containing JavaScript classes and methods, and a WebAssembly file (.wasm) containing the compiled C++ code. The browser may load the JavaScript source file before completely loading and compiling the WebAssembly file. In order to accommodate this, Emscripten libraries provide a ‘then()‘ method for the JavaScript module object similar to a JavaScript Promise. This method accepts a callback that will execute once the WebAssembly is fully downloaded and compiled.

www.manaraa.com

Manual Memory Management

Most modern languages feature some type of automatic garbage collection. However, WebAssembly is a low–level binary–like format, and hence does not provide high–level features like garbage collection. This means that whenever the user creates an object in libsbmljs using the `new` keyword, the user must also destroy the object using `libsbml.destroy(obj)`. In most cases, this simply amounts to destroying the SBML document when it is no longer needed.

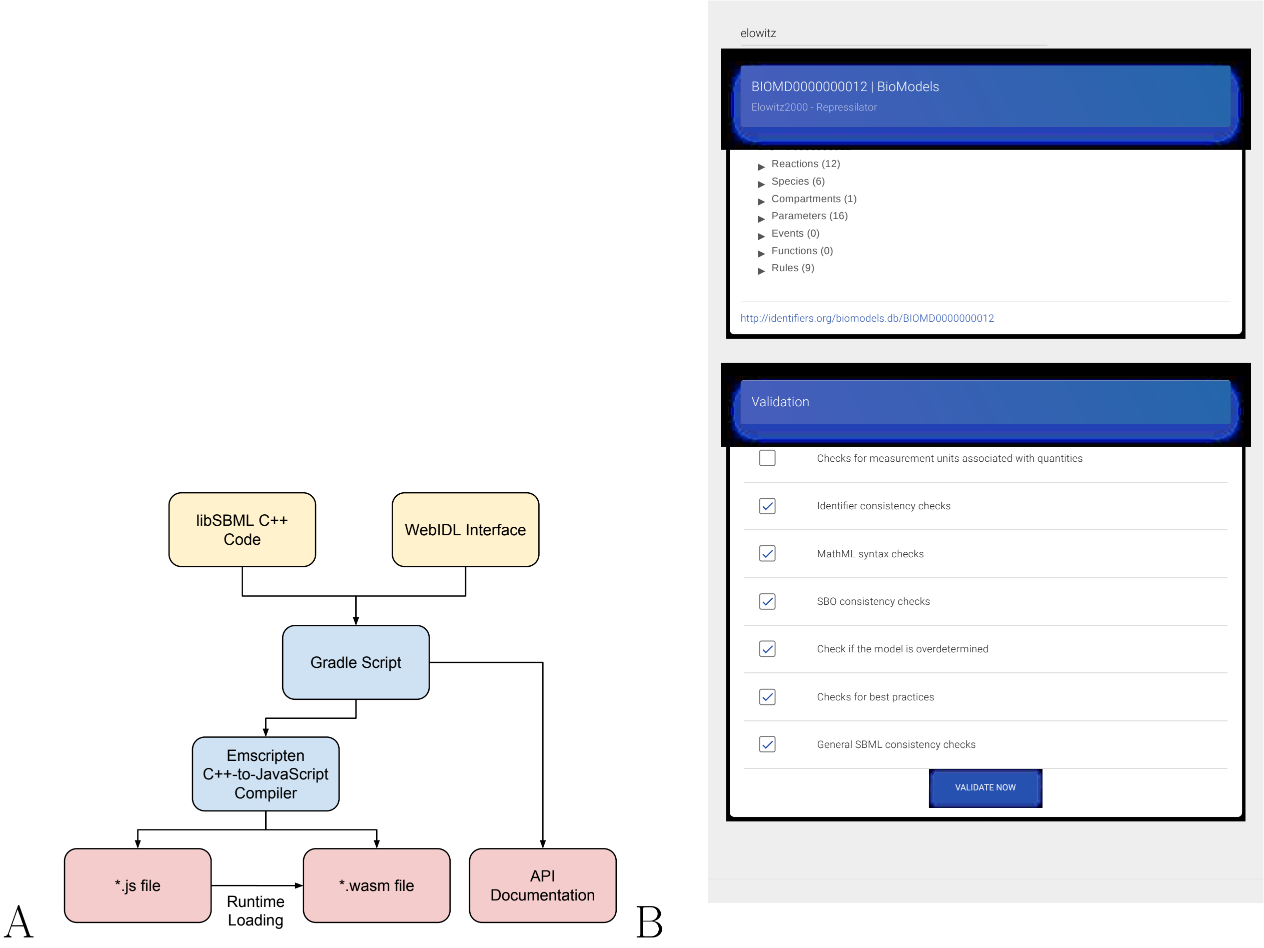
In terms of modern programming languages, this may seem like a significant regression, but it is an unavoidable tradeoff when using C++ compiled WebAssembly, at least for currently available technology (a proposal exists to add garbage collection to WebAssembly [16], but an implementation is not available at the time of writing). In the event that the user forgets to call the `libsbml.destroy` function, the allocated object will persist in the browser's memory until the browser tab is closed. Since our main target users are developers of web applications, and browser tabs are short–lived, this is a significant concern. However, Node.js developers should take care to destroy all created objects. The same requirement also applies to libSBML's native Node.js module.

2.7 Discussion

2.7.1 Tellurium's Approach to Standards Support

In order for the conclusions of a research study to be valid, the models used in the study must be reliable. Using SED–ML to reproduce the dynamics of a model and compare these dynamics with expected values adds crucial value to the integrity and validity of studies that reuse or expand on the model. As an exchangeable format, SED–ML is confined to the intersection of the most common features available in dynamical modeling tools, which leaves out certain useful types of analysis (e.g., bifurcation analysis). However, the use case of SED–ML is not to serve as a replacement for current analysis methods. Instead, SED–ML is a tool to test the dynamical behavior of models before using them. For example,

المنارة للاستشارات
www.manaraa.com



elowitz

BIOMD0000000012 | BioModels

Elowitz2000 - Repressilator

- Reactions (12)
- Species (6)
- Compartments (1)
- Parameters (16)
- Events (0)
- Functions (0)
- Rules (9)

http://identifiers.org/biomodels.db/BIOMD0000000012

Validation

□ Checks for measurement units associated with quantities

☑ Identifier consistency checks

☑ MathML syntax checks

☑ SBO consistency checks

☑ Check if the model is overdetermined

☑ Checks for best practices

☑ General SBML consistency checks

Figure 2.7: **(A)** A workflow diagram of the process used to produce libsbmljs. The libSBML C++ source code and a hand–written WebIDL interface are processed by a Gradle script to produce Emscripten–compiled bytecode and JavaScript API documentation. The Emscripten bytecode is further compiled into separate JavaScript (*.js) and WebAssembly (*.wasm) files. When the JavaScript source file is loaded by the browser, it executes instructions to fetch the corresponding WebAssembly file asynchronously. These two files are then combined into an `npm` package. **(B)** A screenshot of the demo page showing the Repressilator model [88] in the BioModels database (BIOMD0000000012). After selecting a model via using the demo's search bar or uploading an SBML file, the demo allows the user to view SBML content as a tree–like structure and validate the SBML model subject to the validation options provided by libSBML. This particular model can be viewed at `https://libsbmljsdemo.github.io/#/view?m=BIOMD0000000012`

المنارة للاستشارات
www.manaraa.com

while Tellurium was not able to reproduce the bifurcation analysis of the mitotic division study [60] in an exchangeable format, it was possible to verify the observations regarding the shift in regulatory mechanism, and in doing so provided new insight from this alternative approach. A researcher may also wish to verify that the model reproduces certain expected behaviors. For example, if the model is expected to exhibit switch–like behavior, does this behavior occur at the correct input threshold? For models with feedback, such as integral feedback control [53], does the output exhibit robustness in the presence of perturbations? These types of validation require expert knowledge of the system. While there are tools and resources to help with this, the most important point for conveying this information to other researchers is to encode it as transparently and lucidly as possible, which is achieved using the literate notebook approach described here.

Tellurium's approach of blending standards with literate coding enables researchers to create rich, detailed workflows incorporating community standards. Tellurium allows the models and simulations from these notebooks to be shared with other tools via COMBINE archives. This allows other users to import these models and simulations and reproduce them using independently developed software tools. This is consistent with our original definition of reproducibility, as it enables robust cross–validation of results between tools, as opposed to simply repeating a previous simulation. It also helps ensure that the tools themselves are accurate and free of idiosyncrasies that could affect the analysis results. Model repositories such as BioModels [164, 168], JWS Online [229, 219], and the CellML model repository [174] have enabled widespread support for the SBML and CellML standards. Better tool support for SED–ML and COMBINE archives will help create a trend toward better adoption of these formats by repositories.

2.7.2 Comparison with Existing Software

Many dynamical modeling tools support exchanging models via the SBML format, including COPASI [127, 199], SBW [40], iBioSim [207], PathwayDesigner [9], CellDesigner [100, 101], VCell [205, 265, 45], CompuCell3D [293], PySCeS [218], BioNetGen [43], and PySB [176].

المنارة للاستشارات
www.manaraa.com

These tools have diverse feature sets and intended use cases, such as tissue modeling (CompuCell3D), rule–based modeling of molecular complexes (BioNetGen, PySB, VCell), and general modeling and simulation (all others). The tools also have different forms of user interaction, such as graphical user interfaces (COPASI, iBioSim, VCell) and graph–based network editors (CellDesigner, PathwayDesigner). Python–based tools such as PySCeS [218] and PySB [176] can be used with a Jupyter notebook, but do not feature integration of standards with the notebook itself. In general, Tellurium is useful when the user wishes to interactively edit and test standard–encoded models and simulations or produce presentations and PDFs of modeling studies.

Tellurium's Python foundation makes it easy to combine with other Python–based software such as PySCeS, COBRApy [86], and PySB. There are also many specialized Python packages for specific tasks such as moment closure approximation for stochastic models [94], parameter estimation [292], Bayesian inference [171], and estimating rate laws and their parameter values [299].

In biomedical research, certain tools have been created specifically to facilitate reproducible research. One such tool is Galaxy [112]. Galaxy is a web–based tool which allows users to create workflows describing experiments, e.g., metagenomic studies [235]. A similar tool with a focus on web services and which supports SBML–based workflows is Taverna [215]. Galaxy and Taverna allow users to annotate each step of the workflow, which provides a way for others to follow and understand the chain of reasoning used in the workflow's construction. This satisfies the requirement of transparency, as it allows users to view the sequence of steps used to produce a result. Although this approach is very different from a literate notebook in terms of the way the user interacts with the system, it shares the goal of allowing the user to see the sequence of steps used to produce a result and interrogate the specific procedure used in each of the steps. Galaxy and Taverna also allow users to share workflows via the web. However, neither tool attempts to directly address the problem of exchangeability with other software tools.

VisTrails [58] is another workflow system based on visual design. VisTrails focuses pri-

المنارة للاستشارات

www.manaraa.com

marily on generating rich, three–dimensional diagrams and visualizations based on input data and a specific sequence of steps. VisTrails also saves all changes made to a workflow and allows users to view previous versions, a concept termed “retrospective provenance” [233]. However, this approach also lacks exchangeability. Furthermore, while graphical tools may be more accessible because they abstract away the underlying algorithms, it can be difficult to isolate and correct software errors when a step fails due to bad input or an internal error.

Many other research software systems make use of notebooks, and some incorporate special extensions. StochSS [83], the GenePattern Notebook [249], the SAGE math system [92], and the commercial Mathematica software [322] all utilize notebooks which are specially tailored or feature special extensions for each respective application. However, none of these approaches attempt to solve the problem Tellurium addresses: workflow integration with exchangeable standards. Our usage of the literate notebook approach is intended to satisfy two specific requirements, which are distinct from other use cases: 1) to make these standards easy for humans to read, understand, and modify, without requiring expert knowledge of the technical specifications of the standards, and 2) provide an integrated workflow which facilitates exchangeability with other software.

The notebook approach used by Tellurium also has disadvantages. For example, while notebook files can be stored in a version control system, diffing and merging these files is difficult because the files are not line–based. Furthermore, large or complex analyses can be difficult to orchestrate using notebooks, as interacting with a large notebook with many cells can be cumbersome. For this reason, Tellurium is also distributed as a set of Python packages. When a workflow becomes too difficult to manage in a notebook environment, users can easily resort to separate Python, Antimony, and inline OMEX files in order to include the existing work in a more manageable system or to include the work in a version–controlled repository. Nevertheless, Tellurium's approach is highly useful in many crucial use cases, including testing models, experimenting with model variants, and as a final step in producing an analysis for other researchers in a transparent, visual presentation.

المنارة للاستشارات
www.manaraa.com

2.8 Review of Contributions

This chapter of the thesis described a platform for reproducible modeling and the supporting technologies. Due to the myriad of different technologies involved, it may be difficult to distill the core contributions. Therefore, they are summarized here:

- Development of a notebook–based approach to working with standards, along with an implementation of said notebook system.
- Creation of a unified format for representing COMBINE archives in human–readable form (inline OMEX).
- Demonstration of the robustness of this system via a number of examples and tests.
- Finally, the creation of a JavaScript library that can read and write SBML in the browser in order to support web applications using standards.

المنارة للاستشارات
www.manaraa.com

Chapter 3

PARAMETER ESTIMATION VIA THE ISLAND METHOD TO ENABLE SCALABLE MODELING

> The contents of this chapter are based on a manuscript in preparation for submission to *Briefings in Bioinformatics*. The curriculum vitae at the end of this thesis lists all manuscripts and resp. statuses.

Parameter estimation is a critical step in the construction of kinetic models. It is also a major impediment to scalability. Kinetic models become expensive to simulate as the size of the model increases. Additionally, increased dimensionality of the parameter space requires more objective function evaluations in order to achieve convergence. Thus, there are two factors that negatively impact scaling for kinetic models: simulation speed and dimensionality. The combination of these factors effectively creates an upper limit for the size of kinetic models. This is one reason why genome–scale models are often constructed using constraint–based modeling.

However, kinetic models are necessary to study the mechanisms behind transient biological processes such as signaling cascades and circadian rhythms. Thus, there is an impetus to improve the optimization of kinetic models.

Parameter fitting for kinetic models requires solving a non–convex, non–linear optimization problem, which defeats many optimization algorithms. However, one class of algorithms that is able to solve this problem is biologically–inspired population–based algorithms. A population–based algorithm maintains a pool of candidate solutions and improves the pool as a whole over time. Such algorithms are usually designed to balance convergence toward the minimum with diversity of the solution candidates. This class includes population–based

المنارة للاستشارات
www.manaraa.com

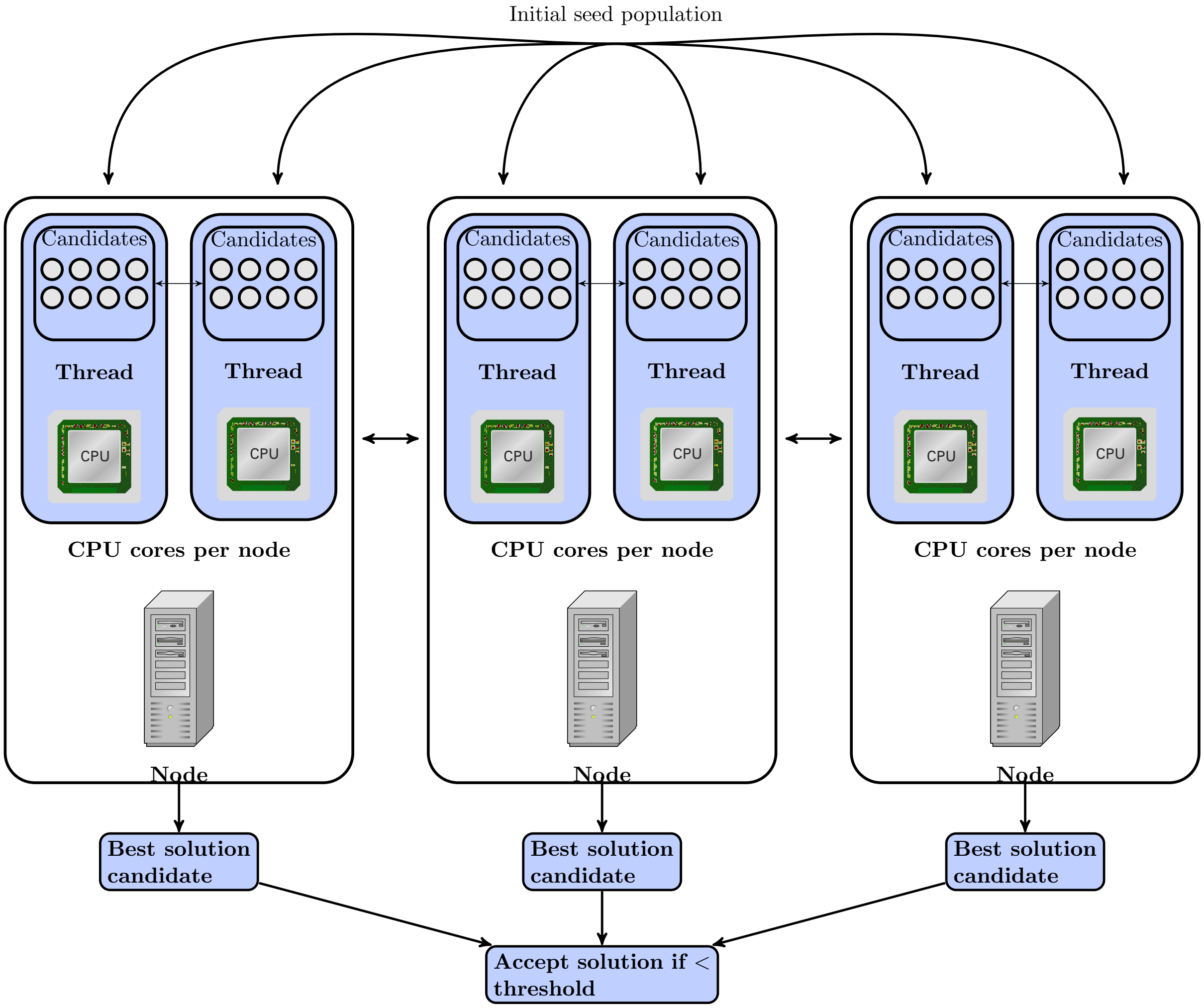


Figure 3.1: A visual depiction of the island method. A cluster containing a number of nodes (machines) and CPUs (one per processor core on each machine) is initialized randomly with a seed population. Each CPU maintains a population of solution vectors corresponding to one island. At the end of k iterations of the local algorithm (which can be different for each CPU), the best M solutions obtained so far are exchanged among CPUs on the same or different machines.

methods such as genetic algorithms (GAs) [76], differential evolution [290], particle–swarm optimization (PSO) [275], harmony search [105], the artificial bee colony algorithm (ABC) [144], and many others.

www.manaraa.com

```
# initialize populations for all threads
initialize_pop()
# run update loop on each cpu core in the cluster
for each thread:
  run local_algorithm k times
  send top_solutions to other_cpus
  update population with incoming_migrants
  if stopping_criterion:
    stop
  else
    repeat
```

Figure 3.2: Island method pseudocode. The island method runs locally on a number of threads, which can be on different CPU cores or different machines (the total number of threads should not exceed the total number of CPU cores in the cluster). In every update loop, the island method first runs the local algorithm (e.g. DE, ABC, etc.) for k iterations on each thread separately. Next, each island i sends its top l solutions to neighboring islands and receives up to $l \times m$ solutions from neighboring islands, where m is the connectivity of island i. This process repeats until a stopping criterion is reached (the objective function reaches a threshold value or the maximum number of iterations is exhausted).

3.1 Parallelization Approaches

The parallel nature of population–based methods lends these algorithms to efficient parallelization [61]. However, modern cloud–based computing is best suited for algorithms that can be efficiently parallelized across nodes on a network. From an implementation perspective, this creates unique challenges that are not present in multithreaded parallelization such as: how much data must be exchanged between nodes, how much synchronization is required,

المنارة للاستشارات
www.manaraa.com

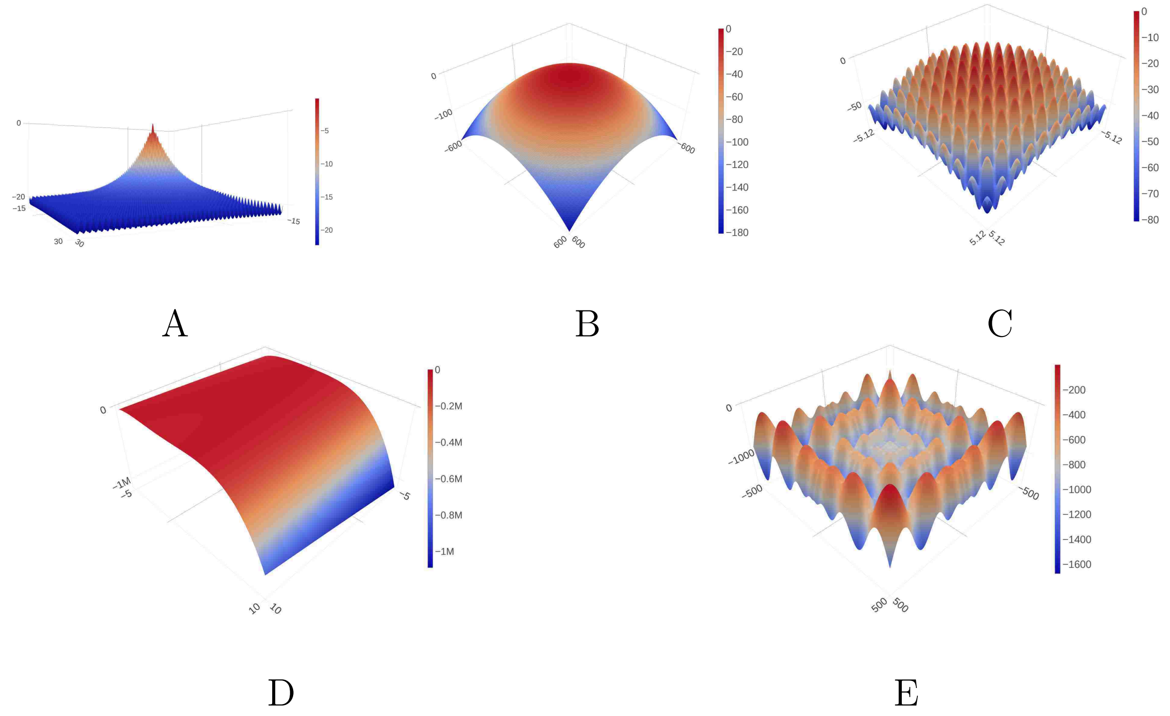


Figure 3.3: Two–dimensional plots of our five elimination benchmark functions. These functions are analytical, and hence allow for testing a much larger number of combinations than the BioPreDyn biochemical model benchmarks. The functions are analytic closed–form expressions: Ackley (A) [20], Griewank (B) [113], Rastrigin (C) [244], Rosenbrock (D) [251], and Schwefel (E) [237] functions. The plots are inverted so that the minimum value of the function is at the highest elevation, which allows for better visualization.

and how can the system be made fault–tolerant? These issues can have a major effect on the scalability of the algorithm.

One approach to parallelizing population–based algorithms is to simply distribute the population update step across different nodes [203]. For example, in differential evolution, new candidate decision vectors can be evaluated on different nodes and later combined to form the complete population for a given iteration. This manner of parallelization offers a potentially linear performance increase, and for this reason it is often used in *multithreaded*

www.manaraa.com

parallelization. However, certain factors limit the scalability of using this same approach for distributed algorithms.

An alternative approach is the so–called generalized island method [135, 253, 42, 183] (also referred to as the island model; this terminology is not used here because the word “model” is reserved for biochemical models). The island method is a meta–algorithm that parallelizes optimization by running different copies of an optimization algorithm (or different algorithms) on different nodes in the network. This is based on the biologically–inspired principle of *punctuated equilibria* [70], which states that isolated, independently–evolving populations tend to quickly reach equilibrium, and that once genetic equilibrium is reached there is little further genetic drift, leading to stagnation.

In the island method, islands running on different nodes periodically exchange solutions, thereby preventing stagnation and leading to better solutions. If the migration is based on the best individuals in each island, the method also exhibits accelerated convergence compared to a single island working alone. Additionally, the island method allows reusing previously existing algorithm implementations. The PaGMO2 library [42, 221], which is used in the optimization scheme described here, provides implementations for 14 global and 16 local fitting algorithms, which was possible in part because of pre-existing implementations of the algorithms.

The island method has been applied to various diverse optimization problems such as nuclear power plant feedwater monitoring [228], vehicle routing [141], and trajectory planning for spacecraft [42, 135]. This section of the thesis applies the island method to a set of very large, challenging dynamical models from systems biology [311] in order to quantify the performance of different algorithms and the performance gains due to parallelization using the island model. The distributed island method is able to accelerate parameter fitting and obtain better solutions for all models tested. As expected, the island method scales sub–linearly, but nevertheless provides a substantial improvement to performance and quality of fit. Our approach is completely asynchronous and uses very little network bandwidth, making it in principle an ideal candidate for scalling to massive numbers of nodes, even if

المنارة للاستشارات
www.manaraa.com

the maximum bandwidth of the network infrastructure is low. A reusable implementation of the software is available at `https://github.com/distrib-dyn-modeling/sabaody`. This software can be easily installed on cloud services such as Amazon Web Services (AWS) and Google Compute Engine.

3.2 Survey of Population–based Algorithms

Here, we briefly review the different population–based optimization algorithms used in this work. All algorithms used here are implemented in the PaGMO library, version 2 [42, 221].

3.2.1 Genetic Algorithms

One of the first biologically–inspirated population–based algorithms to be described, genetic algorithms attempt to mimic the natural process of evolution [162]. The algorithm consists of a population of "individuals" that each have genetic information encoding a candidate solution to the optimization problem. In this work, we are interested in finding parameter values for a model (called the "decision vector", see glossary) that minimize an objective function (in this work, the root mean square of the residuals of a timecourse simulation, or RMSE value).

Genetic algorithms allow the population to evolve by selecting individuals based on fitness (i.e. lower objective function value). Individuals with a better fitness (lower objective function) value have a higher chance of being selected. This ensures that the population approaches the objective function minimum over time, but it also provides robustness against being trapped in local minima because individuals with poor fitness values have a non–zero chance of being selected.

Genetic algorithms consist of a number of iterations. At each iteration, a new population is constructed from the old population. Though variants exist, most genetic algorithms use the following processes:

1. Start with a total population of N random individuals.

المنارة للاستشارات
www.manaraa.com

2. Select a pool of $M < N$ individuals randomly, giving preference for individuals with a better fitness score.

3. Combine the M individuals using the crossover and mutation operations below to produce a new population of size N.

   (a) Crossover: With a certain probability, select two individuals i, j and a random point P in the artificial chromosome (decision vector). The point P divides the chromosome into two halves. Combine the first half of the chromosome of individual i with the second half of the chromosome of individual j to form a new individual k with the new chromosome. Optionally, allow crossover to occur at multiple points.

   (b) Point mutation: With a certain probability, select an individual i to undergo a point mutation. The value of a single element of its decision vector (a single parameter value) is replaced with a new, random value.

Genetic algorithms were the subject of intense research in the 80's and 90's. One result of this research is the differential evolution algorithm, which performs quite well for real–valued decision vectors like the ones used in this work. Due to the success of differential evolution, genetic algorithms, as originally described, are less popular choices for parameter fitting. Nevertheless, they are still a subject of active research and are useful for problems that are not amenable to differential evolution.

3.2.2 Differential Evolution

Differential evolution is a highly successful variant of genetic algorithms that has been successfully used in very large, challenging problems such as parameter estimation in whole–cell models [146] and performs well in the benchmarks presented here.

The distinguishing feature of differential evolution is that it uses combinations of three individuals from the prior population via a difference vector to produce a new solution

المنارة للاستشارات

www.manaraa.com

candidate [290] [1]:

$$\boldsymbol{l} = \boldsymbol{i} + F\left(\boldsymbol{j} - \boldsymbol{k}\right)$$

where F $(0 \leq F \leq 2)$ is a tunable parameter and $\boldsymbol{l}$ is the newly created candidate. This difference vector operation replaces the mutation and crossover operators used in classic genetic algorithms. For each element of the decision vector (each parameter value), a random choice is made whether to draw the value from $\boldsymbol{l}$ or $\boldsymbol{i}$. The reason for the robustness of differential evolution is not well understood, but the authors [290] speculate that the use of a difference vector enables effective navigation of contours in the objective function. On a given contour line, all solutions have equal fitness. The use of a difference vector allows the algorithm to traverse a contour line without incurring a fitness penalty, thus enabling it to explore the contained region more thoroughly.

3.2.3 Differential Evolution Variants: SADE and DE1220

One of the most highly desirable properties for a fitting algorithm is parsimony of tunable parameters [2]. Every tunable parameter must be assigned a value to achieve optimal performance. Typically, these values are manually tuned for a given problem set, which is a laborious and time–consuming process. For example, differential evolution, as originally described [290], has two tunable parameters: F (the variation factor) and CR (the crossover probability) [3]. These values are usually set to reasonable defaults that are known exhibit decent performance for a wide range of test problems, but their optimality for problems outside the test benchmark is unknown (in this work, $F = 0.8$ and $CR = 0.9$ as provided by PaGMO). Moreover, under this scheme, one must solve a fitting problem (by tuning the algorithm parameters) in order to solve the original fitting problem (fitting the model).

[1] There are several variants of differential evolution that are classified based on the number of individuals that are used to calculate the difference vector, the selection of the seed individual $\boldsymbol{i}$, and the crossover scheme. The variant used here is classified as *DE/rand/1/bin* using the nomenclature of [290]

[2] This refers to parameters of the algorithm itself, not the model being fitted.

[3] The population size may also be considered as a tunable parameter, but not for the purposes of self–adaptation.

المنارة للاستشارات

www.manaraa.com

It is therefore not surprising that methods have been developed to automatically tune fitting algorithm parameters. One such method known as jDE uses a random sampling scheme to perturb the values of F and CR [52]. However, this method does not take into account whether these perturbations actually improve the convergence rate or not (in control theory parlance, it is comparable to an open loop controller). Calling this scheme “adaptive” might therefore be misleading, and hence it is not used here. Another variant, iDE, incorporates the F and CR parameters into the artificial chromosome of the individuals. This scheme *does* allow feedback, since F and CR values are selected by fitness over time, and it is denoted by `sade` in Table 3.3 and the figures in this chapter.

Finally, this chapter also makes use of the `de1220` algorithm provided by PaGMO [4]. This is a unique algorithm developed for PaGMO and is currently unpublished. It is essentially a variant of iDE that also augments the artificial chromosome to include an integer that represents the DE mutation variant. The variant of DE used in this chapter is *rand/1/bin*. This string consists of three parts identifying the algorithmic variant:

- *Selection:* determines how to choose the seed individual $\boldsymbol{i}$. Can be either *rand* or *best*.

- *Arity:* determines the number of individuals used to compute the difference vector. Higher values have an averaging effect.

- *Distribution:* The mathematical distribution used to control the crossover rate. In `de1220` it can be either *bin* (binomial) or *exp* (exponential).

PaGMO’s implementation of `de1220` can choose from 18 different configurations of the above variant options, which are not discussed here. A complete list is available at `https://esa.github.io/pagmo2/docs/cpp/algorithms/de1220.html`.

[4] `https://esa.github.io/pagmo2/docs/cpp/algorithms/de1220.html`

www.manaraa.com

3.2.4 Mimicking Animal Social Behavior: Particle Swarm and Artificial Bee Colony (ABC) Algorithms

This section describes algorithms that have a different take on biomimicry. It might be argued that natural evolution is a solution to an optimization problem of some sort (optimizing species for fitness for life on earth), but there are other scenarios where optimality is critical to survival. Examples include a flock of migrating birds, which must reach their destination with a minimal expendature of energy, or a swarm of insects searching for food sources. While the connection between these scenarios and the optimization of science or engineering problems is tenuous at best, algorithms inspired by flocking behavior have certainly proven useful for certain applications.

3.2.5 Other Population–Based Algorithms

Other population–based algorithms include simulated annealing and harmony search. These are only briefly covered here. For a more complete description of simulated annealing and harmony search respectively, see Corana *et al.* [72] and Geem *et al.* [105].

Simulated annealing is inspired by the process of annealing in metallurgy, wherein the heating and slow cooling of a metal allows impurities to be excluded by diffusion, thereby allowing growth and purification of the crystal grains of the metal. Diffusion occurs as a result of stochastic brownian motion. Simulated annealing operates analogously by making random perturbations to the decision vector. The magnitude of the perturbations depends on the "temperature," a numerical value that decreases over time. The system will eventually settle in a state of minimum energy corresponding to an objective function minimum. Out of all the optimization methods discussed here, simulated annealing is the only one with theoretically proven convergence properties. However, in practice, its performance is much worse than any of the evolutionary or swarm–based algorithms, so it is not used in this chapter.

Harmony search gains its name from a group of musicians improvising a melody. When

المنارة للاستشارات
www.manaraa.com

creating a new solution candidate, harmony search combines previous candidates as in the case of genetic algorithms. However, unlike genetic algorithms, which combine two solutions (or three in differential evolution), harmony search uses a combination of many solutions in its “repotoire” (harmony memory) to generate a new candidate.

3.3 Description of Benchmarks

The main objective of this chapter is to test whether the island method can improve the convergence time and solution quality for large, challenging dynamical models, and to quantify the performance of different algorithms and algorithm combinations. In order to perform these tests, a series of three benchmarks were used. The first benchmark exists to narrow down the combinatorial complexity of the different configuration options for the island method. This allowed us to run the remaining benchmarks in a realistic timeframe. The second benchmark tests the scaling and convergence performance of the island method on more challenging problems from the BioPreDyn benchmark, a set of large, challenging kinetic models. Table 3.2 summarizes the benchmark problems in this suite. Finally, the third benchmark, examines the performance of different algorithms and combinations. Partitioning the benchmarks in this way allowed us to test the main objectives in turn while keeping the number of different configurations within feasible limits.

The island method is highly configurable. In addition to specifying potentially different fitting algorithms for each island, the migration routes between the islands can be connected in any arbitrary topology. Although the system allows for fully customizable user–specified migration topologies, the implementation discussed here provides a set of 15 default topologies that can be generated automatically for any island number. These topologies are listed in Table 3.1. Since PaGMO2 contains 14 global and 16 local optimization algorithms, there are $30 \cdot 15 = 450$ algorithm / topology combinations to choose from for a given number of islands, not including algorithm combinations. Additionally, the island method can be used with different migration policies. This work uses a policy that selects the best M individuals from a source island, and replaces the worst M individuals in the destination island if the

المنارة للاستشارات

www.manaraa.com

Table 3.1: Topology presets included with the implementation. These presets can be used to generate a topology for any number of islands (except the hypercube, which requires that the number of islands be a power of two).

Topology
One-way Ring
Bidirectional ring
Bidirectional chain
Lollipop
Rim
1–2 Ring
1–2–3 Ring
Fully Connected (Complete Graph)
Broadcast
Hypercube
Watts-Strogats
Erdös-Rényi
Barabási-Albert
Extended Barabási-Albert
Extended Ageing Barabsi-Albert

More detailed information, including topology illustrations, is available at `https://sabaody.readthedocs.io/en/latest/topologies.html`.

replacement candidate has a better (lower) score. Another strategy is random selection and replacement, which can lead to more genetic diversity but has a less pronounced acceleration effect.

This large combination of options requires a systematic test to eliminate configurations that are unlikely to perform well. A set of 1120 benchmarks using different algorithm and

المنارة للاستشارات
www.manaraa.com

Table 3.2: Benchmark problems from the BioPreDyn suite. Four problems from the BioPreDyn benchmark suite [311] were used in this chapter. The original BioPreDyn suite contained a total of six problems [311]. However, B5 and B6 were not provided in a simulatable SBML format, nor indeed any standard, exchangeable format. An SBML *qual* package encoding of B5 was provided, but a qual encoding does not provide enough information to simulate the model. Additionally, the B6 model was based on spatial processes and thus would not have been usable with this workflow even if an SBML encoding had been provided.

	Param.	Obs.	Description	Data / noise process
B1	1759	44	Genome–scale model of yeast	Simulated Gaussian noise
B2	116	9	Central carbon metabolism (CCM) in *E. coli*	Experimental measurements
B3	178	47	Central carbon metabolism (CCM) and select transcription / translation processes in *E. coli*	Simulated with no noise
B4	117	13	Fermentation in Chinese Hamster Ovary (CHO) cells	Simulated with time–varying Gaussian noise

topology combinations was constructed based on five analytic objective functions plotted in two–dimensions in Figure 3.3. These analytic functions have a much lower computational cost than the BioPreDyn biochemical benchmark models, and are often used to evaluate newly developed nonlinear optimization algorithms. This set of tests is referred to as the elimination benchmark.

Table 3.3 shows that algorithm choice is more strongly correlated with performance than topology in these tests. Therefore, subsequent benchmarks focused on the effect of algorithm choice. The second test consists of benchmarks of problems B1 and B3 from the BioPreDyn suite [311] on the same topology / algorithm combination while varying the number of islands.

المنارة للاستشارات
www.manaraa.com

Table 3.3: Ranking of the elimination benchmark problems. The bottom row shows the mean difference between the highest and lowest ranking.

Topology	Range	Algorithm	Range
Hypercube	1–9	de1220	2–4
Rim	2–8	de+nelder mead	1–8
Bidirectional ring	4–7	bee colony	1–10
Bidirectional chain	1–11	de+praxis	2–9
Barabási–Albert [21]	1–13	de+sade	4–7
1–2 Ring	3–12	sade	2–9
Broadcast	1–14	de+de1220	4–9
Fully Connected	2–13	de	6–8
One–way ring	7–8	de+pso	7–10
1–2–3 Ring	4–12	de+nsga2	8–10
Ageing Extended Barábasi–Albert	6–10	pso	11–14
Erdös–Rényi [91]	3–14	ihs	11–14
Extended Barabási–Albert	5–12	xnes	11–14
Watts–Strogatz [318]	5–14	nsga2	12–14
Avg. diff.	**8.00**	**Avg. diff.**	**3.81**

These problems are expensive, but also provide a way to critically test the convergence properties and performance of the island method. Next, different algorithm combinations were run on problems B2 and B4 from BioPreDyn. The purpose of this test was to look for trends in algorithm performance, and to investigate whether a “synergistic” effect could be observed from combinations of different algorithms.

المنارة للاستشارات

www.manaraa.com

3.4 Details of Each Benchmark

3.4.1 Elimination Benchmark

Ten–dimensional versions of the functions in Figure 3.3 were run for three replicates on the combinations of topologies and algorithms shown here. In some cases, algorithm combinations were used, such as `de+sade`. In such cases, the algorithms were alternated along the topology structure if linear or ring–shaped, or arbitrarily distributed otherwise (e.g. hypercube and random graph–based models). However, once assigned, the algorithm positions were not changed for the duration of all benchmarks. Results were grouped by topology and algorithm/combination respectively and independently ranked for each of the five problems according to the total number of rounds required for convergence. A lower score implies that the benchmark finished in fewer rounds. This benchmark consisted of up to 2000 rounds of migration, each interspersed with 1000 iterations of the local algorithm on each node. The fitting problem was terminated when the MSE of any decision vector dropped below a cutoff value of 0.01 with respect to the best known solution. Migration was fully asynchronous.

3.4.2 Scalability Benchmark

In order to quantify the performance improvement of the island method, problems B1 and B3 from the BioPreDyn suite were benchmarked for various numbers of islands, but using a fixed algorithm. B1 is a genome scale *S. cerevisiae* model whereas B3 is a model of central carbon metabolism and transcription in *E. coli*. Both of these are large and challenging models, with the original authors reporting $\approx$ 1–week fitting times for both [311]. Figure 3.4 shows that reasonable fits can be obtained in a day for B3 on a 16–core cluster.

3.4.3 Algorithm Benchmark

Table 3.3 suggests that combining local and global fitting algorithms can lead to better performance than either type of algorithm individually. For example, `de+nelder mead`, `de+praxis`, `de+sade`, and `de+de1220` are all ranked better than `de` alone. To test whether

www.manaraa.com

this trend would hold for more realistic fitting problems, problems B2 and B4 from the BioPreDyn suite were tested against different algorithms and algorithm combinations, and the results were quantified in Table 3.4. Figures 3.5 and 3.6 show the value of the current champion (the individual with the best fitness) in the population over time. Figures 3.7–3.14 show the best fits obtained and normal probability plots of the respective residuals.

3.5 Scaling Behavior

Figure 3.15 shows that the island method typically increases the quality of fit by a factor of 2 when going from 1 to 16 islands. However, in the case of B3, the quality of fit increases by more than an order of magnitude. All problems except B3 have simulated or experimental noise, which narrows the gap between the initial population and the final convergent solution. When the fitness of the space of solutions spans several orders of magnitude, as with B3, the island method does deliver more than an order of magnitude improvement when going from 1 to 16 islands.

There are two principal benefits of the island method. The first is due to elitism in the migration scheme. In our implementation, each island sends its best scoring individuals to neighboring islands. These individuals then improve the overall fitness of the population during the next iteration of the algorithm. This helps increase the speed of convergence.

The second benefit is due to increased "genetic diversity" of the total population among all islands. Having a diverse population of decision vectors is important for thoroughly exploring the search space. In an island configuration with N nodes each having a population of size M, the total number of individuals is $N \cdot M$. Instead of using the island method, one could use a single algorithm with a total population size of $N \cdot M$ to achieve equally good quality of fit. However, the N islands operate in parallel whereas a single population would not (by assumption).

www.manaraa.com

Table 3.4: Results for problems B2 and B4. The objective function value (score) and timings are given as mean ± standard deviation.

Prob.	Algorithm	Num. islands	Score ± std. dev.	Hours	Rounds	Converged?	Replicates
B2	bee colony	1	0.3 ± 0.0091	3.1 ± 0.11	1000.0 ± 0.0	No	3
		16	0.34 ± 0.013	5.4 ± 1.8	1000.0 ± 0.0	No	3
	de	1	0.31 ± 0.011	2.0 ± 0.0042	1000.0 ± 0.0	No	3
		16	0.28 ± 0.004	2.0 ± 1.2	740.0 ± 450.0	Yes	3
	de+nelder mead	1	0.32 ± 0.0075	2.0 ± 0.026	1000.0 ± 0.0	No	3
		16	0.29 ± 0.016	3.1 ± 0.86	930.0 ± 130.0	Yes	3
	de+praxis	1	0.32 ± 0.011	2.3 ± 0.0031	1000.0 ± 0.0	No	3
		16	0.29 ± 0.019	2.0 ± 1.2	770.0 ± 400.0	Yes	3
	de+sade	1	0.32 ± 0.0053	2.2 ± 0.013	1000.0 ± 0.0	No	3
		16	0.28 ± 0.012	1.9 ± 0.88	660.0 ± 300.0	Yes	3
	de1220	1	0.3 ± 0.012	2.0 ± 0.049	1000.0 ± 0.0	No	3
		16	0.28 ± 0.012	0.62 ± 0.19	200.0 ± 120.0	Yes	3
	sade	1	0.31 ± 0.0068	2.3 ± 0.028	1000.0 ± 0.0	No	3
		16	0.28 ± 0.0011	3.3 ± 1.0	1000.0 ± 0.0	No	3
B4	bee colony	1	0.08 ± 0.0043	1.8 ± 0.71	500.0 ± 0.0	No	3
		2	0.12 ± 0.02	1.1 ± 0.059	500.0 ± 0.0	No	3
		4	0.12 ± 0.031	1.1 ± 0.007	500.0 ± 0.0	No	3
		8	0.098 ± 0.008	1.1 ± 0.023	500.0 ± 0.0	No	3
		16	0.089 ± 0.0073	1.1 ± 0.034	500.0 ± 0.0	No	3
	de	1	0.086 ± 0.0074	0.76 ± 0.027	500.0 ± 0.0	No	3
		2	0.076 ± 0.0051	0.84 ± 0.067	500.0 ± 0.0	No	3
		4	0.07 ± 0.006	0.84 ± 0.061	500.0 ± 0.0	No	3
		8	0.067 ± 0.0034	0.63 ± 0.35	380.0 ± 200.0	Yes	3
		16	0.069 ± 0.0069	0.47 ± 0.34	290.0 ± 200.0	Yes	3
	de+nelder mead	1	0.089 ± 0.0046	0.78 ± 0.035	500.0 ± 0.0	No	3
		2	0.11 ± 0.015	0.82 ± 0.062	500.0 ± 0.0	No	3
		4	0.074 ± 0.0091	0.75 ± 0.091	490.0 ± 15.0	Yes	3
		8	0.084 ± 0.0034	0.79 ± 0.008	500.0 ± 0.0	No	3
		16	0.076 ± 0.011	0.58 ± 0.37	370.0 ± 220.0	Yes	3
	de+praxis	1	0.085 ± 0.0031	0.77 ± 0.032	500.0 ± 0.0	No	3
		2	0.11 ± 0.0053	0.98 ± 0.31	500.0 ± 0.0	No	3
		4	0.09 ± 0.013	0.82 ± 0.045	500.0 ± 0.0	No	3
		8	0.088 ± 0.02	0.81 ± 0.056	500.0 ± 3.0	No	3
		16	0.072 ± 0.0098	0.64 ± 0.29	410.0 ± 160.0	Yes	3
	de+sade	1	0.088 ± 0.0028	0.76 ± 0.034	500.0 ± 0.0	No	3
		2	0.084 ± 0.0074	0.82 ± 0.0099	500.0 ± 0.0	No	3
		4	0.066 ± 0.0024	0.51 ± 0.27	330.0 ± 150.0	Yes	3
		8	0.071 ± 0.007	0.67 ± 0.29	410.0 ± 150.0	Yes	3
		16	0.07 ± 0.0098	0.41 ± 0.34	260.0 ± 210.0	Yes	3
	de1220	1	0.083 ± 0.0064	0.71 ± 0.0091	500.0 ± 0.0	No	3
		2	0.074 ± 0.0075	0.8 ± 0.024	500.0 ± 0.0	No	3
		4	0.07 ± 0.0086	0.21 ± 0.11	140.0 ± 74.0	Yes	3
		8	0.067 ± 0.0034	0.47 ± 0.28	300.0 ± 170.0	Yes	3
		16	0.066 ± 0.00095	0.67 ± 0.31	400.0 ± 170.0	Yes	3
	sade	1	0.088 ± 0.0051	0.78 ± 0.085	500.0 ± 0.0	No	3
		2	0.099 ± 0.032	0.63 ± 0.033	500.0 ± 0.0	No	3
		4	0.073 ± 0.01	0.42 ± 0.3	280.0 ± 190.0	Yes	3
		8	0.072 ± 0.006	0.83 ± 0.043	500.0 ± 0.0	No	3
		16	0.07 ± 0.0067	0.64 ± 0.31	390.0 ± 190.0	Yes	3

3.6 Comparison with Distributed Algorithms

In comparison with the island model, distributed algorithms divide a fixed amount of work equally among N cores. Hence, when using 16 cores, a distributed algorithm would be

المنارة للاستشارات
www.manaraa.com

expected to be 16x faster. The primary reason to prefer the island model over distributed algorithms is the ability to re-use pre-existing algorithm implementations. If a distributed implementation is available, it is usually preferred. Nevertheless, there are some cases where this is not true. First, the bandwidth requirements of a distributed algorithm are much larger than the island method.

For example, differential evolution forms new decision vectors from a linear combination of three individuals from the prior iteration. For a population of size M with N nodes, each node will evaluate M/N decision vectors and require $3M/N$ prior decision vectors to be sent over the network interface, which imposes bandwidth requirements for the network infrastructure. Our benchmarks used up to 2 million total iterations (2000 rounds of 1000 iterations) across a total population size of $16 \cdot 300 = 4800$. Assuming a decision vector of length 100, double–precision, this equates to 20 Tb (terabits) of information that must be sent over the network. In comparison, the island method using a migration rate of 4 individuals per island per round would require only $4 \cdot 16 \cdot 2000 \cdot 100 \cdot 64 = 820Mb$. Whether or not these requirements pose a limtation depends on the networking infrastructure used.

More importantly, distributed optimization algorithms must be re-implemented using a distributed computing framework such as Spark [27] or Dask [23]. In many cases, an independently–validated reference implementation of an algorithm is available in C/C++ for running locally on a single node, but not on distributed nodes. Thus, if a researcher wishes to evaluate a given fitting algorithm using distributed computing, the researcher must first re-implement the algorithm itself using distributed technologies. Since computing frameworks like Spark [27] or Dask [23] are evolving rapidly, it is not known whether a re-implementation will be rendered obsolete by technology changes, thereby re-creating the same dilemma again in the future.

3.7 Hardware Specification

Our hardware used for this study consisted of two workstations with a 10–core Intel® Xeon® CPU E5–2660V3 at 2.6 GHz with 24 GB RAM and a 6–core Intel® Xeon® CPU E5–2620V2

المنارة للاستشارات
www.manaraa.com

at 2.1 GHz with 64 GB RAM.

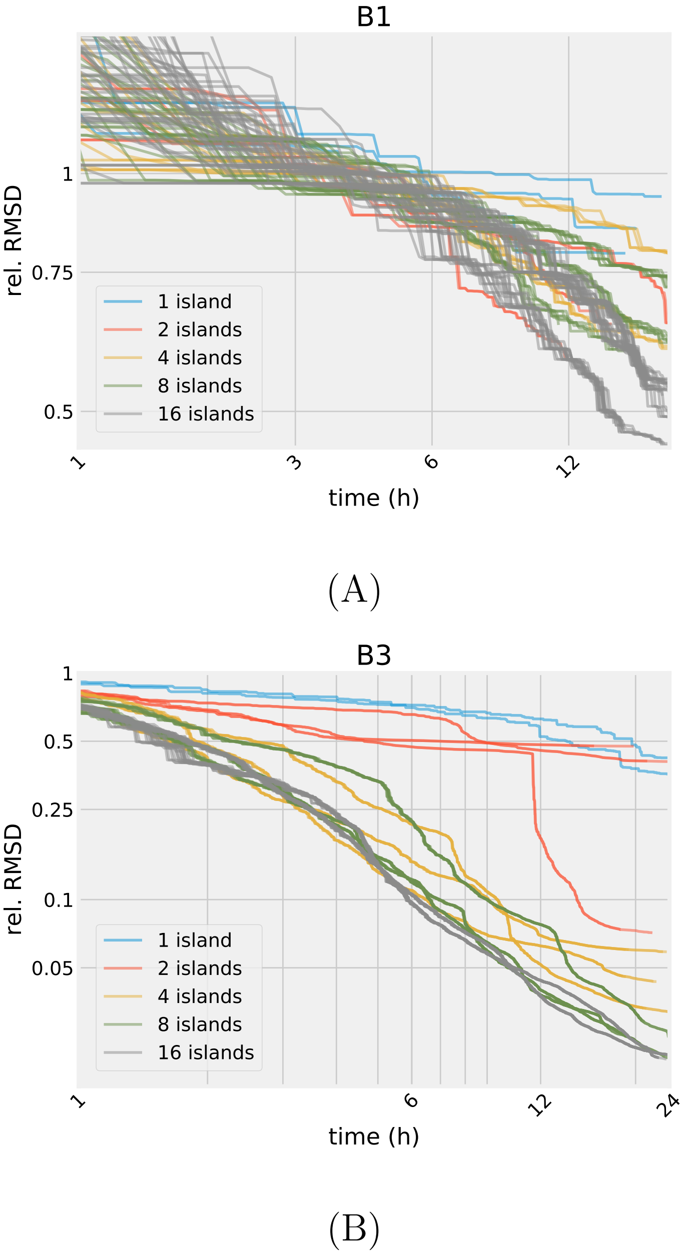
3.8 Conclusions

The aim of this chapter was to develop an implementation of the island method and test several hypotheses. Chief among the hypotheses was whether the island method is viable for efficiently parallelizing fitting algorithms, which was verified to be true. The island method provided considerable speedups for all benchmark problems tested here. The next hypothesis was whether different local algorithms or combinations of algorithms have significant performance advantages over a single, homogeneous algorithm configuration. The results shown here indicate that there is little variability in algorithm performance for different problems, and that combinations of different algorithm types (in a heterogeneous island configuration) does not significantly enhance performance. This is in contrast to the results reported by Villaverde *et al.* [310], which suggest that hybrid algorithms (combinations of locally convergent and globally convergent algorithms similar to the heterogeneous island configurations used here) do speed up fitting. However, Villaverde *et al.* did not test the highly performant evolutionary algorithms used here (e.g. DE variants), and also report difficulty fitting the B3 problem, suggesting that their method may be sensitive to an abundance of local minima.

In summary, the results shown here indicate that the island method is a useful approach when fitting large kinetic models, especially where the quality of fit is a major concern. Algorithm and topology choice had only a minor effect on overall performance, which suggests that a configuration consisting of the *de1220* algorithm with the rim topology exhibits reasonable performance across problems.

للاستشارات المنارة

www.manaraa.com



(A)

(B)

(C) Settings

	B1	**B3**
Pop. per island	300	300
Rounds	100	2000
Generations bet. rounds	1000	1000
Parameter range	0.1–10x	0.1–10x

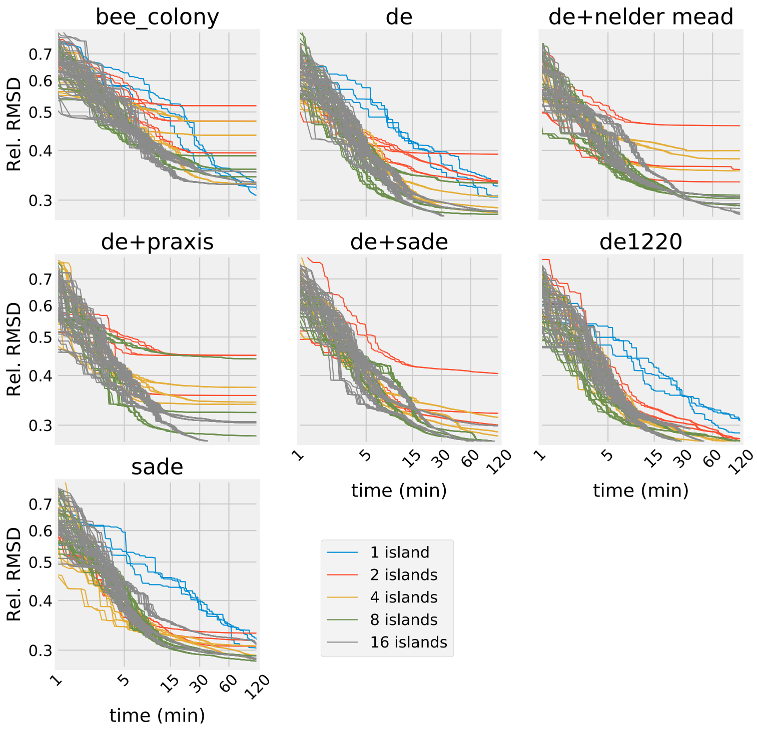
(D) Results

Problem	Num. islands	Mean score ± std. dev.	Replicates
B1	1	0.86 ± 0.072	3
	2	0.61 ± 0.031	3
	4	0.61 ± 0.022	3
	8	0.61 ± 0.035	3
	16	0.48 ± 0.033	3
B3	1	0.33 ± 0.0047	2
	2	0.32 ± 0.22	2
	4	0.045 ± 0.014	2
	8	0.022 ± 0.0016	2
	16	0.02 ± 0.00047	2

Figure 3.4: Convergence curves for problems B1 (A) and B3 (B) from the BioPreDyn–Bench suite [311] for various numbers of islands. Each curve plots the best *champion fitness* (i.e. the best solution up to the current time) per island over time. The settings for each benchmark are shown (C). For a given number of rounds, table (D) shows that increasing the island size yields an improvement in fitted parameters. As with BioPreDyn, the values of all fitted parameters were constrained to 0.1–10× the nominal value. The fitness value in (A) and (B) is computed from the root–mean–square deviation for each state variable normalized by the state variable's average (in B1, the values are also rescaled to balance error contributions). Not all islands finish at the same time due to the use of variable–step integration in SBML simulation.

المنارة للاستشارات

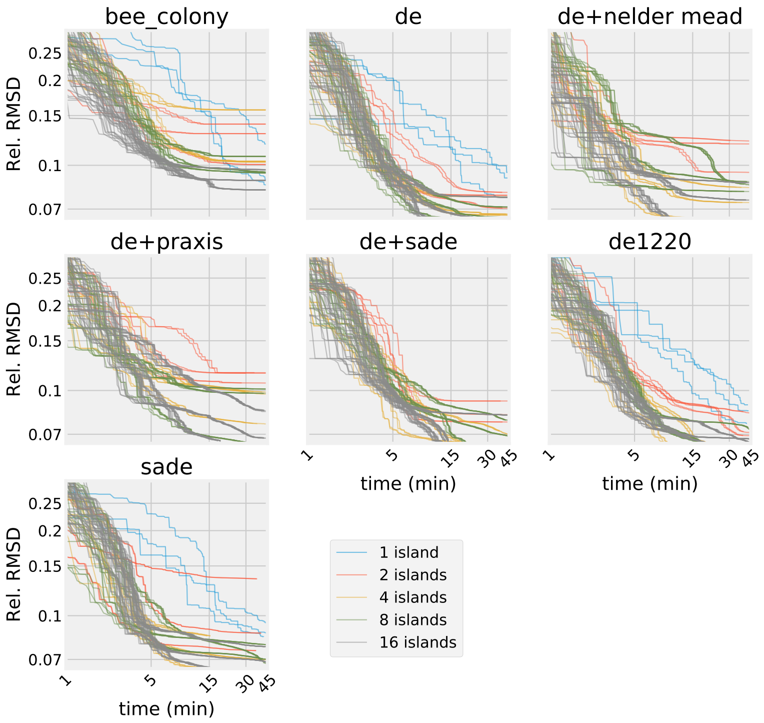
www.manaraa.com



	B2
Pop. per island	100
Max. rounds	1000
Generations bet. rounds	1000
Parameter range	0.1–10×

Figure 3.5: Convergence curves for problem B2 from the BioPreDyn suite. To test the effect of different algorithms and combinations of algorithms on the convergence rate, the top performing algorithms and combinations determined from the elimination benchmark were employed in the next benchmark. Each trace represents the champion value (the individual with the best fitness) for a single island running on 1 CPU core. Single island variants are elided from combination benchmarks containing two or more algorithms.

www.manaraa.com



	B4
Pop. per island	100
Max. rounds	500
Generations bet. rounds	1000
Parameter range	0.1–10×

Figure 3.6: Convergence curves for problem B4 from the BioPreDyn suite. As with Figure 3.5, the plot shows the champion values over time for each benchmark configuration. Traces are as in Figure 3.5.

المنارة للاستشارات

www.manaraa.com

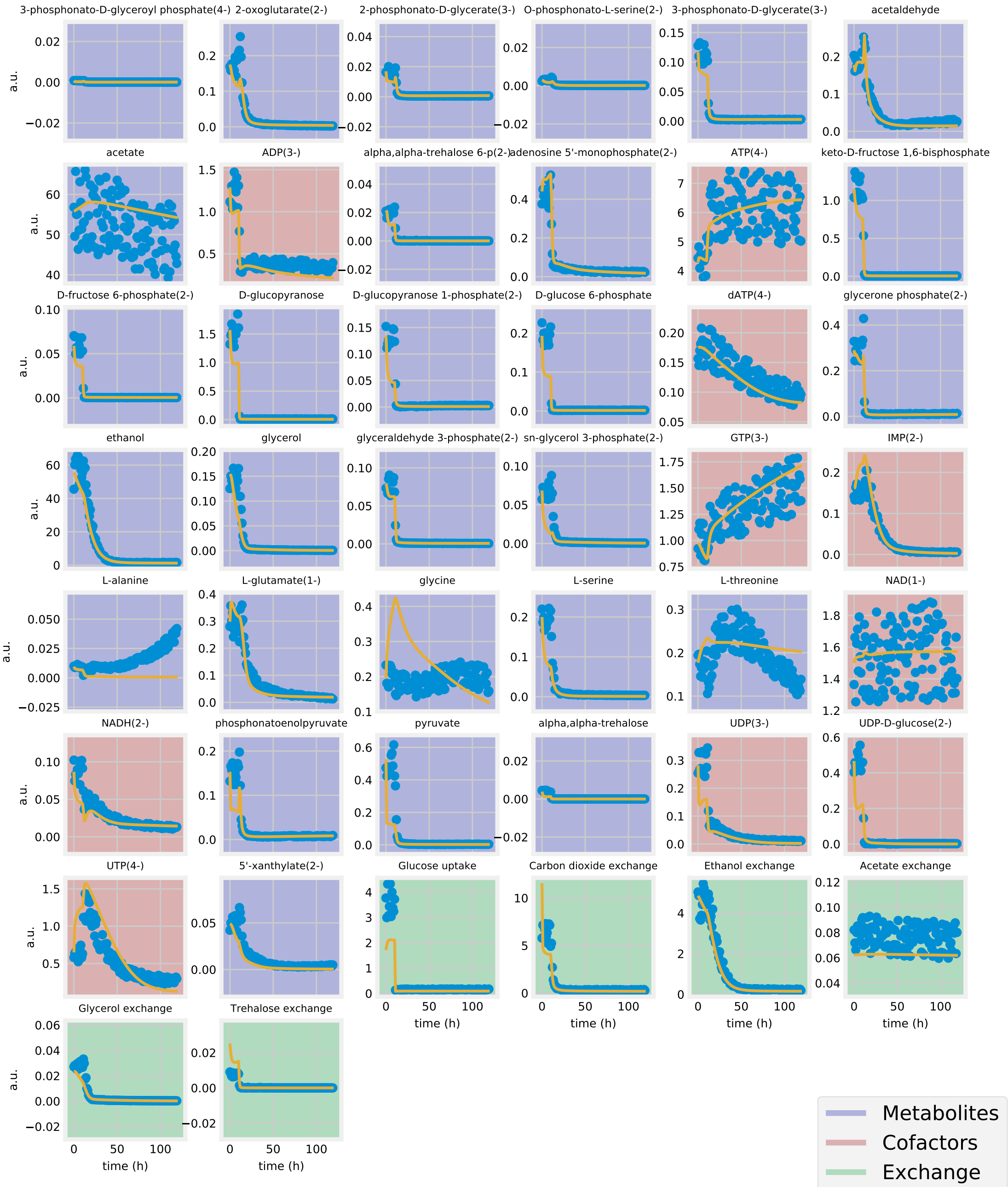


Figure 3.7: The best solution obtained by the island method for problem B1 from the BioPreDyn suite after 100 rounds. This benchmark was not intended to run to convergence due to the large size (1759 parameters) of the B1 model and corresponding long fitting time. The objective function for this benchmark is calculated by normalizing the RMSE value for each quantity to the reference RMSE value (for each quantity) and taking the RMS of the weighted values.

www.manaraa.com

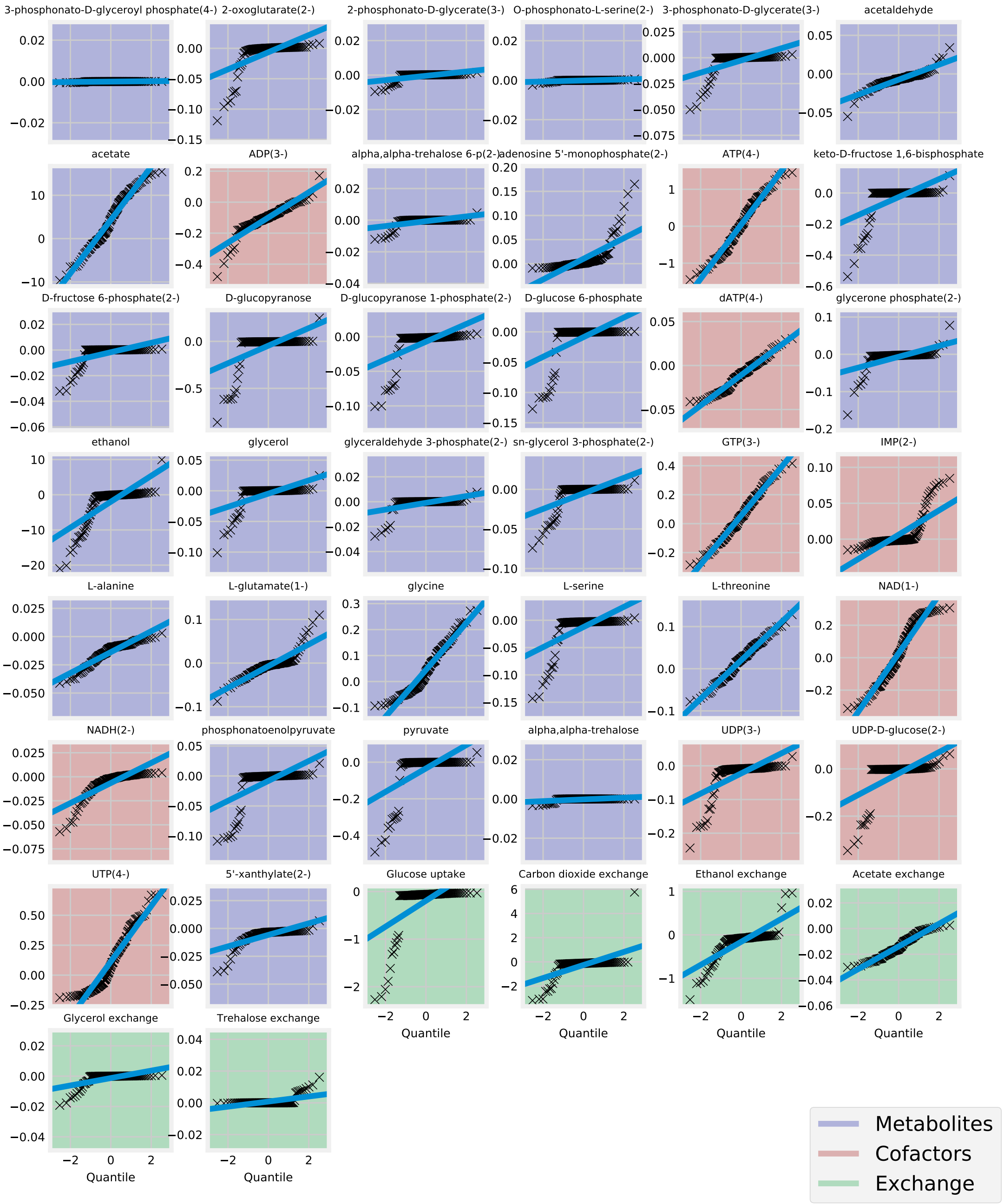


Figure 3.8: B1 normal probability plots. Residuals for each of the fitted quantities in Figure 3.7 is shown as a normal probability plot. Residuals that lie along a straight line are normally distributed. Theoretical quantiles for each quantity are plotted on the x axis (calculated using the sample mean and variance), whereas the y axis shows the residual value. 100 rounds was insufficient to ensure convergence for this large (1759–parameter) model after one day of fitting. Hence, some quantity residuals deviate from normally distributed values.

www.manaraa.com

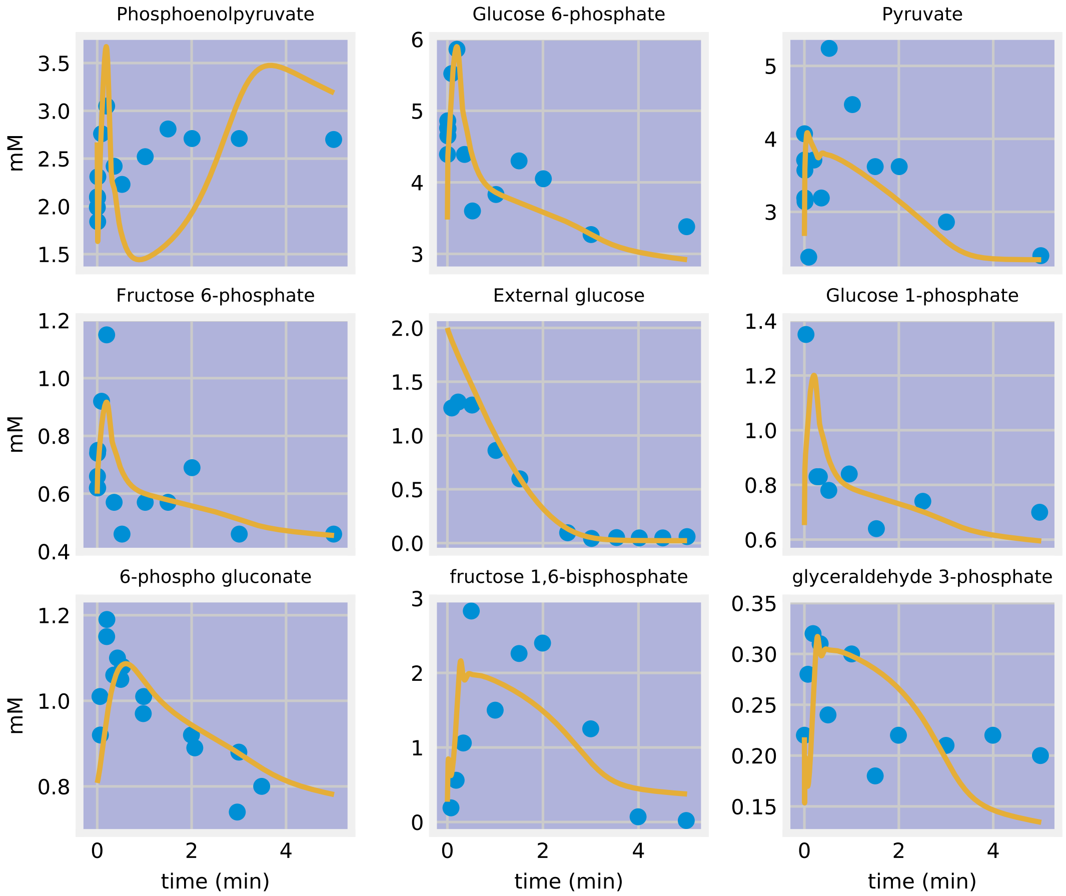


Figure 3.9: The best solution obtained by the island method for problem B2 from the BioPreDyn suite after 1000 rounds. The objective function for this benchmark is calculated by normalizing the RMSE value for each quantity to the quantity mean and taking the RMS of the weighted values. This objective function is slightly different from B1. Qualtities are normalized to the means here whereas they are normalized to RMSE per quantity in B1. The objective function so–calculated can be thought of as the “overall” RMSE and is ≈ 2% for this fit (the B3 benchmark contains no noise).

المنارة للاستشارات
www.manaraa.com

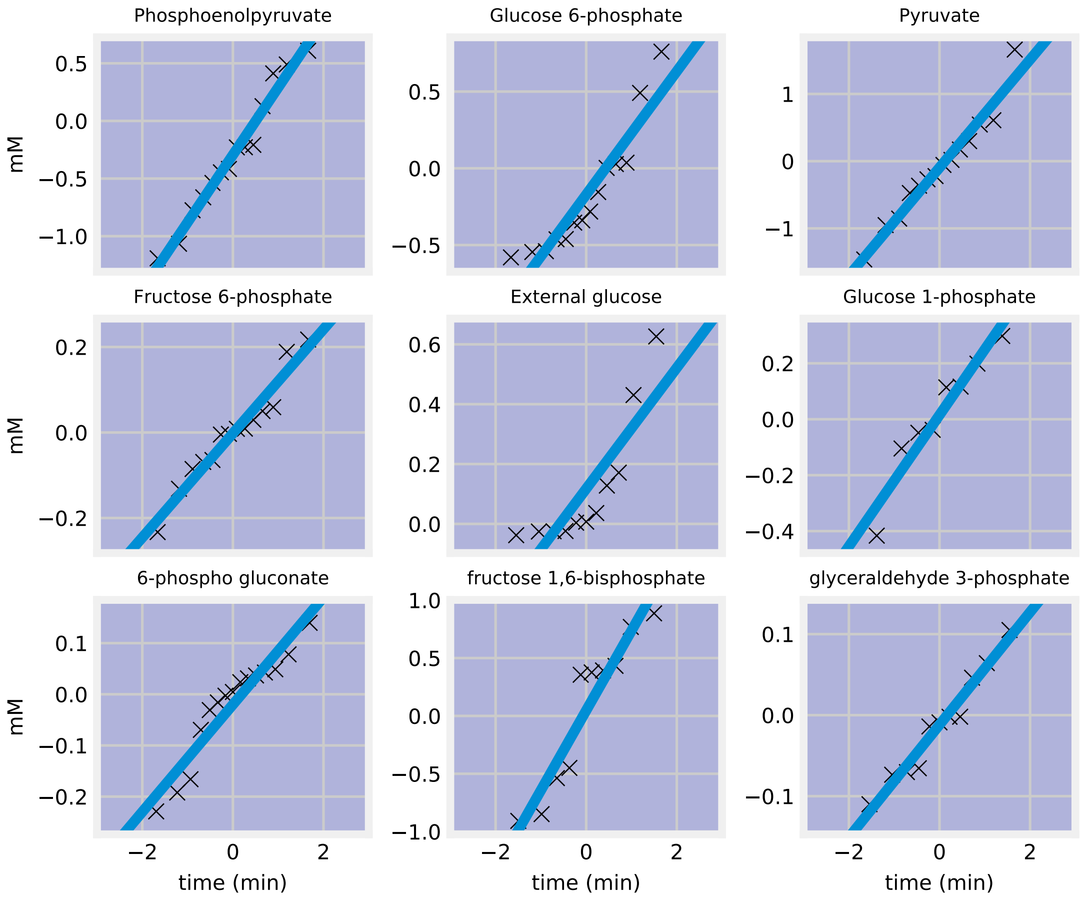


Figure 3.10: B2 normal probability plots. Residuals for each of the fitted quantities in Figure 3.9 is shown as a normal probability plot.

www.manaraa.com

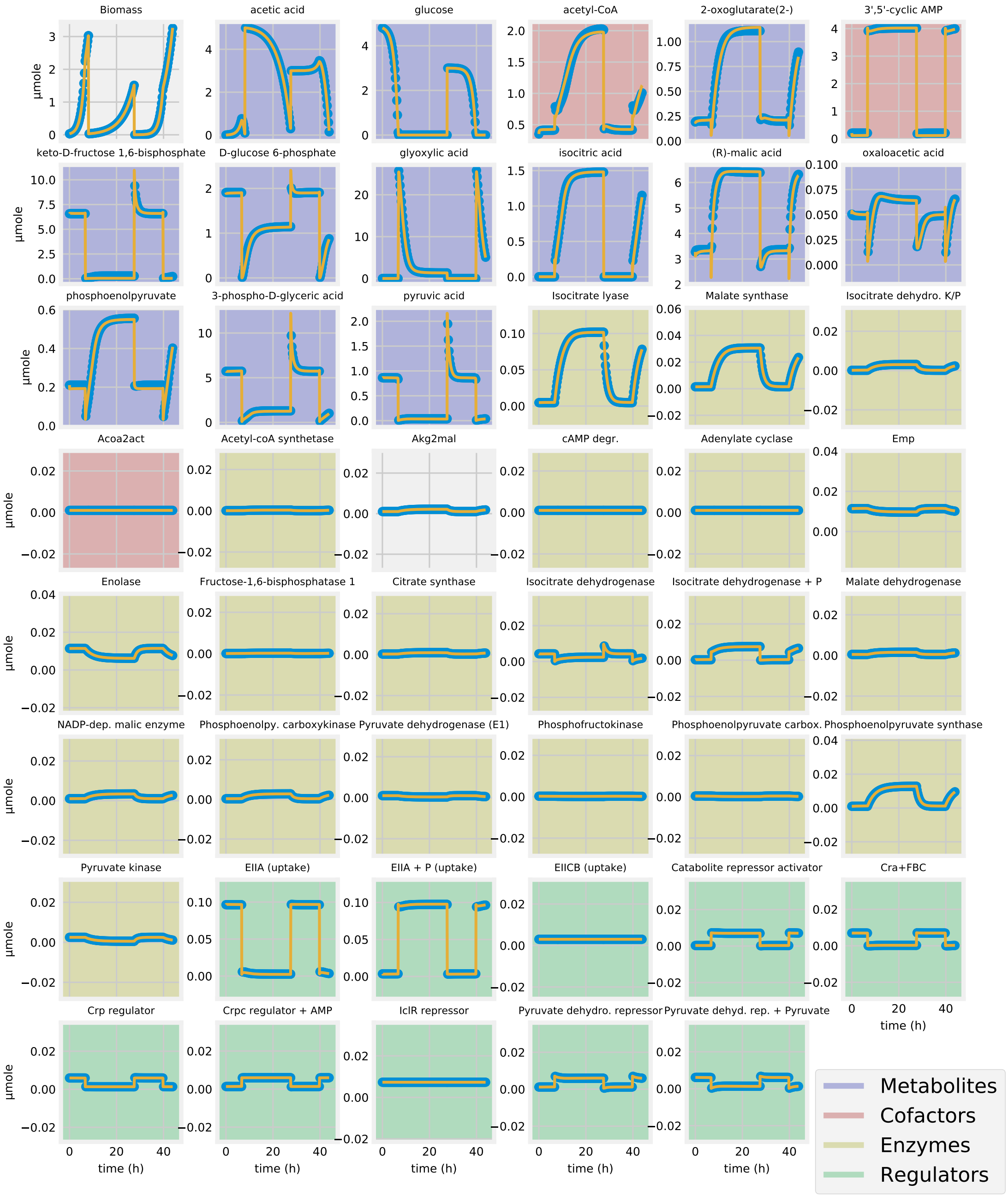


Figure 3.11: The best solution obtained by the island method for problem B3 from the BioPreDyn suite after a maximum of 2000 rounds. The objective is calculated identically to B2. The B3 model is an *E. coli* central carbon metabolism (CCM) model combined with a limited gene regulatory model (hence the enzyme and regulator variables). This model simulates diauxic shifts, which appear as discontinuities in most plots (esp. biomass).

www.manaraa.com

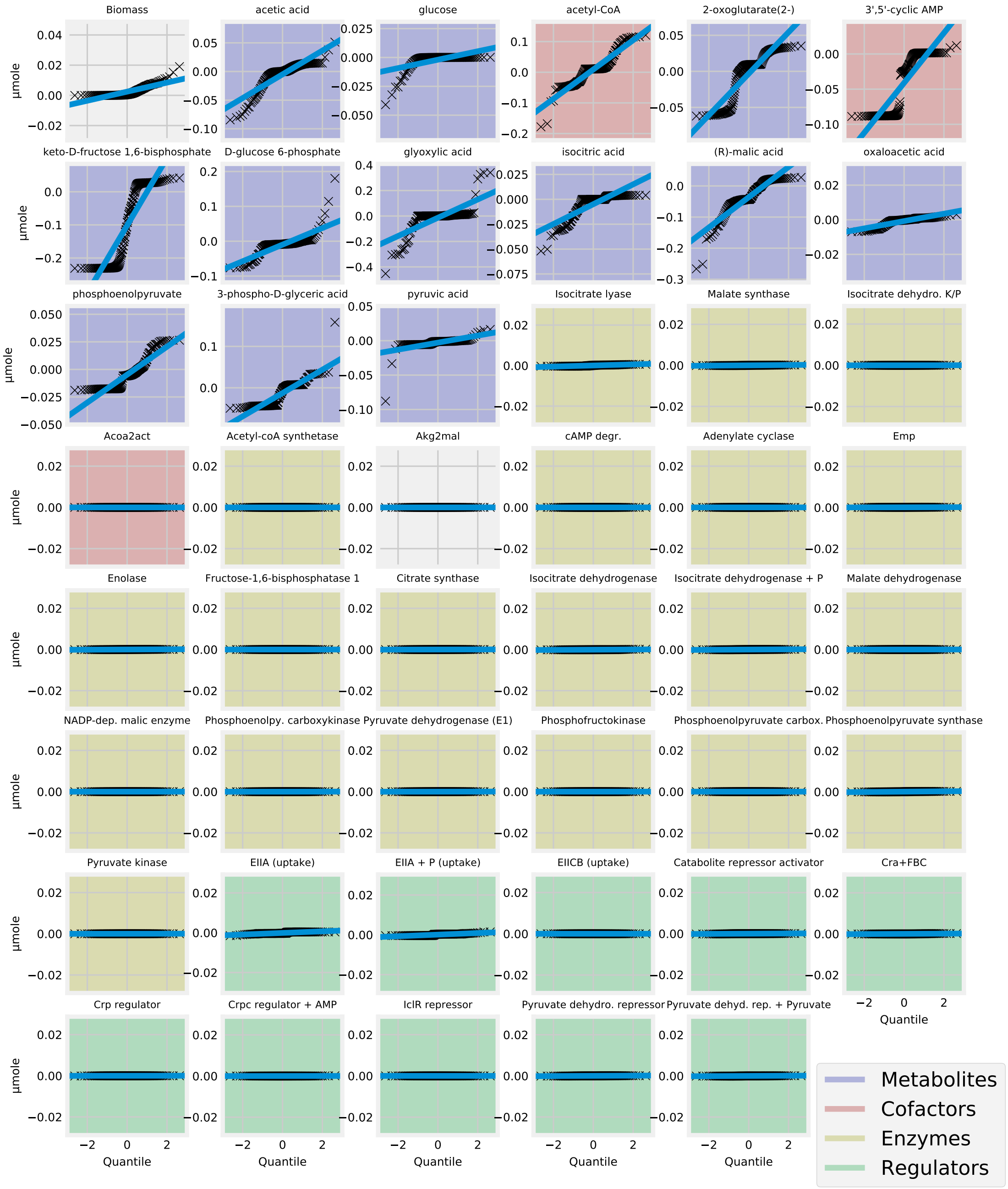


Figure 3.12: B3 normal probability plots. Residuals for each of the fitted quantities in Figure 3.11 is shown as a normal probability plot. Residuals that lie along a straight line are normally distributed. The B3 benchmark contains no noise, so residuals are not expected to be normally distributed.

المنارة للاستشارات
www.manaraa.com

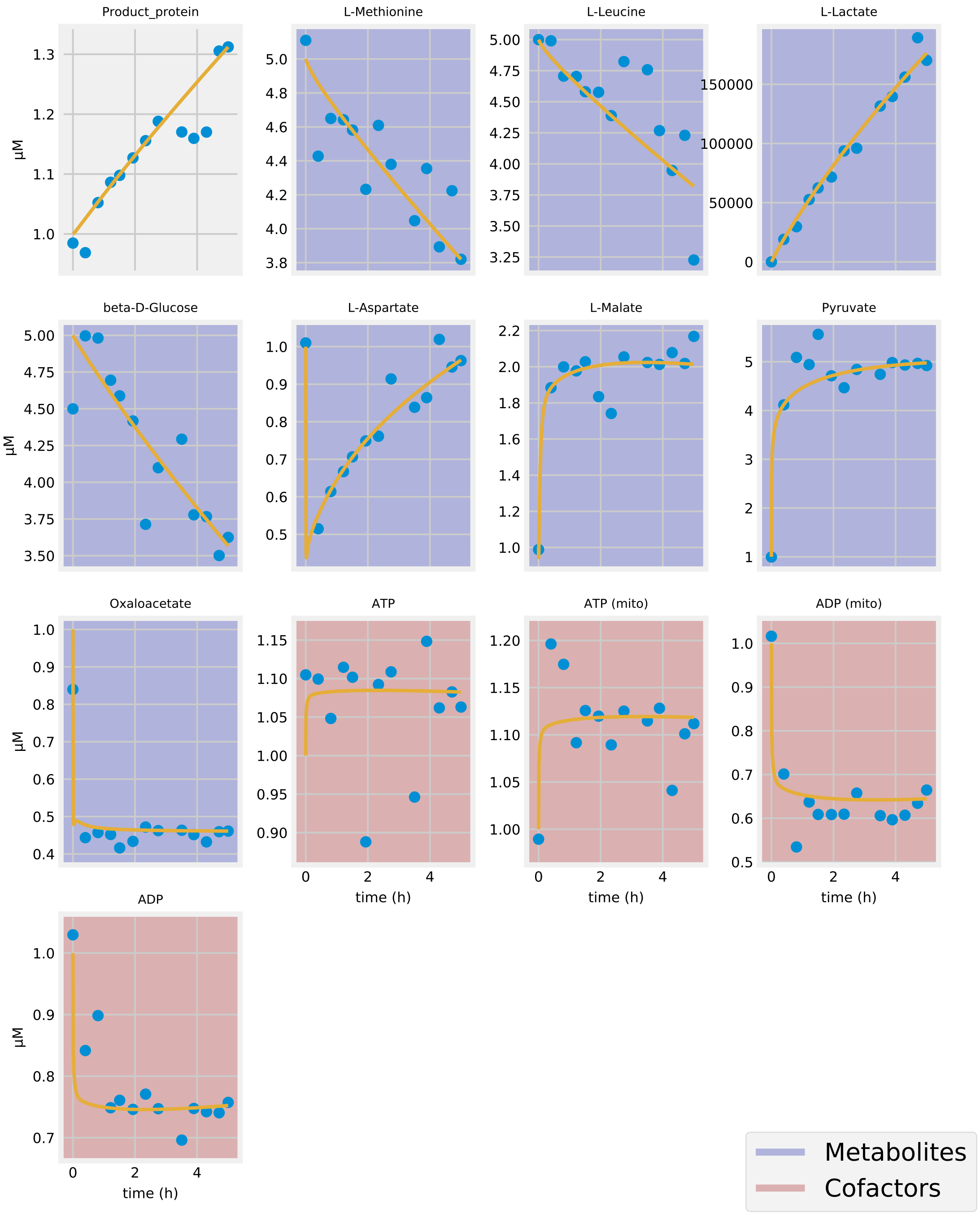


Figure 3.13: The best solution obtained by the island method for problem B4 from the BioPreDyn suite after a maximum of 500 rounds. The objective is calculated identically to B3. The B4 model is a fermentation process in Chinese Hamster Ovary (CHO) cells. Lactate is a by–product of fermentation, hence the divergent curve.

المنارة للاستشارات

www.manaraa.com

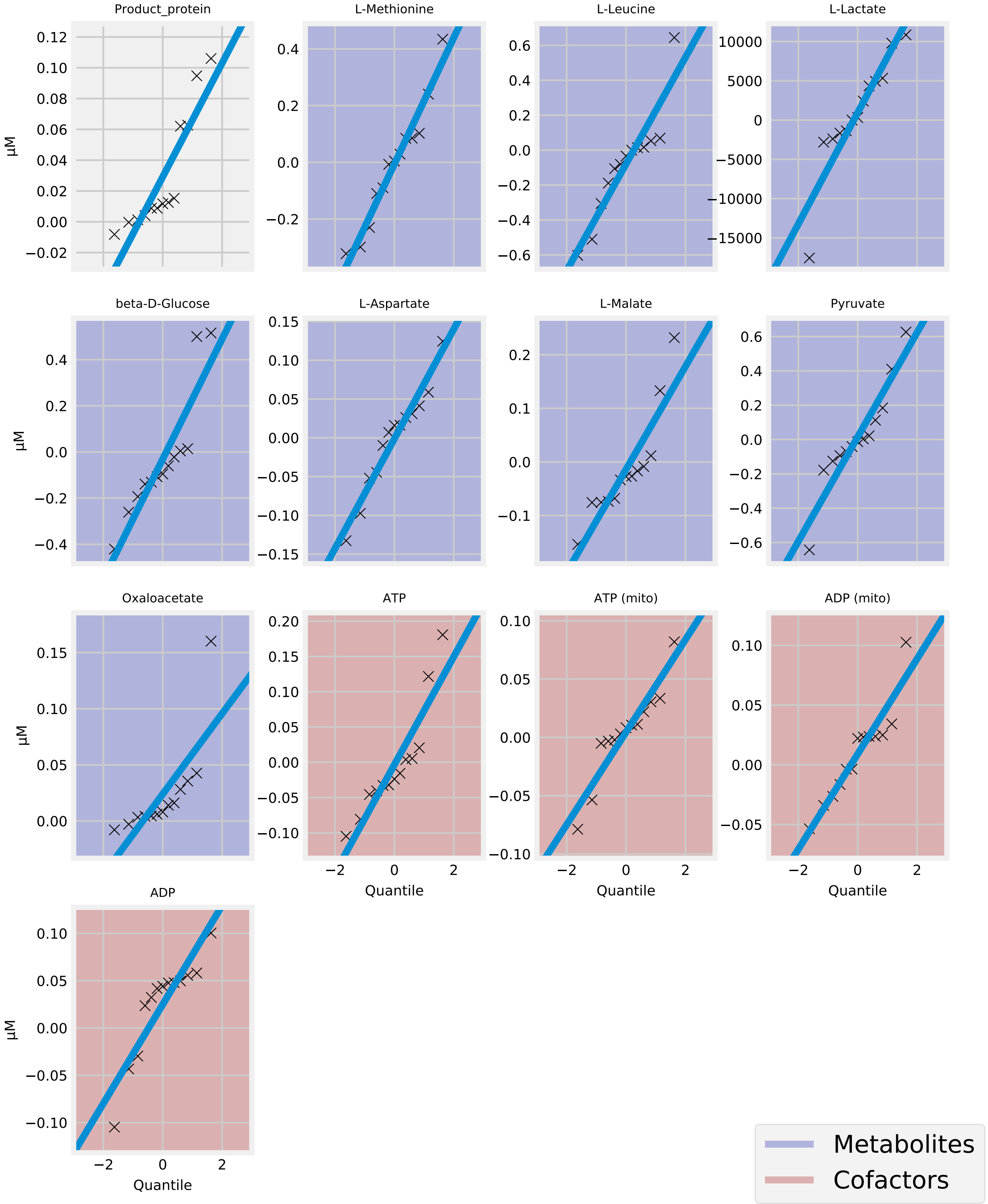


Figure 3.14: B4 normal probability plots. Residuals for each of the fitted quantities in Figure 3.13 is shown as a normal probability plot. Residuals that lie along a straight line are normally distributed. The noise level in problem B4 is normally distributed with time–varying variance.

المنارة للاستشارات

www.manaraa.com

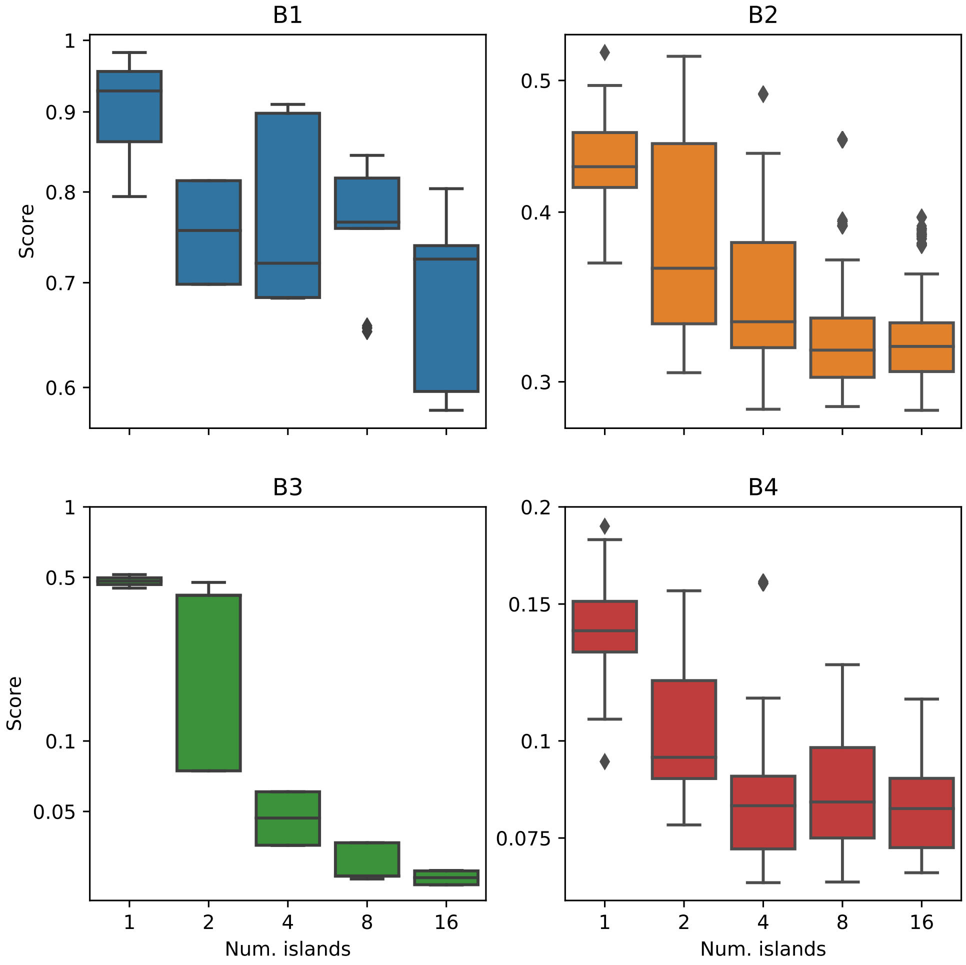


Figure 3.15: Objective function value for different numbers of islands. For a given amount of computing time, the island method provides a better fit by distributing fitting across different nodes and CPU cores. This diagram is intended to show the "payoff" versus number of threads used. Each island uses a single thread. It can be seen that after a certain number of threads, the benefit of parallelization diminishes. In this figure, all islands have an identical population size.

www.manaraa.com

Chapter 4

ACCELERATED SIMULATIONS VIA SPECIAL–PURPOSE HARDWARE

The contents of this chapter are based on a manuscript in preparation for submission to *PLoS Computational Biology.* The curriculum vitae at the end of this thesis lists all manuscripts and resp. statuses.

4.1 Introduction

Digital logic circuits have grown considerably in complexity since the inception of microprocessors. This growth was made possible in large part by technologies that automate the low–level record keeping, database management, routing, and placement of circuit components [55]. Digital system designers have long used hardware description languages such as VHDL (VHSIC Hardware Description Language) and Verilog to design the logic operations of digital circuits. However, many important computing problems can benefit from analog, rather than digital, circuit design. Important examples arise from the field of biomimicry, including neuromorphic chips, which emulate biological neurons [133], and cytomorphic chips, which emulate the behavior of cellular metabolic, signaling, and genetic pathways. This section of the thesis uses a previously described programmable cytomorphic chip capable of emulating a wide range of biological reaction networks much more efficiently than a digital computer [324, 325]. However, configuring the chip for a given network currently requires manual intervention, which is a tedious process that must be repeated for every new biological pathway. Whereas many design automation tools exist for designing digital hard-

المنارة للاستشارات
www.manaraa.com

ware, tools for the design and modeling of special–purpose analog circuits are comparatively rare. Circuit–simulation tools have been applied to neural biomimetic [166] and prosthetics [35, 128] devices and for simulating neuromorphic chips [209]. VLSI–inspired methods have been used in tools such as Cello [211] and iBioSim [180, 317] to verify genetic circuits, but there does not appear to be any existing system which transforms a high–level biological model (a chemical reaction network, in this work) into a low–level representation for running on programmable analog hardware. This section presents a cytomorphic compiler – a software tool which takes as input biological pathway models in Systems Biology Markup Language (SBML) format [131] and generates a cytomorphic chip configuration as output. Our compiler provides a bridge from existing systems biology standards to cytomorphic hardware, thereby increasing the versatility of special–purpose biomimetic hardware and bringing biomimetic computing closer to practical actualization.

Models of biological networks play important roles in our understanding of disease biology [227, 307], cancer [28], drug discovery [57], metabolic regulation [201], and many other subjects. However, simulation of large kinetic network models continues to be a major challenge, despite recent progress in high–performance simulation software [283, 118, 294]. The growth in size and complexity of biological pathway models has exceeded the growth of simulation hardware and software. In one study, a whole–cell *M. genitalium* model required 10 hours on a 128 node Linux cluster in order to simulate a single cell cycle [145]. Large–scale examples of kinetic simulations also arise in genome–scale kinetic models [279, 278]. Common simulation bottlenecks arise in parameter fitting and calibration of models, which require many simulations [146]. Thus, improvements in simulation performance are necessary for better and more comprehensive model fitting and to enable larger, more robust models.

In many real–world computing tasks, the relevant metric for performance is not the total computing power of the system, but rather the computations–per–watt. As demonstrated by the major cryptocurrency Bitcoin, special–purpose, highly efficient hardware has largely replaced graphics processing units (GPUs) as the main source of mining hardware. Special–purpose mining hardware had 40 times the performance–per–watt of an AMD 7970 GPU

المنارة للاستشارات

www.manaraa.com

in the year 2012, with further improvements occurring since that time. Cytomorphic chips operate at tens of milliwatts, yet in many cases still perform equal or faster simulations than desktop computers operating at tens of watts, representing a more than 1000–fold improvement in performance–per–watt. This efficiency improvement may be used to package more units onto a die, thereby allowing more simulations to run in parallel. Thus, cytomorphic chips represent a simulation technology with a potentially larger growth potential than conventional digital simulation tools.

The present work focuses on generalized, digitally–programmable cytomorphic hardware described previously [325]. The hardware is designed to solve systems of ordinary differential equations based on the observation that the concentration of a chemical species can be likened to electrical current (see [325] for details). Thus, all biological model variables, including species concentrations, parameters, and reaction rates are represented in the hardware by current values. However, a naïve approach at solving ODEs in this way will encounter the problem described in the next section.

4.2 The Divergence Problem

Consider the situation shown in Figure 4.1. This figure dipicts a kinase cascade the active form of kinase A, represented as A^P, which in–turn phosphorylates kinase B, with their corresponding rates of change:

$$\begin{aligned}
\frac{dA}{dt} &= -k_{fA}\frac{A}{1+A} + k_{rA}A^P \\
\frac{dA^P}{dt} &= k'_{fA}\frac{A}{1+A} - k'_{rA}A^P \\
\frac{dB}{dt} &= -k_{fB}A^P\frac{B}{1+B} + k_{rB}B^P \\
\frac{dB^P}{dt} &= k'_{fB}\frac{B}{1+B} - k'_{rB}B^P
\end{aligned} \tag{4.1}$$

Since the total amount of the kinase A is constant, the rate of change of $A + A^P$ should be zero. It can be seen from these equations that this will only occur if either α and β are

المنارة للاستشارات
www.manaraa.com

zero, or $\frac{A}{A^P} = \frac{\beta}{\alpha}$. However, the steady state values of A and A^P are already fixed by the equilibrium ratio of the first step in the cascade. Therefore, for general perturbations α and β the quantity $A + A^P$ will change over time, violating conservation laws. This example consists of a kinase cascade because it shows how the divergence problem clearly violates conservation laws, but this phenomenon actually applies to all networks that reach a steady state, regardless of whether conserved quantities exist in the network or not.

To address this problem, the cytomorphic chip is designed to operate on conserved quantities of the system. In the example in Figure 4.1, the conserved quantites are $A + A^P = A_{tot}$ and $B + B^P = B_{tot}$. The system can be described using a pair of differential equations corresponding to A^P and B^P:

$$\begin{aligned}
\frac{dA^P}{dt} &= k'_{fA}\frac{A_{tot} - A^P}{1 + (A_{tot} - A^P)} - k'_{rA}A^P \\
\frac{dB^P}{dt} &= k'_{fB}\frac{B_{tot} - B^P}{1 + (B_{tot} - B^P)} - k'_{rB}B^P \\
A &= A_{tot} - A^P \\
A &= B_{tot} - B^P
\end{aligned} \tag{4.2}$$

4.3 Chip Layout

This section briefly reviews the layout and specifications of the cytomorphic chip. A more complete description of the hardware can be found in [325].

Figure 4.2 depicts the layout of the cytomorphic chip. The chip is composed of 20 blocks, each designed to solve a single biochemical reaction of the form:

$$A + B \rightleftharpoons C + D$$

where the rate of this reaction is $k_f * A * B - kr * C * D$. In practice, modelers are accustomed to working with more complex reactions with lumped kinetic expressions such as Michaelis–Menten kinetics. However, physical processes at the molecular level invariably

المنارة للاستشارات

www.manaraa.com

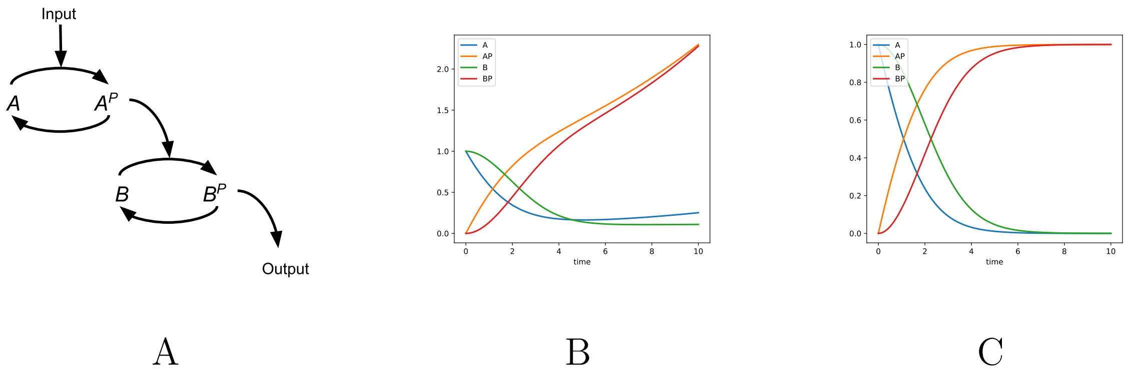


Figure 4.1: Demonstration of the divergence problem. In the phosphorylation cascade in (A), the quantities $A + A^P$ and $B + B^P$ should be constant in time. However, letting $k'_{fA} = k_{fA} + \alpha$ and $k'_{rA} = k_{rA} - \alpha$ results in the loss of this conservation relationship, as shown by the value of A^P in the numerical integration of this ODE system (B), which exceeds the total starting amount of $A + A^P = 1$. This phenomenon also applies to networks that do not have conserved quantities, as any steady–state value will tend to drift over time. Using the total quantity representation of eq (4.2), this problem can be eliminated (C).

المنارة للاستشارات

www.manaraa.com

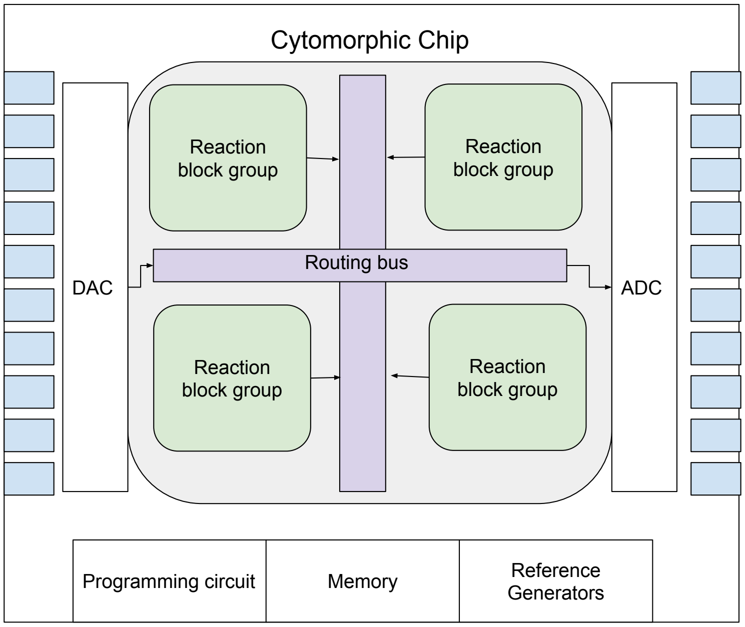


Figure 4.2: Layout of the cytomorphic chip. The chip is compsed of 4 groups of 5 blocks each of reaction units (see Figure 4.3). The routing bus allows the reaction units to be programmatically connect to DAC (for input) and ADCs (for output) and also allows for arbitrary connections between different reaction units.

fall into this binary mass–action category. In protein complex formation, subunits are added one–at–a–time, and in enzyme catalysis, substrates bind the enzyme in an intermediate state. Lumped kinetic expressions can be used with the hardware as shown later.

4.4 Methods

Figure 4.4 shows a high–level overview of the compiler. The compiler accepts as input a SBML [165] model parsed using JSBML [165, 82], or an Antimony file [281], which is a human–readable format directly interconvertible with SBML. SBML containing arbitrary

المنارة للاستشارات
www.manaraa.com

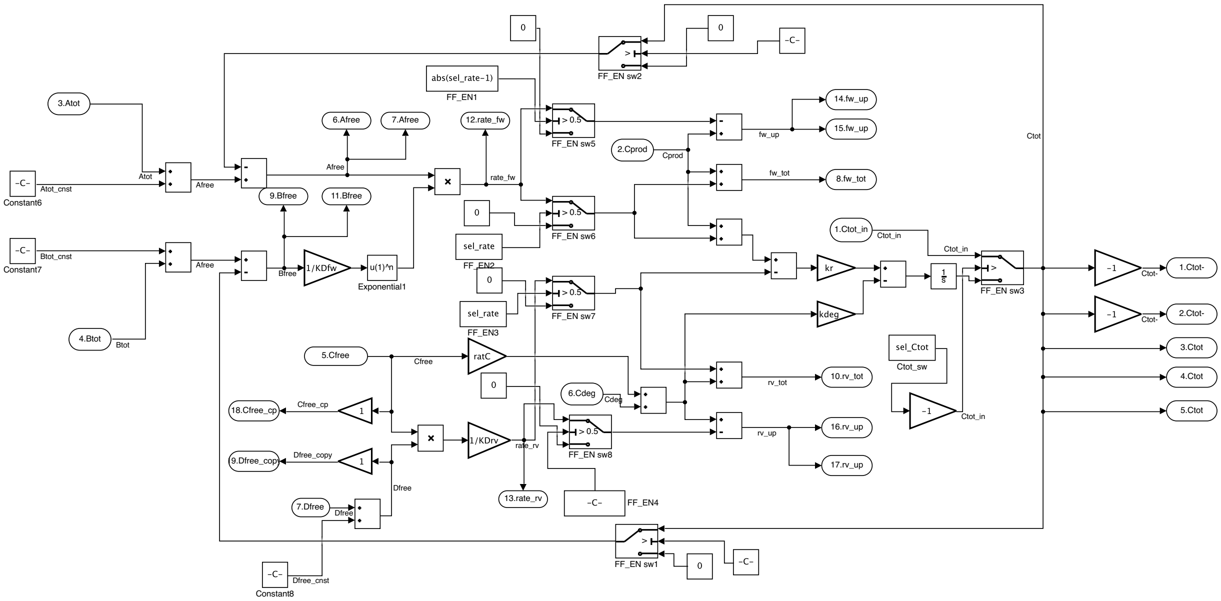


Figure 4.3: Block diagram of a reaction unit. Each reaction unit accepts as input A_{tot} and B_{tot} values for the two reactants and calculates the rate of change of the output C according to equation 4.3 (in the default switch configuration shown here). There are 7 switches that control the behavior of the block. The FF_EN switches are used in the fan–in configuration to enable computation of separate back–fluxes per branch. The section on building blocks describes this setup in more detail. The switches A_FB_EN and B_FB_EN are used to control substrate depletion. In the default configuration shown here, substrate depletion does occur. In the inverted configuration, the reaction does not consume A or B (biological examples of this include transcription and translation). Finally, the Ctot_in switch controls whether $C|tot$ is computed by the block (default) are specified by an external source (also used in the fan–in configuration). Each chip contains 20 such blocks.

المنارة للاستشارات
www.manaraa.com

rate laws cannot be run on the cytomorphic hardware. The compiler uses a method of "expanding" lumped kinetic expressions such as Michaelis–Menten, Botts–Morales, and repressor binding kinetics that allows networks using these formulations to be compiled onto the hardware.

The output of the cytomorphic compiler is two files: configuration of the shift registers (which specify the parameters of each block), and the SRAM (which specifies connections between blocks). Each of these file types is covered below.

Terminology & Validation Methods

Table 4.1 lists definitions for terminology used in this chapter.

In addition to producing programming files for the cytomorphic hardware, the cytomorphic compiler produces two other types of output that can be used to validate the compiler: (1) A Simulink model containing the blocks, parameters, and connections produced by the compiler, or (2) a differential equation system called a *block simulation* based on the block diagram of Figure 4.2B. Either of these outputs can be used to simulate the circuit behavior over time, similar to the SPICE analog circuit simulator [26] used in circuit design, except that the simulations are transfer function–based (i.e. they are based on the block diagram of Figure 4.2B, which uses gains, multipliers, and summation blocks for each stage instead of individual circuit components). In fact, these two formats are numerically equivalent, but serve different use cases. Simulink diagrams are used to visualize the block wiring, whereas block simulations are used to plot and compare compiler output in Jupyter notebooks. In the case of mass–action networks, the block simulation should correspond exactly to the SBML simulation. However, the underlying differential equations in the block simulation are based on total quantities, whereas SBML uses free quantities.

For validating the output of the compiler, block simulations are a primary source of quantitative comparison. The block simulation should match the SBML simulation output exactly for mass–action networks, and approximately for lumped kinetics expressions. Hardware simulations are also shown where available. However, these cannot be used for

www.manaraa.com

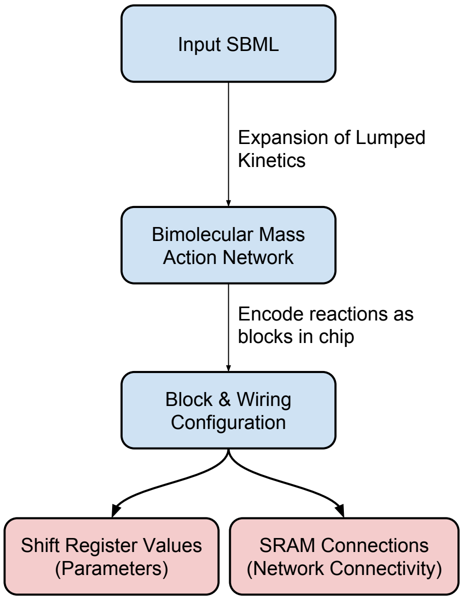

Input SBML
Expansion of Lumped Kinetics
Bimolecular Mass Action Network
Encode reactions as blocks in chip
Block & Wiring Configuration
Shift Register Values (Parameters)
SRAM Connections (Network Connectivity)

Figure 4.4: A flow diagram for the cytomorphic compiler. The compiler processes the input SBML model to “expand” (see below) lumped kinetic expressions into constituent bimolecular elementary processes. Elementary reactions are then mapped to blocks on the chip (sometimes to multiple blocks, as in fan–out reactions described below). Each block is assigned parameter values based on the forward and reverse rate constants of its respective reaction, and potentially degradation of the product. Blocks are connected together based on the topology of the reaction network, but care must be taken to maintain a single “total” value for each species, as described in the “Network Motifs” section. The final output of the compiler is a configuration for the shift registers (which store parameter values) for all used blocks and SRAM (which connects block input and output ports).

المنارة للاستشارات
www.manaraa.com

Table 4.1: **Cytomorphic terminology used in this chapter.**

Term	**Description**
Cytomorphic Chip	A single chip with 20 reaction blocks.
Antimony	A human–readable and writable representation of SBML.
Block simulation	A circuit–level simulation (performed on a computer) of a specific configuration of the cytomorphic chip (including parameters and connections). This simulation is based on the ODE model of Figure 4.3, which shows the transfer function for every component in the block. A block simulation can be performed using either Simulink or libroadrunner [283]. In either case, the files to run the block simulation are generated by the compiler.
Kinetic expansion	Classical enzyme kinetics like the Michaelis–Menten rate law are based on lumped processes. A Michaelis–Menten process represents substrate binding and catalysis in one step but these are mechanistically two separate processes. The cytomorphic compiler breaks these lumped expressions down into their constituent components, using lumped constants such as the Michaelis constant K_M to determine rate constants.
Archetype	We use this term to denote the canonical form of a rate law expression such as the Michaelis–Menten formula $\frac{V_{max}S}{K_M+S}$. See the section “Matching Algorithm for Lumped Kinetics” below.

المنارة للاستشارات
www.manaraa.com

quantitative comparison due to manufacturing variations and the fact that the block simulation does not take into account the component–level capacitance of the block integrator, or the analog–to–digital converter (ADC) clockspeed of the cytomorphic hardware. Nevertheless, the shapes of the waveforms match closely, as can be seen from the repressilator example below.

4.5 Shift Registers & Block Parameters

For the default configuration of the `FF_EN_sw` switches in Figure 4.2B, it is apparent that the rate of production of the block's main product, C, is given by:

$$\begin{aligned}\frac{dC}{dt} &= k_r \left((A_{tot} - C_{tot} \cdot A_{FB\_EN}) \left(\frac{B_{tot} - C_{tot} \cdot B_{FB\_EN}}{KD_{fw}} \right)^n + C_{prod} \right) \\ &- \left(\frac{C_{free} D_{free}}{KD_{rv}} \right) - K_{deg} \left(C_{free}(ratC) + C_{deg} \right)\end{aligned} \tag{4.3}$$

The forward and reverse rates for each block are determined by a combination of k_r and either KD_{fw} or KD_{rv}. Examining the equation shows that the forward and reverse rate constants are given by $\frac{k_r}{KD_{fw}}$ and $\frac{k_r}{KD_{rv}}$ respectively. Thus, the forward and reverse dissociation constants can be used to tune the relative forward and reverse rates, whereas the overall reaction rate of all blocks in the network can be tuned by changing the k_r value. Also, any block that produces a species can also serve as a degradation reaction by utilizing $ratC$.

4.6 SRAM & Network Motifs

The chip's SRAM is used to program connections between the input and output ports of the blocks. The input and output ports for each block are shown in Figure 4.5.

The method for connecting input and output ports is best illustrated with a series of examples. This section shows a series of motifs that, taken together, form a basis for most larger reaction networks. For cytomorphic hardware, which relies on total quantities, the two most important and most challenging cases are "fan–in" and "fan–out" configurations,

المنارة للاستشارات

www.manaraa.com

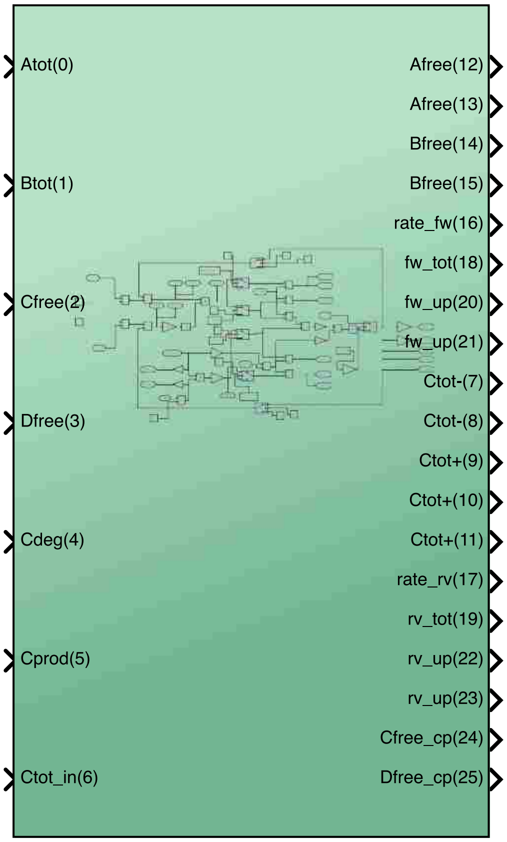


Figure 4.5: Input and output ports for the cytomorphic block. The two main inputs A_{tot} and B_{tot} are used to compute the forward rate. Internally, the block subtracts its own C_{tot} value from A_{tot} and B_{tot} to compute A_{free} and B_{free} (see Figure 4.2B). The chip's main output is C_{tot}. If there is another reaction that consumes C, then C_{tot} should be connected to the A_{tot} or B_{tot} input of the consumer block. Otherwise, C_{tot} should be connected to the block's own C_{free} input to allow calculating the reverse rate. If the reaction is reversible, D_{free} may also receive input from another block or be simply wired to unity in the case of a single product C. Other ports describing the blocks forward and reverse rates are used in in certain motifs described below. The chip has 2 copies of `Afree`, `Bfree`, `fw_up`, `rv_up`, and 5 copies of `Ctot` (2 negative and 3 positive). In addition, the chip also copies the values of its C_{free} and D_{free} inputs to the `Cfree_cp` and `Dfree_cp` output ports respectively. These copies are used to route the input values to additional blocks in fan–in configurations (see below).

المنارة للاستشارات
www.manaraa.com

where a single species is produced or consumed by multiple reactions respectively. These cases require careful accounting to ensure that the total quantities are properly handled. In what follows, species names such as S, T, etc. are used to distinguish species in the network from the port names on the block (A, B, C, and D). Figs 4.6–4.9 show the different motifs used to validate the compiler.

4.7 Lumped Kinetics

The term “lumped kinetics,” as used here, refers to any kind of process in a model that represents multiple elementary steps, where an elementary step is defined as a bimolecular mass action reaction (e.g. substrate binding / unbinding, or the catalytic step in enzyme catalysis). A common instance of this is enzyme kinetics: $E+S \rightleftharpoons ES \longrightarrow E+P$, which represents an enzyme E that converts substrate E into product P. These reactions occur at the following rates:

$$\begin{aligned} E+S &\rightleftharpoons ES, & \quad & k_f E \cdot S - k_r ES \\ ES &\longrightarrow E+P, & \quad & k_{cat} ES \end{aligned} \tag{4.4}$$

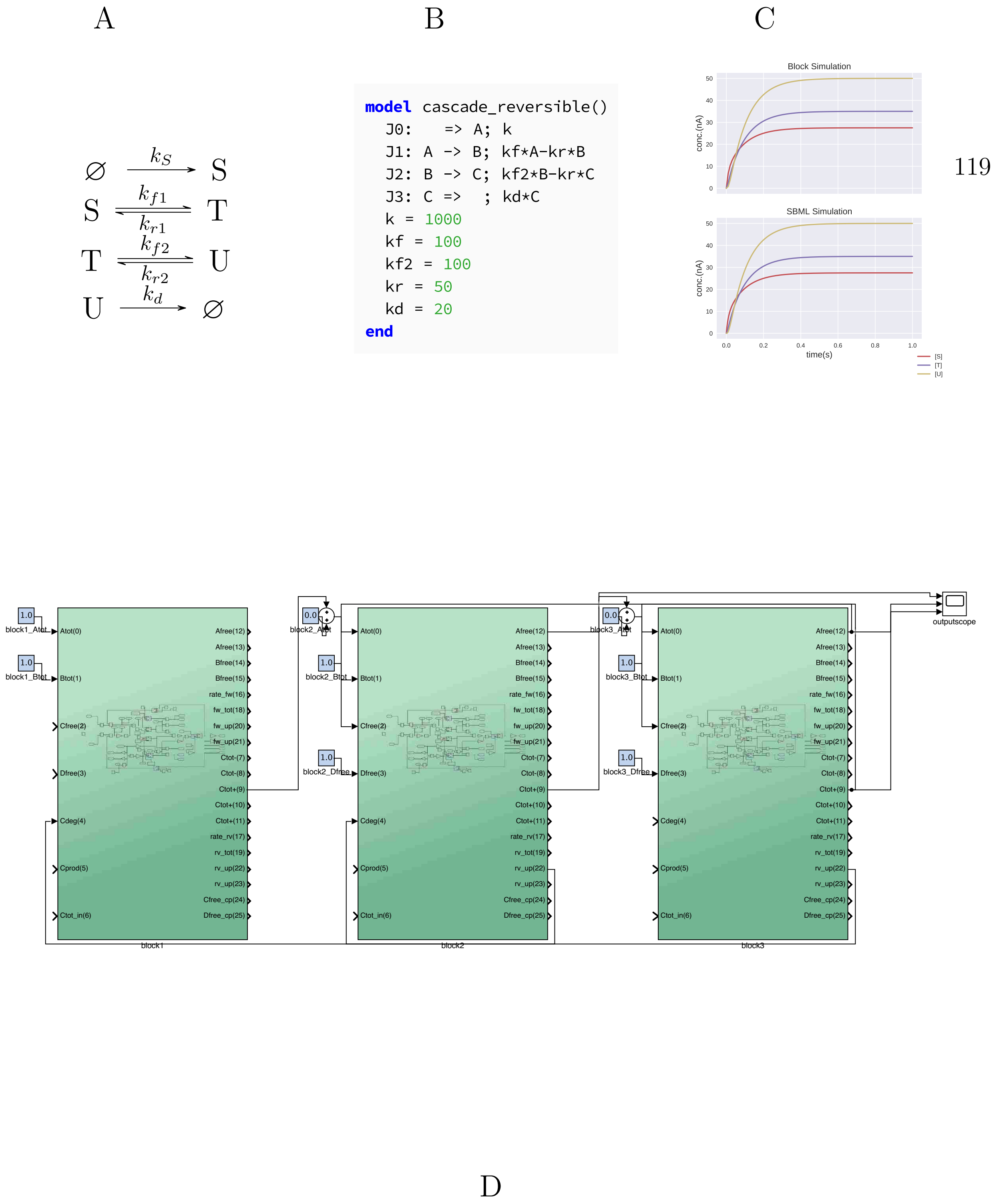
The individual forward and reverse rate constants of the binding step are difficult to measure directly, so these two processes are usually condensed into a single process by assuming either rapid equilibrium of the binding process (which leads to the well–known Michaelis–Menten kinetics [200]) or by assuming the enzyme–substrate complex is at steady–state (Briggs–Haldane kinetics [54]). In either case, the rate law for the resulting lumped process can be expressed as:

$$k_{cat} E \frac{S}{K_M + S}$$

where $K_M = \frac{k_r + k_{cat}}{k_f}$ for Briggs–Haldane kinetics or k_r/k_f for Michaelis–Menten kinetics.

Our general approach to simulating these lumped expressions on the cytomorphic chip is to break them down into the constituent steps of eq 4.4. However, this re-creates the

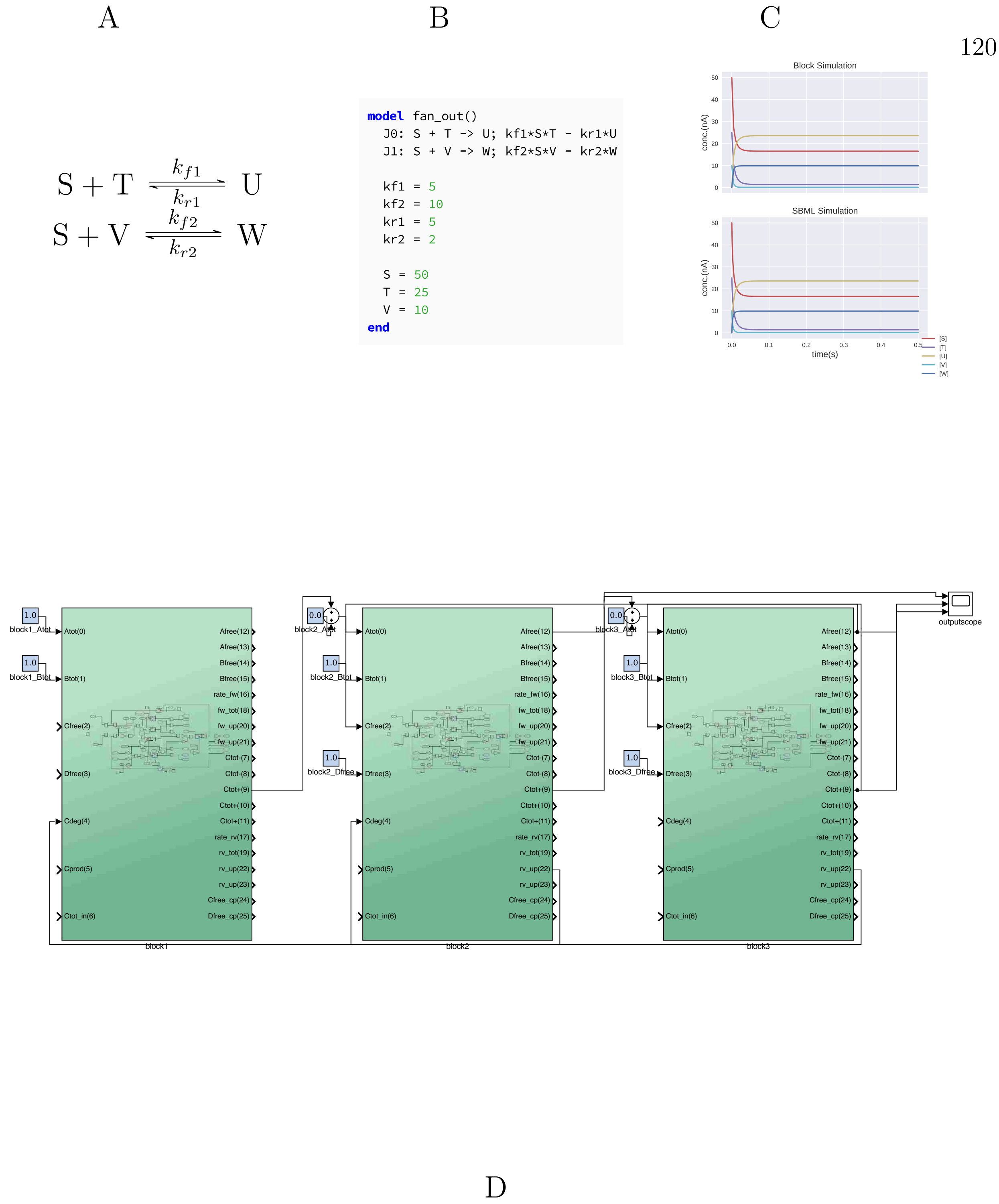
المنارة للاستشارات
www.manaraa.com


A
B
C
D
$\varnothing \xrightarrow{k_S} S$
$S \underset{k_{r1}}{\overset{k_{f1}}{\rightleftharpoons}} T$
$T \underset{k_{r2}}{\overset{k_{f2}}{\rightleftharpoons}} U$
$U \xrightarrow{k_d} \varnothing$
Block Simulation
SBML Simulation
conc.(nA)
time(s)
[S]
[T]
[U]
block1_Atot
block1_Btot
block2_Atot
block2_Btot
block2_Dfree
block3_Atot
block3_Btot
block3_Dfree
block1
block2
block3
outputscope

```
model cascade_reversible()
  J0:    => A; k
  J1: A -> B; kf*A-kr*B
  J2: B -> C; kf2*B-kr*C
  J3: C =>   ; kd*C
  k = 1000
  kf = 100
  kf2 = 100
  kr = 50
  kd = 20
end
```

Figure 4.6: A three–step feed–forward network. This network is comprised of a linear chain of unimolecular mass–action processes (A). To convert this network to a wiring, blocks 1–3 are designated as the main producer of species S, T, and U respectively. Block 1 produces S, hence its main output port C_{tot} is connected to the A_{tot} input of block 2 and so on. This signal is summed with the initial value of T (if non–zero). Since the cytomorphic chip uses currents for computation, summing signals is achieved simply by connecting multiple signals to the same input port. The last block in the chain, which produces U, also serves as a degradation reaction U $\xrightarrow{k_d}$ $\varnothing$. The block has its `ratC` parameter set to the degradation rate k_d. Additionally, the amount of U degraded must also be subtracted from the total values of S and T. The `rv_up` port computes the total loss in C for each block and is propagated to the block immediately upstream by connecting to the `Cdeg` port. The Antimony/SBML model for this motif (B) is converted to a block simulation that gives identical output to the SBML simulation (C). Since this is a mass–action network, numerical differences can be made arbitrarily small by adjusting integrator tolerances. All connections created by the compiler are shown in the wiring diagram (D).

المنارة للاستشارات
www.manaraa.com



$$\mathrm{S}+\mathrm{T} \underset{k_{r1}}{\overset{k_{f1}}{\rightleftharpoons}} \mathrm{U}$$
$$\mathrm{S}+\mathrm{V} \underset{k_{r2}}{\overset{k_{f2}}{\rightleftharpoons}} \mathrm{W}$$

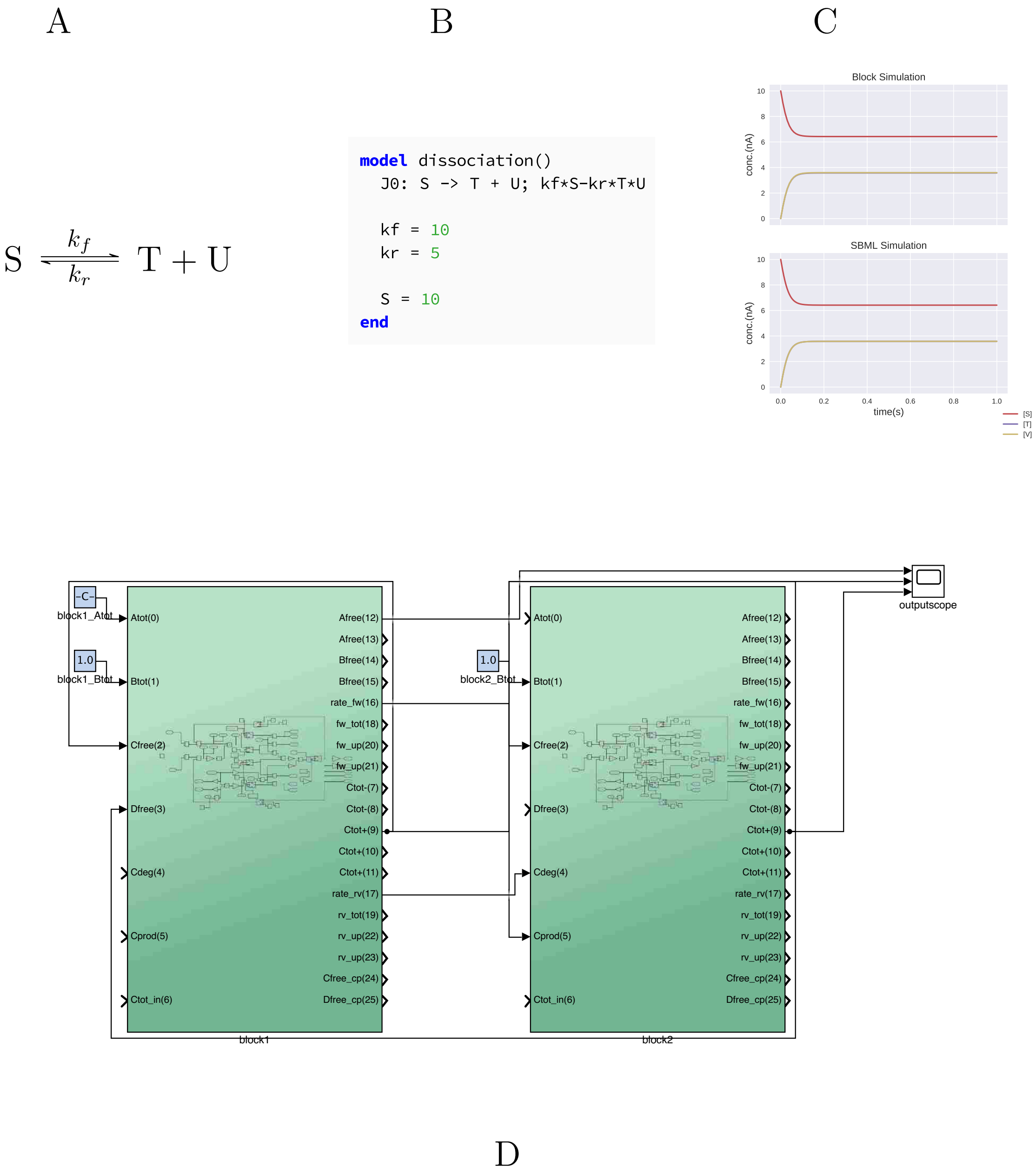
```
model fan_out()
  J0: S + T -> U; kf1*S*T - kr1*U
  J1: S + V -> W; kf2*S*V - kr2*W

  kf1 = 5
  kf2 = 10
  kr1 = 5
  kr2 = 2

  S = 50
  T = 25
  V = 10
end
```

Figure 4.7: A "fan–out" reaction occurs when multiple reactions consume the same reactant (S in this case). In such a case, the blocks are termed a *consumer group*. One of these blocks will compute S_{free} by subtracting its own C_{tot} value. This S_{free} value is then connected to the A_{tot} input for the second block. In addition, the second block also subtracts its own C_{tot} value from the input value of S (via routing from one of the inverted output ports for C_{tot}). Thus, the computed value S_{free} is equal to S_{tot} minus U_{tot} (due to the feedback within the first block) and minus V_{tot} (due to the extra C_{tot} connection).

المنارة للاستشارات
www.manaraa.com



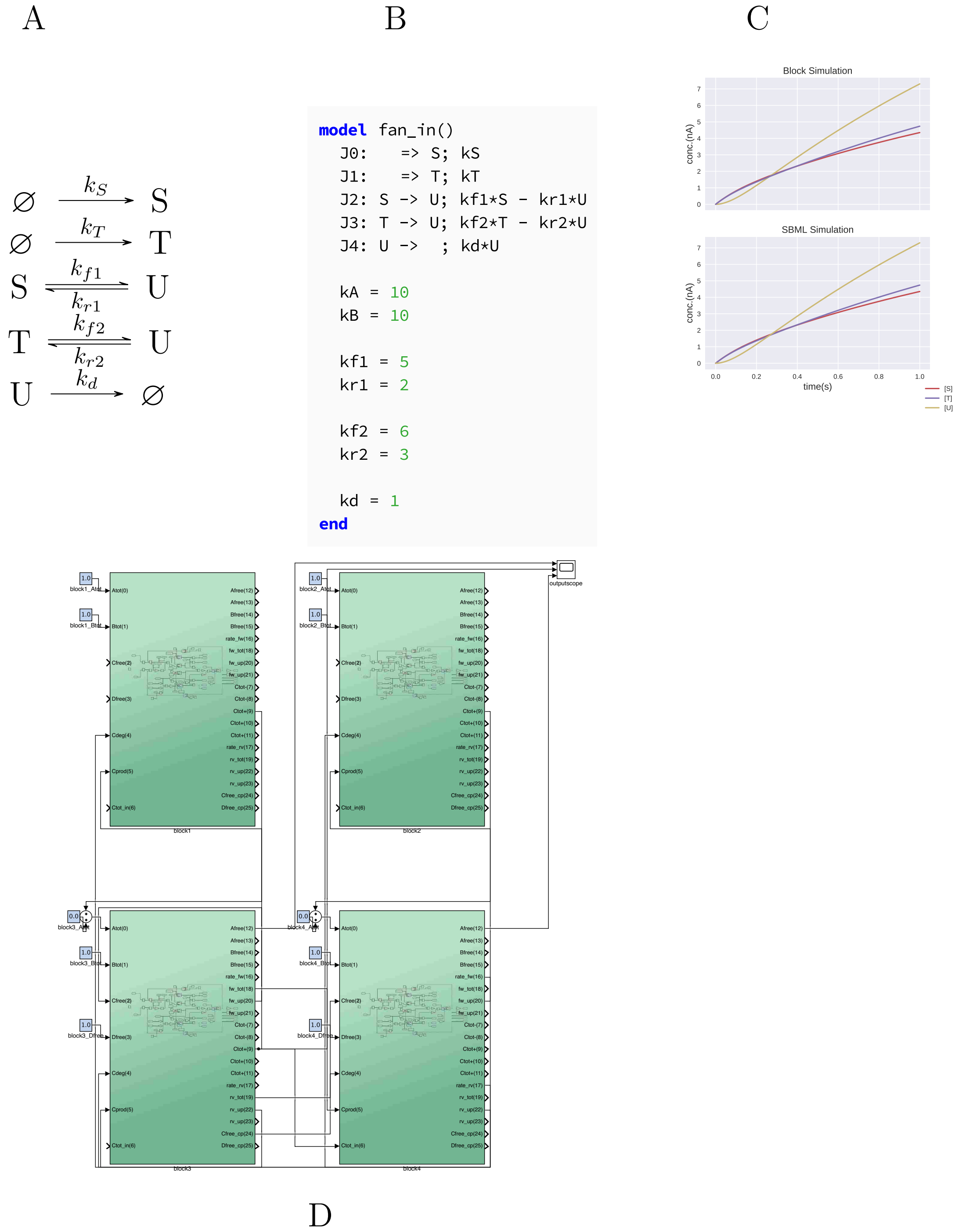
```
model dissociation()
  J0: S -> T + U; kf*S-kr*T*U

  kf = 10
  kr = 5

  S = 10
end
```

Figure 4.8: A dissociation reaction. A single block only contains a single integrator circuit, which is used to compute its main output C_{tot}. When multiple outputs are present, they can, in general, possess different degradation rates and be produced and consumed by different sets of reactions. In order to account for this, the cytomorphic compiler creates two blocks for each dissociation reaction. The first block computes T_{tot} and sends its forward and reverse rates to the `Cprod` and `Cdeg` ports of the second block, which computes U_{tot}. The second block uses the forward and reverse rates to compute the change in U_{tot} according to eq. 4.3 with $A_{tot} = B_{tot} = 0$. The production of U_{tot} will thus be the same as T_{tot} except each block may have a different degradation constant `ratC` and may be independently connected to

www.manaraa.com



```
model fan_in()
  J0:    => S; kS
  J1:    => T; kT
  J2: S -> U; kf1*S - kr1*U
  J3: T -> U; kf2*T - kr2*U
  J4: U ->   ; kd*U

  kA  = 10
  kB  = 10

  kf1  = 5
  kr1  = 2

  kf2  = 6
  kr2  = 3

  kd  = 1
end
```

Figure 4.9: A “fan–in” reaction occurs when multiple reactions produce the same species U. In such a case, the blocks are called a *producer group*. Since the cytomorphic chip uses “total” quantities for computations, these different sources for U need to be summed together to provide a single value for U_{tot}. This is accomplished as follows: (1) designate a single block as the “main” producer of U. Other blocks that produce U will send their forward and reverse rates `rate_fw` and `rate_rv` to the main block’s `Cprod` and `Cdeg` ports resp. In turn, the main block sends its own computed value of U_{tot} as well as the total forward and reverse rates for U to all other blocks in the producer group via `fw_tot` and `rv_tot`. Finally, when U is removed from the system, the total values of its upstream nodes S and T must be adjusted accordingly. Internally, all non–main blocks in the producer group compute the total production and consumption of U and subtract their individual contributions to this amount (via inverting the `FF_EN_sw` switches in Figure 4.2B). This left–over amount is the amount by which the total value of the upstream nodes S and T changes as a result of external sources of U. This scheme, while complicated, allows total quantities for all species to be computed.

www.manaraa.com

problem of determining the forward and reverse rate constants k_f and k_r respectively. From the lumped expression, we can determine k_{cat} and K_M. However, we need one additional constraint to specify both k_f and k_r. This constraint comes from the upper limit of the chip's simulation speed.

Consider the block diagram of a reaction unit containing a negitive feedback loop around the part of the circuit that processes A as highlighted in Figure 4.10. In designing electronic amplifiers, it is common to account for the so–called *phase margin*. In an amplifier circuit as well as in the highlighted feedback loop, there exists the possibility that the output signal can be 180° out–of–phase with the input. Since the feedback is negative, the signal will be inverted and cause constructive interference with the input, leading to instability. Furthermore, this feedback loop also possesses a parasitic pole due to the current mirror that produces the A_{free} signal. Taken together, these conditions lead to the stability rule [323]:

$$\frac{B_{tot}}{KD_{fw}}\frac{r}{C} < \frac{A_{free}}{C_{par}}$$

where C_{par} is the parasitic capacitance at the gate node of the A_{free} current mirror, $C = 0.1\mu F$ is the capacitance of the integrator capacitor, and r is the overall rate of the block (used as a scaling factor for both the forward and reverse rates). In the preceding expression, we assumed that $A_{free} < B_{tot}$. The roles of A and B can be reversed if this is not the case. This equation can be simplified to:

$$k_f < \frac{A_{free}}{B_{tot}}\frac{C}{C_{par}} = \rho \tag{4.5}$$

This gives an upper bound for the value of k_f based on the global value $\frac{C}{C_{par}}$ and the local value $\frac{A_{free}}{B_{tot}}$ which depends on the reaction and simulation conditions. These two values can be condensed into a single constant ρ, called as the margin, which in general varies per reaction. Several more examples of lumped expressions are considered in this chapter that expand into

www.manaraa.com

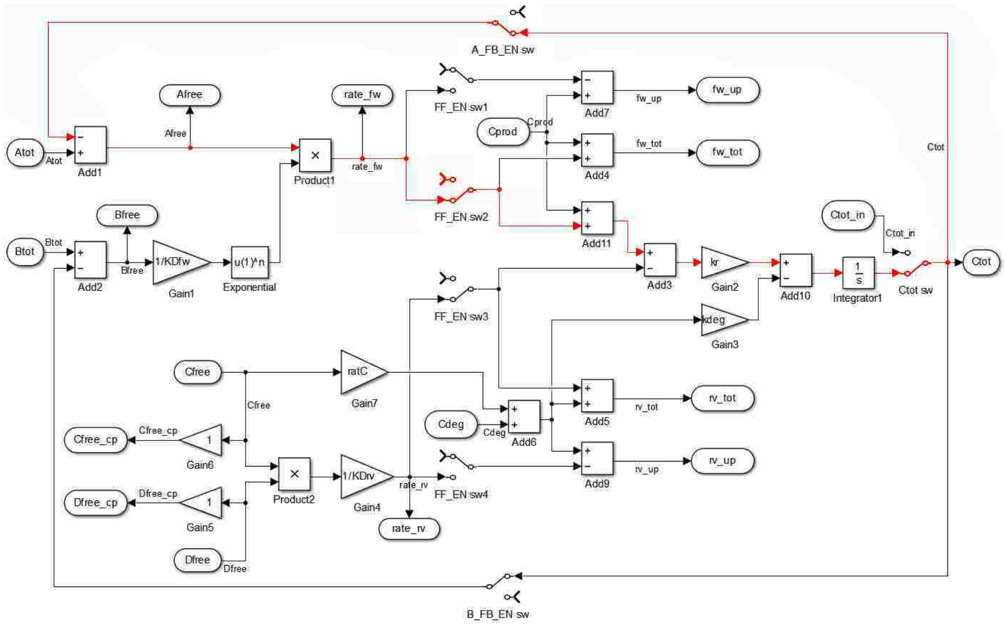


Figure 4.10: Negative feedback around the circuit for A.

elementary processes that occur at rate ρ. In practice, we can assume a reasonable lower bound for ρ and refine the estimate after simulating the network. If we assume $C_{par} = 1pF$, this gives an upper–bound of $10^5 \frac{A_{free}}{B_{tot}}$. The value of $\frac{A_{free}}{B_{tot}}$ is model and simulation–dependent, but by assuming a reasonable upper bound of 100, we obtain $\rho = 1000$. This value can be used to run digital simulations for a given hardware configuration which in turn can be used to refine the value ρ.

Returning to enzyme kinetics, the higher the value we choose for k_f (and hence k_r), the more rapid the enzyme–substrate binding. Since Michaelis–Menten kinetics are derived based on an equilibrium assumption, a larger k_f will tend to make this assumption more valid. Therefore, choosing $k_f \approx \min\left(\frac{A_{free}}{B_{tot}}\right)\frac{C}{C_{par}} = \rho$ yields the best approximation to Michaelis–Menten kinetics without causing instability.

Figure 4.11 shows the result of plotting the dynamics of the elementary binding / catalysis network versus the original Michaelis–Menten lumped single–process network for various

المنارة للاستشارات
www.manaraa.com

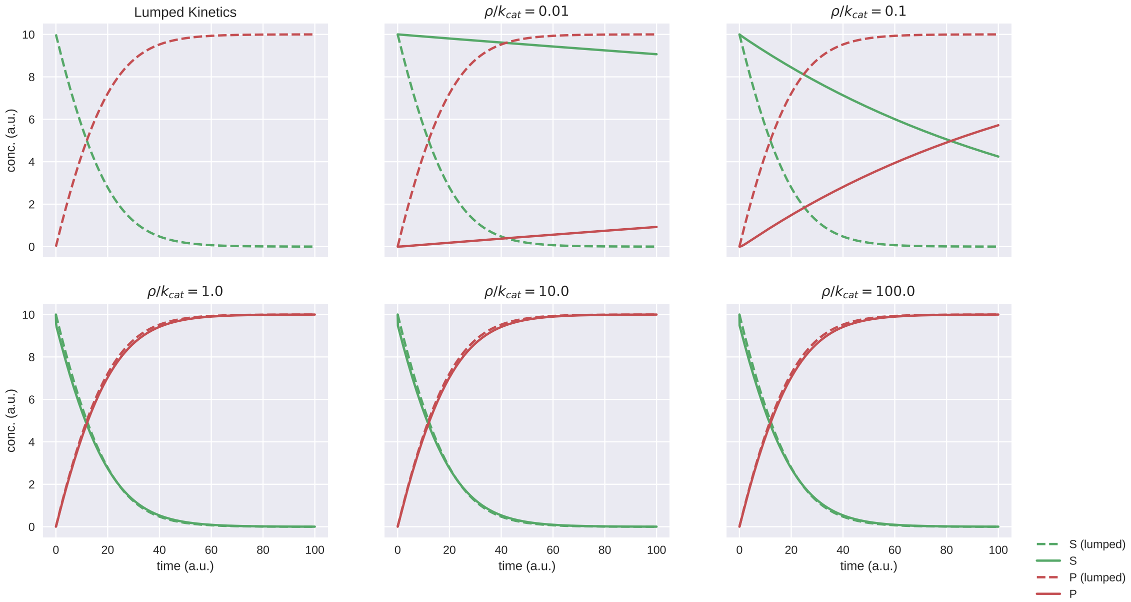


Figure 4.11: Different margin values and their respective simulations. In these simulations, $E = 1$, $S_{initial} = 10$, $k_{cat} = 1$, and $K_M = 10$. For the elementary two–process network, the reverse binding rate is calculated automatically by the compiler using the supplied value of ρ as $k_r = \rho K_M - k_{cat}$ if $\frac{k_{cat}}{k_r} \ll 1$ or $k_r = \rho K_M$ otherwise.

values of ρ.

It is worth observing that, from a design standpoint, our compiler uses the two–step network to “approximate” models with lumped processes such as Michaelis–Menten kinetics. However, from a mechanistic standpoint, Michaelis–Menten kinetics represents an approximation of the corresponding physical two–step process. In effect, this line of reasoning has taken us full-circle from a mechanistic representation to a lumped process and back again. However, for the purpose of practical modeling, this is a necessary detour, since it is feasible to measure the K_M and V_{max} values for enzymes, but in general it is not possible to measure

المنارة للاستشارات
www.manaraa.com

binding apart from catalysis (i.e. the parameters k_f, k_r, and k_{cat}).

The meaning of this observation is that while elementary, mass–action processes are biologically accurate, they are not feasible from a modeling standpoint. Thus, modelers should continue to use kinetics that can be parameterized with quantifiable parameters, while specialized hardware should continue to use elementary processes that are conducive to efficient implementation. Breaking–down these high–level expressions into low–level expressions (hereafter referred to as “expansion”) is one of the main functions of the cytomorphic compiler.

A limitation of this “expansion” method is that when the substrate is not in excess of the enzyme (i.e. $E \ll S$ does not hold), then there is significant deviation between the two–step mechanistic process and the idealized Michaelis–Menten approximation (the mechanistic process will tend to lag behind the lumped process), regardless of the value of ρ. However, Michaelis–Menten kinetics also reilies on the assumption that $E \ll S$, and thus would be a physically inaccurate modeling assumption in this case.

Matching Algorithm for Lumped Kinetics

In order to successfully expand lumped kinetic expressions into constituent components, it is necessary to (1) identify, from the rate law, what type of lumped expression is represented, and (2), obtain the values of all lumped constants and use these to compute the individual rate constants.

One approach to solving (1) would be to simpify and canonicalize the rate law expression and compare this simplified version with each known kinetic expression on a tree–basis (canonical expressions like the Michaelis–Menten formula $\frac{V_{max}S}{K_M+S}$ are referred to as “archetypes”). However, this approach is sensitive to different factorizations of the expression and requires that all archetypes also be in canonical form. Instead, the compiler uses an algorithm for determining the equivalence of expressions based on hash–coding [184]. This algorithm is computationally efficient and maps algebraic expressions into finite rings.

In the compiler implementation, additional simplifications are used to further increase

المنارة للاستشارات

www.manaraa.com

the performance. The algorithm is split into two stages. The first stage assigns numerical values to all substrates and products in an expression, and unity to all other symbols. In principle, this introduces ambiguity. However, the compiler requires classifying biochemical rate laws, and no two rate laws are identical up to arbitrary constants. Thus, it is safe to discard constant values at this stage. The output of the first stage is the matched archetype along with the positions of the substrates and products in the expression. In the second stage, rate laws are matched constants one–by–one. This allows us, for example, to match the Michaelis constant K_M. For each constant, a numeric value is assigned, setting all other constants to unity. The assigned constant, as well as all substrates and products, must be relatively prime. If the computed hash code matches the reference expression, the value of the SBML parameter is assigned to the current constant.

The procedure for the algorithm is described in pseudocode in Figure 4.12.

4.8 Compiling Gene Regulatory Kinetics

Another major type of lumped kinetics occurs in models of gene regulatory networks. Consider the LacI repressor, which controls expression of the *lac* operon (*lacO*) in bacteria. The LacI repressor is a homotetramer, but might be better described as a dimer of dimers. Each dimer subunit contains a DNA–binding site for *lacO*. The repressor binds to *lacO* as a two–step processes. Binding of allolactose to LacI causes the repressor to enter an inactive state P (protruded) with decreased overall affinity, releasing the operator site. Using lumped kinetics, the transcription rate of the operator is given by [88, 260]:

$$\nu = \alpha_0 + \alpha \frac{K_m^2}{K_m^2 + R^2}$$

where ν is the transcription rate, α is an experimentally determined rate constant, K_m is the equivalent equilibrium constant of the two binding steps, R is the concentration of the active repressor, and α_0 is the basal level of transcription under fully repressed conditions (the “leakage” rate).

www.manaraa.com

```
# stage 1: match the archetype
given an input expression e
for a in archetypes
    for symbol s in e
        # map all symbols to a prime number if it is a substrate/product, 1 otherwise
        s -> prime() if s is in [substrates,products,modifiers]; 1 otherwise

    if hash(e,s) = hash(a,s)
        # if the hash matches the archetype, return the archetype and matched symbols
        return a, symbols in e

# stage 2: map all constants
given an input expression e, archetype a, and symbols from stage 1
for symbol s in a
    if s is not mapped
        # map the constants to primes one-at-a-time
        s -> prime()

        for SBML param p in e
            # get a trial parameter and map it to the same value as s
            p -> value(s)
            if hash(e,s+p) = hash(a,s)
                assign SBML initial value of p to the symbol s
```

Figure 4.12: Pseudocode for the Martin matching algorithm.

المنارة للاستشارات
www.manaraa.com

In order to expand this process to a simulatable form, we could decompose binding into a two–step process. However, the cytomorphic chip provides a Hill function that allows this two–step process to be modeled as a single step (labelled n in Figure 4.2). Using the Hill function, we could write the overall binding process as:

$$R + O \rightleftharpoons B, \qquad k_f^2 R^2 O - k_r B$$

where R is the active repressor, O is the operator site, and B is the bound (repressed) complex. Transcription can then be modeled as a simple first–order process without substrate depletion (both the `A_FB_EN` and `B_FB_EN` in 4.2 should be off). This expanded model exhibits a time delay proportional to $\tau = kr + (k_f \cdot R)^2$ compared to the lumped expression, which assumes rapid equilibrium of the binding process. Our approach to minimizing this discrepancy is the same as in the enzyme kinetics case – we maximize the forward rate constant k_f subject to the margin ρ defined in eq 4.5. The expanded model can reproduce the dynamics of complex networks to a high degree of accuracy, as shown below.

In a highly influential study in 2000, Elowitz *et al.* showed that a genetic oscillator (the "repressilator") can be constructed from three genetic repressors [88]. To validate the cytomorphic compiler's ability to translate repressor kinetics, a dynamical model of the repressilator from the BioModels database was used as a starting point [164, 89]. This model contains a total of 12 reactions, half of which are degradation reactions. To reduce the number of blocks required to encode the model, the degradation reactions were condensed into the production rate laws for the three genes and proteins in the system:

$$\varnothing \longrightarrow X, \qquad \alpha_0 + \alpha \frac{K_m^2}{K_m^2 + R^2} - k_{d1} X \tag{4.6}$$

$$\varnothing \longrightarrow P_X, \qquad k_t X - k_{d2} P_X \tag{4.7}$$

and similarly for Y and Z. The parameter values were adjusted to comply with the acceptable current ranges for the chip.

www.manaraa.com

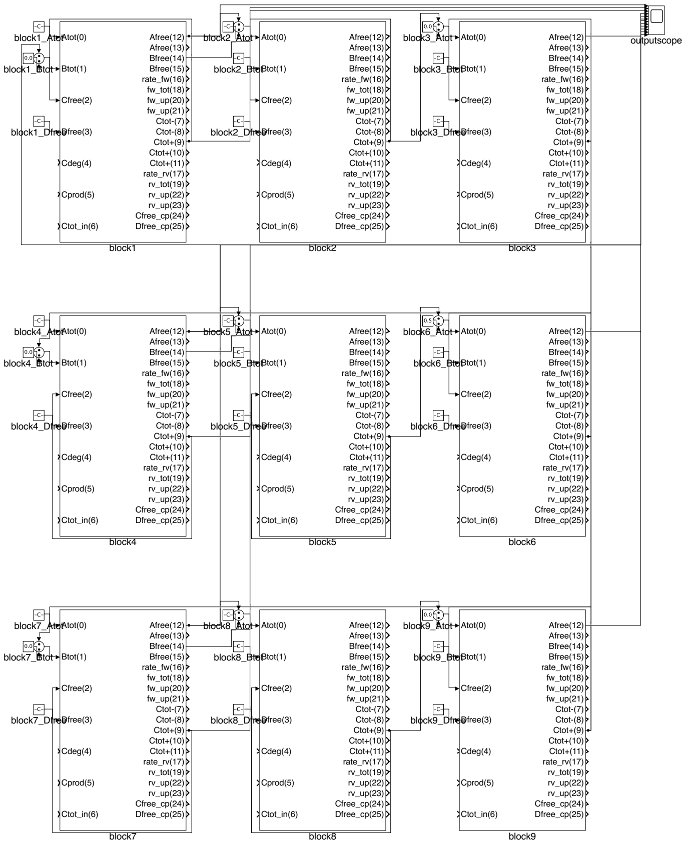


Figure 4.13: Wiring diagram of the repressilator model. To validate the cytomorphic compiler output for repressor kinetics, an SBML model containing six transcription / translation / degradation reactions was used (see eq 4.7). The compiler transformed this SBML model into a block configuration containing nine blocks (A). Each transcription reaction is expanded to repressor binding (first column) and transcription (second column), whereas each translation / degradation reaction is represented by a single block (third column). A digital simulation of this block configuration is shown in Figure 4.14B.

www.manaraa.com

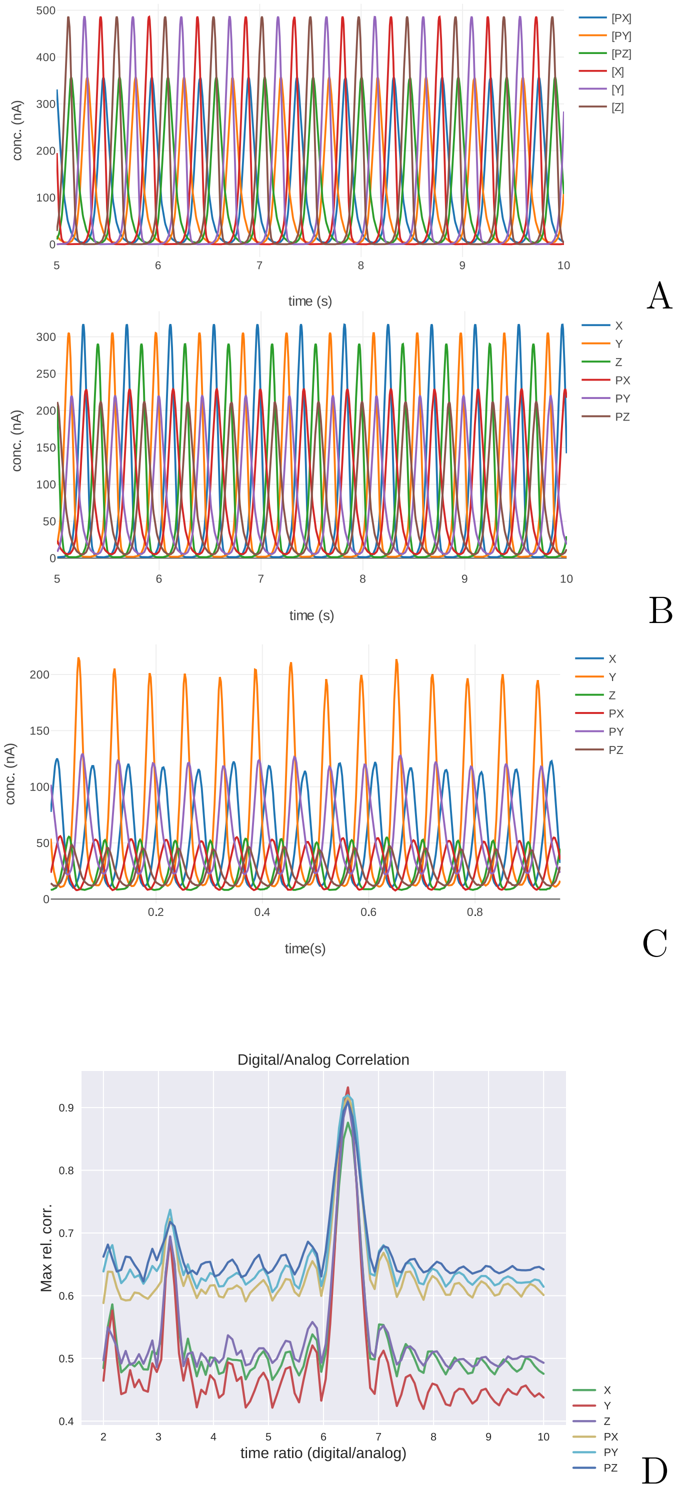


Figure 4.14: Comparison of repressilator model simulations. Also shown is an SBML simulation of the original model (B), a block simulation (C), and data collected from the cytomorphic chip (D). Due to manufacturing variations, blocks in the cytomorphic chip have different gains, hence the peak heights are different. The time axis on the chip data plot corresponds to “simulation time,” i.e. the actual duration of running the simulation, as opposed to “model time”, the time according to the dynamics of the model. To be useful, a hardware simulation should take less time to run than the timescale according to the model’s dynamics. This allows for multiple *in silico* expriments to be performed in for the amount of time a single physical experiment would take. To quantify this ratio, we performed correlation analysis on the chip data against the block simulation based on the wiring on Figure 4.13 (D). The cross correlation shows a peak at 6.5 seconds, indicating a six–fold difference between model and simulation time. This does not represent a speedup over a software SBML simulation, but the cytomorphic chip exhibits constant scaling up to the maximum number of blocks.

المنارة للاستشارات
www.manaraa.com

4.9 Higher–Order Compilation

While the requirement that all processes in a model be reducible to mass–action kinetics may seen restrictive, there is a very large class of models that consist only of this type of process. The field of rule–based modeling is a very active area of research (see [68] for a review). Rule–based models are composed of multi–state species. For example, a protein can have multiple phosphorylation sites, and each site can exist in either a “phosphorylated” or “unphosphorylated” state. These rules can be used to generate a network of all enumerable molecular states, or alternately simulated stochastically without enumeration using agent–like methods such those employed by the simulator NFSim [282].

Rule–based models invariably generate mass–action networks when enumerated or otherwise reduced to a simulatable form (this is not strictly a requirement, but non–mass action networks are a rare use–case for rule–based modeling and are not considered here). Furthermore, whereas NFSim scales linearly with the number of rules [282], the cytomorphic chip has constant scaling up to the maximum number of blocks (with the ability to connect to additional chips and thus increase the maximum network size in the future).

Taken together, these factors suggest that the ideal application of cytomorphic hardware could be the simulation of rule–based mass–action networks. A model of a MAPK signaling cascade from the rule–based modeling platform PySB [176] and based on a previous study [64] was used to validate the cytomorphic compiler. This cascade consists of the MAPK ERK and its upstream activators Ras and Raf. A contact map for this model, generated using RuleBender [280], is shown in Figure 4.15.

Rule–based models are one approach to managing complexity. They allow the user to specify models in terms of biomolecules with multiple states (such as multiple phosphorylation sites and multiple binding sites that can either be occupied or not) and automatically enumerate all possible discrete states. Similarly, electronics designers have long used similar bookkeeping technologies (very large scale integration, or VLSI) to generate chip layouts from high–level logic specifications. Just as VLSI was a necessary technology for enabling

المنارة للاستشارات
www.manaraa.com

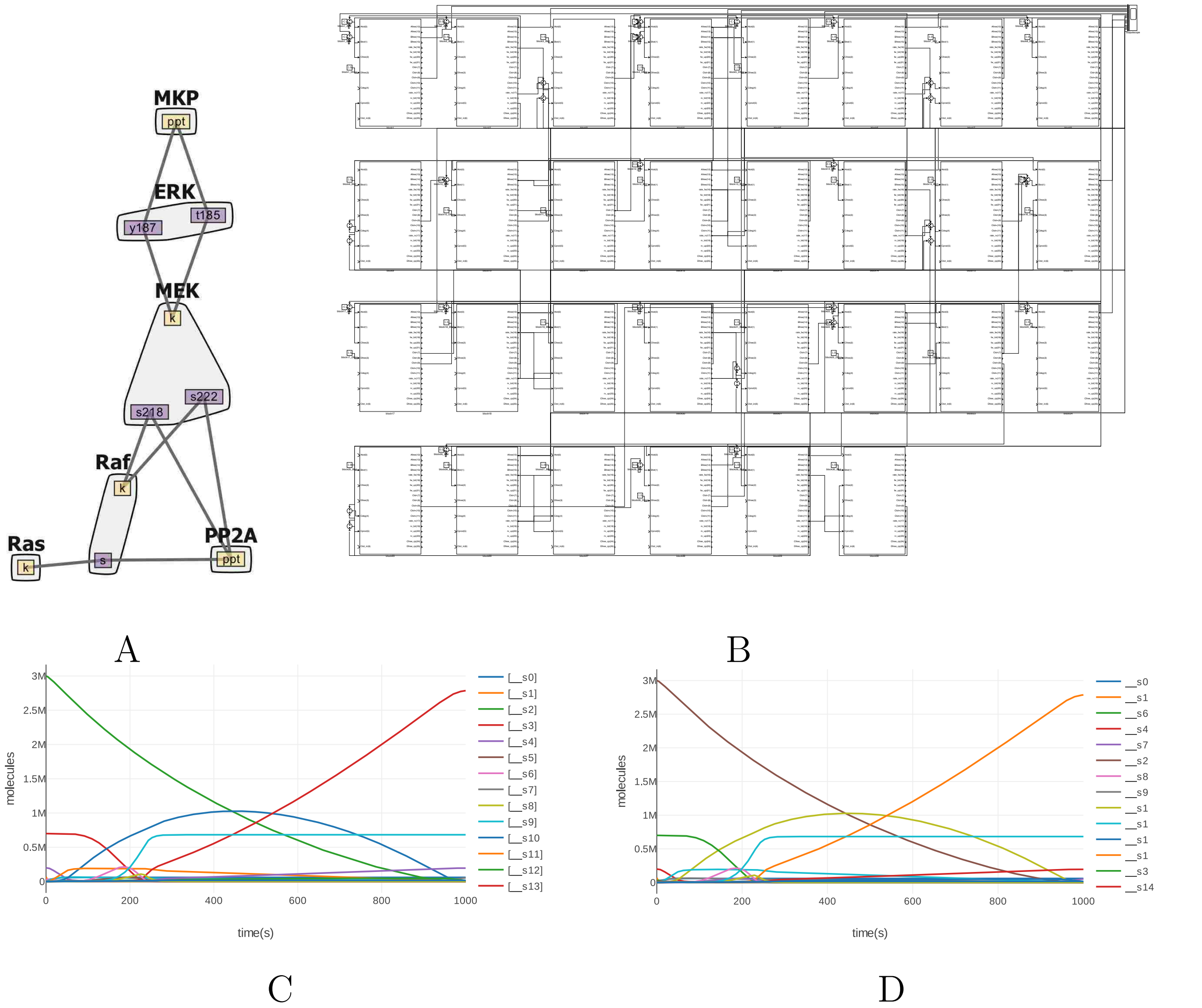


Figure 4.15: A rule–based MAP kinase model [176, 64] and its corresponding block configuration produced by the cytomorphic compiler. (A) A contact map for the kinase cascade generated using RuleBender [280]. The contact map shows the molecular species present in the model — Ras, Raf, MAPK/ERK Kinase (MEK), extracellular signal-regulated kinase (ERK), MAP kinase phosphatase (MKP), and Protein phosphatase 2 (PP2A). When expanded to an SBML mass–action network representation, this rule–based model expands into a network with 20 reactions and 21 distinct dynamical states. (B) The block layout of the PySB kinase cascade model compiled for the cytomorphic chip showing 30 blocks representing the 20 SBML reactions. This diagram is a Simulink model generated automatically by the compiler for validating the output. An SBML simulation of the flattened rule–based model (C) and a block simulation (D) are identical up to 3 significant figures (simulated using libRoadRunner, CVODES solver, absolute and relative tolerances 10^{-20} and 10^{-12} respectively.

المنارة للاستشارات

www.manaraa.com

rapid growth in complexity of integrated circuits (ICs), technologies such as the cytomorphic compiler presented here will be necessary for rapid growth of biomimetic electronics.

www.manaraa.com

Table 4.2: **Tunable Parameters for Each Block.**

Parameter	**Description**
`ratC`	Controls the degradation rate of the main product C when this block also serves as a degradation reaction.
`n`	Hill coefficient for forward binding. Useful in repressor kinetics.
`KDfw`	Forward–binding dissociation constant. Used to specify the forward rate via eq. 4.3.
`KDrv`	Reverse–binding dissociation constant. Used to specify the forward rate via eq. 4.3.
`kr`	The overall rate of the block. Can be used to tune the forward and reverse rates simultaneously (trading speed for stability or vice–versa).
`kdeg`	Auxiliary degradation rate used in the fan–in configuration.
`A_FB_EN,B_FB_EN`	Substrate depletion switches for reactants A and B respectively. When turned off, product of C does not consume A or B resp. (useful for modeling transcription and translation reactions).
`FF_EN_sw1,2,3,4`	Switches controlling the internal forward–reverse rate computation of the block. Only used in fan–in configurations, which each block must subtract the "main" production rate from its own rate.
`Ctot_sw`	A switch controlling whether the block's main output `Ctot` is supplied externally (not used in most configurations).

Programmable parameters for each cytomorphic block.

المنارة للاستشارات
www.manaraa.com

Chapter 5

CONCLUSION

5.1 Contributions to Standards Integration

In order to build larger, more complete, and more accurate dynamical models of cells and tissues, it will be necessary to reuse models of subsystems. This is currently very difficult due to the time–consuming and laborious process of manually reconstructing models from literature, or manually verifying third–party SBML models. Standards for encoding models exist, but they require considerable technical knowledge to use, thus causing many researchers to opt for a simpler, non–reproducible approach.

One contribution of this thesis is a platform that allows researchers to encode models using standard formats without requiring specialized knowledge of standards, thereby providing reproducibility benefits for the systems biology community. Tellurium also supports many other features — an integrated simulation engine and plotting facilities, bifurcation diagrams, and more, but the main benefit of the contributions described here is the integration of standards in a notebook environment.

Tellurium notebooks provide support for encapsulating both a model and its dynamics in a community–developed standard format, the COMBINE archive. This archive can contain the model as well as a number of simulations which test various dynamical properties of the model. Tellurium allows users to create COMBINE archives easily from SBML models, or import and modify preexisting COMBINE archives.

Tellurium integrates SBML, SED–ML, and COMBINE archives within a notebook environment, making it exceptionally easy for users to work with these standards, and obviating the need for users to understand the technical specifications of the standards. The availability of authoring tools such as Tellurium will make it possible for model repositories to

المنارة للاستشارات
www.manaraa.com

begin implementing support for SED–ML and COMBINE archives. Indeed, the JWS Online repository [229, 219] already has support for exporting COMBINE archives of models and simulations, which can be read by Tellurium. Other databases will hopefully follow suit so that it will be possible to automatically extract dynamical information from these repositories.

Tellurium's human–readable representation of COMBINE archives is highly important for facilitating model modification as described in this section. This feature enables researchers to experiment with models using alternate parameterizations in order to test the dynamical behavior of the models under varying conditions. This may lead to more robust models which lead to biological insight by providing predictions under a wide range of circumstances, as with the case studies presented in Chapter 2.

5.2 Contributions to Web–based Support for Standards

In order to support online repositories and modeling tools, better web–based support for standards is needed. Currently, there is no web–capable library that can read and write SBML models. Chapter 2 introduced libsbmljs, a WebAssembly / JavaScript library that can read and write all SBML packages. Tutorials, examples and extensive API documentation for potential users are all provided. A modular build system is also provided that can be used to regenerate the wrapper from any recent checkout of the libSBML C++ library from the stable or experimental branch, as well as in–browser tests of the wrapper using the Karma testing engine. This wrapper will hopefully enable the development of systems biology web applications and services that can use the SBML standard.

5.3 Contributions to Scalable Kinetic Modeling

Model calibration is often a major bottleneck in constructing dynamical models. The ability to accelerate this process would allow modelers more freedom to experiment with different model variants, and shorten the overall length of model development. We have shown here that the asynchronous, distributed island method yields accelerated convergence and better

المنارة للاستشارات
www.manaraa.com

quality of fit for very large, challenging optimization problems in systems biology.

However, the island method introduces new tunable hyperparameters, such as the migration topology, local algorithms for each island, and others. This can make using the island method difficult in practice. In our tests, topology and algorithm choice only exhibited a minor role in determining performance. Thus, the island method is useful as a general parallelization scheme without the need to tune these hyperparameters. Our tests indicate that the top–performing combination consisting of the *de1220* algorithm with the rim topology exhibits good general performance and scalability and should be sufficient for most users. Users can scale this combination by setting the number of islands equal to the total number of CPU cores in the cluster.

5.4 Contributions to Simulation Technology

Biomimetic special–purpose hardware utilizing analog computing has shown great promise for simulating chemical reaction networks more efficiently and (with a high–yield industrial manufacturing process[1]) faster than digital computers. This thesis described a compiler to convert from SBML models to a configuration for running simulations on specialized hardware.

This compiler forms a bridge between standard–encoded models and special purpose hardware. Previously, models had to be manually translated by an expert in both the hardware and SBML models, a laborious and time–consuming process. The significance of this contribution is that brings this specialized simulation method closer to practical realization by providing an automatic pathway for this translation process. It also enables scalability

[1] The prototype cytomorphic hardware is currently produced using a 130 nm process. Each reaction block in the chip uses fewer than 100 transistors and is more than 1000 times more energy efficient than a workstation computer. Modern ICs typically contain hundreds of millions of transistors. Thus, if commercialized using industry–standard manufacturing, cytomorphic hardware could reach over one million blocks per chip and an increase in simulation speed per block. This would allow the simulation of networks of millions of reactions in the 1s time regime. By contrast, the libRoadRunner SBML simulator, which attains near–optimal simulation speeds (as measured by comparing against manually written C++ model–specific code), requires $\approx 6s$ to simulate a Brusselator network of 2000 reactions on an Intel® Core™i7 3770.

المنارة للاستشارات
www.manaraa.com

to larger models. In the early years of computer science, it quickly became apparent that assembly language was not a viable tool for building large–scale software systems, which lead to the development of compilers for higher–level languages. This is also true for biochemical models, where increasing size and complexity creates a limit to the size of models that can be translated by hand. Cytomorphic compilers such as the archetypal implementation presented here are thus necessary for the continued development of cytomorphic hardware, and indeed suggest routes of advancement for other biomimetic systems.

The compiler presented here solves two main problems in the translation process: propagation of fluxes to avoid the divergence problem, and reduction of higher–order kinetics. While the approach described here is not completely general, the current rate law reductions cover a broad class of models and can, in principle, be extended to cover all known lumped kinetics expressions. Furthermore, a large body of rule–based models exists [68, 280, 43, 265, 45, 282, 176] that are already expressible in reduced mass–action form.

المنارة للاستشارات

www.manaraa.com

BIBLIOGRAPHY

[1] Apache zeppelin. https://zeppelin.apache.org/.

[2] Electron: Build cross platform desktop apps with javascript, html, and css. https://electronjs.org/.

[3] Git scm. https://git-scm.com/.

[4] The lsoda algorithm for differential equations as a shared library. https://github.com/sdwfrost/liblsoda.

[5] Maple. https://www.maplesoft.com/products/Maple/.

[6] Mathcad. https://www.ptc.com/engineering-math-software/mathcad/.

[7] Mathematica. https://www.wolfram.com/mathematica/.

[8] nteract: Revolutionizing the notebook experience. https://moderndata.plot.ly/nteract-revolutionizing-notebook-experience.

[9] Pathwaydesigner. http://pathwaydesigner.org/.

[10] Sbml flux balance constraints. http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/Flux_Balance_Constraints_(flux).

[11] Sbml geometry package motivation. http://sbml.org/images/f/ff/SpatialGeometry.pdf.

[12] Sbml geometry package wiki. http://sbml.org/Community/Wiki/SBML_Level_3_Geometry.

[13] Sbml multistate / multicomponent species. http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/spatial.

[14] Sbml spatial processes. http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/multi.

للاستشارات المنارة

www.manaraa.com

[15] Webassembly. https://webassembly.org.

[16] Webassembly garbage collection. https://github.com/WebAssembly/design/issues/1079.

[17] Reproduction of calzone et al. fig 1 and 3 as a combine archive. https://github.com/0u812/tellurium-combine-archive-test-cases/blob/master/demos/calzone-fig1-fig3.omex, 2017.

[18] Sbml test suite. http://sbml.org/Software/SBML_Test_Suite, 2017.

[19] Testing feedback regulation in the calzone model. https://github.com/0u812/tellurium-combine-archive-test-cases/blob/master/demos/calzone-feedback-study.omex, 2017.

[20] David Ackley. *A connectionist machine for genetic hillclimbing*, volume 28. Springer Science & Business Media, 2012.

[21] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[22] Rui Alves, Fernando Antunes, and Armindo Salvador. Tools for kinetic modeling of biochemical networks. *Nature biotechnology*, 24(6):667–672, 2006.

[23] Anaconda, Inc. Dask: Scalable analytics in python. https://dask.org/, 2019.

[24] Lauren B Andrews, Alec AK Nielsen, and Christopher A Voigt. Cellular checkpoint control using programmable sequential logic. *Science*, 361(6408):eaap8987, 2018.

[25] Christof Angermueller, Heather J Lee, Wolf Reik, and Oliver Stegle. Deepcpg: accurate prediction of single-cell dna methylation states using deep learning. *Genome biology*, 18(1):67, 2017.

[26] Paolo Antognetti and Guiseppe Massobrio. *Semiconductor device modeling with SPICE*. McGraw-Hill, Inc., 1990.

[27] Apache Software Foundation. Apache spark. https://spark.apache.org/, 2019.

[28] Theinmozhi Arulraj and Debashis Barik. Mathematical modeling identifies lck as a potential mediator for pd-1 induced inhibition of early tcr signaling. *PloS one*, 13(10):e0206232, 2018.

المنارة للاستشارات

www.manaraa.com

[29] Hnin W. Aung, Susan A. Henry, and Larry P. Walker. Revising the representation of fatty acid, glycerolipid, and glycerophospholipid metabolism in the consensus model of yeast metabolism. *Industrial Biotechnology*, 9(4):215–228, aug 2013.

[30] Bryan Bartley, Jacob Beal, Kevin Clancy, Goksel Misirli, Nicholas Roehner, Ernst Oberortner, Matthew Pocock, Michael Bissell, Curtis Madsen, Tramy Nguyen, et al. Synthetic biology open language (sbol) version 2.0. 0. *Journal of Integrative Bioinformatics (JIB)*, 12(2):902–991, 2015.

[31] Bryan A Bartley, Kyung Kim, J Kyle Medley, and Herbert M Sauro. Synthetic biology: Engineering living systems from biophysical principles. *Biophysical journal*, 112(6):1050–1058, 2017.

[32] David M Beazley et al. Swig: An easy to use tool for integrating scripting languages with c and c++. In *Tcl/Tk Workshop*, 1996.

[33] Scott A Becker, Adam M Feist, Monica L Mo, Gregory Hannum, Bernhard Ø Palsson, and Markus J Herrgard. Quantitative prediction of cellular metabolism with constraint-based models: the cobra toolbox. *Nature protocols*, 2(3):727, 2007.

[34] Marie Béněteau, Barbara Zunino, Marie A Jacquin, Ophélie Meynet, Johanna Chiche, Ludivine A Pradelli, Sandrine Marchetti, Aurore Cornille, Michel Carles, and Jean-Ehrland Ricci. Combination of glycolysis inhibition with chemotherapy results in an antitumor immune response. *Proceedings of the National Academy of Sciences*, 109(49):20071–20076, 2012.

[35] Theodore W Berger, Dong Song, Rosa HM Chan, Vasilis Z Marmarelis, Jeff LaCoss, Jack Wills, Robert E Hampson, Sam A Deadwyler, and John J Granacki. A hippocampal cognitive prosthesis: multi-input, multi-output nonlinear modeling and vlsi implementation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 20(2):198–211, 2012.

[36] Frank T Bergmann, Richard Adams, Stuart Moodie, Jonathan Cooper, Mihai Glont, Martin Golebiewski, Michael Hucka, Camille Laibe, Andrew K Miller, David P Nickerson, et al. Combine archive and omex format: one file to share all information to reproduce a modeling project. *BMC bioinformatics*, 15(1):369, 2014.

[37] Frank T. Bergmann and Sarah M. Keating. libCombine: a C++ API library supporting the COMBINE Archive, September 2016.

[38] Frank T Bergmann, David Nickerson, Dagmar Waltemath, and Martin Scharm. Sed-ml web tools: generate, modify and export standard-compliant simulation studies. *Bioinformatics*, 33(8):1253–1254, 2017.

المنارة للاستشارات
www.manaraa.com

[39] Frank T Bergmann, David Nickerson, Dagmar Walthemath, and Martin Scharm. Sed-ml web tools: Generate, modify and export standard-compliant simulation studies. *Bioinformatics*, pages 1253–1254, 2017.

[40] Frank T Bergmann and Herbert M Sauro. Sbw-a modular framework for systems biology. In *Simulation Conference, 2006. WSC 06. Proceedings of the Winter*, pages 1637–1645. IEEE, 2006.

[41] Frank T Bergmann and Herbert M Sauro. Comparing simulation results of sbml capable simulators. *Bioinformatics*, 24(17):1963–1965, 2008.

[42] Francesco Biscani, Dario Izzo, and Chit Hong Yam. A global optimisation toolbox for massively parallel engineering optimisation. *arXiv preprint arXiv:1004.3824*, 2010.

[43] Michael L Blinov, James R Faeder, Byron Goldstein, and William S Hlavacek. Bionetgen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–3291, 2004.

[44] Michael L Blinov, James R Faeder, Byron Goldstein, and William S Hlavacek. A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity. *Biosystems*, 83(2-3):136–151, 2006.

[45] Michael L. Blinov, James C. Schaff, Dan Vasilescu, Ion I. Moraru, Judy E. Bloom, and Leslie M. Loew. Compartmental and spatial rule-based modeling with virtual cell. *Biophysical Journal*, 113(7):1365–1372, 2017/10/03 2017.

[46] Igor W Bogorad, Tzu-Shyang Lin, and James C Liao. Synthetic non-oxidative glycolysis enables complete carbon conservation. *Nature*, 502(7473):693, 2013.

[47] Nikolay Borisov, Edita Aksamitiene, Anatoly Kiyatkin, Stefan Legewie, Jan Berkhout, Thomas Maiwald, Nikolai P Kaimachnikov, Jens Timmer, Jan B Hoek, and Boris N Kholodenko. Systems-level interactions between insulin–egf networks amplify mitogenic signaling. *Molecular systems biology*, 5(1):256, 2009.

[48] Benjamin J Bornstein, Sarah M Keating, Akiya Jouraku, and Michael Hucka. Libsbml: an api library for sbml. *Bioinformatics*, 24(6):880–881, 2008.

[49] Amin Espah Borujeni, Daniel Cetnar, Iman Farasat, Ashlee Smith, Natasha Lundgren, and Howard M. Salis. Precise quantification of translation inhibition by mRNA structures that overlap with the ribosomal footprint in n-terminal coding sequences. *Nucleic Acids Research*, 45(9):5437–5448, February 2017.

المنارة للاستشارات

www.manaraa.com

[50] Amin Espah Borujeni, Anirudh S. Channarasappa, and Howard M. Salis. Translation rate is controlled by coupled trade-offs between site accessibility, selective RNA unfolding and sliding at upstream standby sites. *Nucleic Acids Research*, 42(4):2646–2659, November 2013.

[51] Amin Espah Borujeni and Howard M Salis. Translation initiation is controlled by RNA folding kinetics via a ribosome drafting mechanism. *Journal of the American Chemical Society*, 138(22):7016–7023, May 2016.

[52] Janez Brest, Sao Greiner, Borko Boskovic, Marjan Mernik, and Viljem Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE transactions on evolutionary computation*, 10(6):646–657, 2006.

[53] Corentin Briat, Ankit Gupta, and Mustafa Khammash. Antithetic integral feedback ensures robust perfect adaptation in noisy biomolecular networks. *Cell systems*, 2(1):15–26, 2016.

[54] George Edward Briggs and John Burdon Sanderson Haldane. A note on the kinetics of enzyme action. *Biochemical journal*, 19(2):338, 1925.

[55] Robert W Brodersen. *Anatomy of a silicon compiler*, volume 181. Springer Science & Business Media, 2012.

[56] Nicolaas A Buijs, Verena Siewers, and Jens Nielsen. Advanced biofuel production by the yeast saccharomyces cerevisiae. *Current opinion in chemical biology*, 17(3):480–488, 2013.

[57] Eugene C Butcher, Ellen L Berg, and Eric J Kunkel. Systems biology in drug discovery. *Nature biotechnology*, 22(10):1253, 2004.

[58] Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. Vistrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747. ACM, 2006.

[59] Laurence Calzone, François Fages, and Sylvain Soliman. Biocham: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, 2006.

[60] Laurence Calzone, Denis Thieffry, John J Tyson, and Bela Novak. Dynamical modeling of syncytial mitotic cycles in drosophila embryos. *Molecular systems biology*, 3(1):131, 2007.

المنارة للاستشارات
www.manaraa.com

[61] Erick Cantu-Paz. *Efficient and accurate parallel genetic algorithms*, volume 1. Springer Science & Business Media, 2000.

[62] Yangxiaolu Cao, Marc D Ryser, Stephen Payne, Bochong Li, Christopher V Rao, and Lingchong You. Collective space-sensing coordinates pattern scaling in engineered bacteria. *Cell*, 165(3):620–630, 2016.

[63] Deepak Chandran, Frank T Bergmann, and Herbert M Sauro. Tinkercell: modular cad tool for synthetic biology. *Journal of biological engineering*, 3(1):19, 2009.

[64] William W Chen, Birgit Schoeberl, Paul J Jasper, Mario Niepel, Ulrik B Nielsen, Douglas A Lauffenburger, and Peter K Sorger. Input–output behavior of erbb signaling pathways as revealed by a mass action model trained against dynamic data. *Molecular systems biology*, 5(1):239, 2009.

[65] Ying-Nan P Chen, Matthew J LaMarche, Ho Man Chan, Peter Fekkes, Jorge Garcia-Fortanet, Michael G Acker, Brandon Antonakos, Christine Hiu-Tung Chen, Zhouliang Chen, Vesselina G Cooke, et al. Allosteric inhibition of shp2 phosphatase inhibits cancers driven by receptor tyrosine kinases. *Nature*, 535(7610):148, 2016.

[66] Kiri Choi, J Kyle Medley, Matthias König, Kaylene Stocking, Lucian Smith, Stanley Gu, and Herbert M Sauro. Tellurium: An extensible python-based modeling environment for systems and synthetic biology. *Biosystems*, 171:74–79, 2018.

[67] Kiri Choi, Lucian P Smith, J Kyle Medley, and Herbert M Sauro. phrased-ml: A paraphrased, human-readable adaptation of sed-ml. *Journal of Bioinformatics and Computational Biology*, page 1650035, 2016.

[68] Lily A Chylek, Leonard A Harris, Chang-Shung Tung, James R Faeder, Carlos F Lopez, and William S Hlavacek. Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 6(1):13–36, 2014.

[69] Scott D Cohen, Alan C Hindmarsh, Paul F Dubois, et al. Cvode, a stiff/nonstiff ode solver in c. *Computers in physics*, 10(2):138–143, 1996.

[70] James P Cohoon, Shailesh U Hegde, Worthy N Martin, and D Richards. Punctuated equilibria: a parallel genetic algorithm. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987., 1987.

المنارة للاستشارات
www.manaraa.com

[71] Edward F Connor and Daniel Simberloff. The assembly of species communities: chance or competition? *Ecology*, 60(6):1132–1140, 1979.

[72] Angelo Corana, Michele Marchesi, Claudio Martini, and Sandro Ridella. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Transactions on Mathematical Software (TOMS)*, 13(3):262–280, 1987.

[73] Mélanie Courtot, Nick Juty, Christian Knüpfer, Dagmar Waltemath, Anna Zhukova, Andreas Dräger, Michel Dumontier, Andrew Finney, Martin Golebiewski, Janna Hastings, et al. Controlled vocabularies and semantics in systems biology. *Molecular systems biology*, 7(1):543, 2011.

[74] Autumn A Cuellar, Catherine M Lloyd, Poul F Nielsen, David P Bullivant, David P Nickerson, and Peter J Hunter. An overview of cellml 1.1, a biological model description language. *Simulation*, 79(12):740–747, 2003.

[75] Dan Davidi, Elad Noor, Wolfram Liebermeister, Arren Bar-Even, Avi Flamholz, Katja Tummler, Uri Barenholz, Miki Goldenfeld, Tomer Shlomi, and Ron Milo. Global characterization of in vivo enzyme catalytic rates and their correspondence to in vitrokcat-measurements. *Proceedings of the National Academy of Sciences*, 113(12):3401–3406, March 2016.

[76] Lawrence Davis. Handbook of genetic algorithms. 1991.

[77] Charlotte M Deane, Łukasz Salwiński, Ioannis Xenarios, and David Eisenberg. Protein interactions: two methods for assessment of the reliability of high throughput observations. *Molecular & Cellular Proteomics*, 1(5):349–356, 2002.

[78] Kirill Degtyarenko, Paula De Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic acids research*, 36(suppl_1):D344–D350, 2007.

[79] Bryan S Der, Emerson Glassey, Bryan A Bartley, Casper Enghuus, Daniel B Goodman, D Benjamin Gordon, Christopher A Voigt, and Thomas E Gorochowski. Dnaplotlib: programmable visualization of genetic designs and associated data. *ACS synthetic biology*, 6(7):1115–1119, 2016.

[80] Harish Dharuri, Bela Novak, and John J Tyson. Biomd0000000107, 2007.

[81] Antony N Dodd, Neeraj Salathia, Anthony Hall, Eva Kévei, Réka Tóth, Ferenc Nagy, Julian M Hibberd, Andrew J Millar, and Alex AR Webb. Plant circadian

www.manaraa.com

clocks increase photosynthesis, growth, survival, and competitive advantage. *Science*, 309(5734):630–633, 2005.

[82] Andreas Dräger, Nicolas Rodriguez, Marine Dumousseau, Alexander Drr, Clemens Wrzodek, Nicolas Le Novre, Andreas Zell, and Michael Hucka. JSBML: a flexible Java library for working with SBML. *Bioinformatics*, 27(15):2167–2168, 06 2011.

[83] Brian Drawert, Andreas Hellander, Ben Bales, Debjani Banerjee, Giovanni Bellesia, Bernie J Daigle Jr, Geoffrey Douglas, Mengyuan Gu, Anand Gupta, Stefan Hellander, et al. Stochastic simulation service: Bridging the gap between the computational expert and the biologist. *PLoS computational biology*, 12(12):e1005220, 2016.

[84] Chris Drummond. Replicability is not reproducibility: nor is it good science. 2009.

[85] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. IEEE.

[86] Ali Ebrahim, Joshua A Lerman, Bernhard O Palsson, and Daniel R Hyduke. Cobrapy: constraints-based reconstruction and analysis for python. *BMC systems biology*, 7(1):74, 2013.

[87] Brace A Edgar, Frank Sprenger, Robert J Duronio, Pierre Leopold, and Patrick H O'Farrell. Distinct molecular mechanism regulate cell cycle timing at successive stages of drosophila embryogenesis. *Genes & development*, 8(4):440–452, 1994.

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

[89] Michael B Elowitz and Stanislas Leibler. Biomd0000000012, 2007.

[90] Saber M Elsayed, Ruhul A Sarker, and Daryl L Essam. Differential evolution with multiple strategies for solving cec2011 real-world numerical optimization problems. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 1041–1048. IEEE, 2011.

[91] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[92] Burçin Eröcal and William Stein. The sage project: Unifying free mathematical software to create. In *Mathematical Software-ICMS 2010: Third International Congress on Mathematical Software, Kobe, Japan, September 13-17, 2010, Proceedings*, volume 6327, page 12. Springer, 2010.

المنارة للاستشارات
www.manaraa.com

[93] Roland Ewald and Adelinde M Uhrmacher. Sessl: A domain-specific language for simulation experiments. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 24(2):11, 2014.

[94] Sisi Fan, Quentin Geissmann, Eszter Lakatos, Saulius Lukauskas, Angelique Ale, Ann C Babtie, Paul DW Kirk, and Michael PH Stumpf. Means: python package for moment expansion approximation, inference and simulation. *Bioinformatics*, 32(18):2863–2865, 2016.

[95] Jérôme Feret, Vincent Danos, Jean Krivine, Russ Harmer, and Walter Fontana. Internal coarse-graining of molecular systems. *Proceedings of the National Academy of Sciences*, 106(16):6453–6458, 2009.

[96] David Filmore. It's a gpcr world. *Modern drug discovery*, 7:24–28, 2004.

[97] Alexei V Finkelstein and Oleg Ptitsyn. *Protein physics: a course of lectures*. Elsevier, 2016.

[98] Max Franz, Christian T Lopes, Gerardo Huck, Yue Dong, Onur Sumer, and Gary D Bader. Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics*, 32(2):309–311, 2015.

[99] Fabian Fröhlich, Fabian J Theis, and Jan Hasenauer. Uncertainty analysis for non-identifiable dynamical systems: Profile likelihoods, bootstrapping and more. In *International Conference on Computational Methods in Systems Biology*, pages 61–72. Springer, 2014.

[100] Akira Funahashi, Yukiko Matsuoka, Akiya Jouraku, Mineo Morohashi, Norihiro Kikuchi, and Hiroaki Kitano. Celldesigner 3.5: A versatile modeling tool for biochemical networks. *Proc. IEEE*, 96(8):1254–1265, Aug 2008.

[101] Akira Funahashi, Mineo Morohashi, Hiroaki Kitano, and Naoki Tanimura. Celldesigner: a process diagram editor for gene-regulatory and biochemical networks. *BIOSILICO*, 1(5):159 – 162, 2003.

[102] Michal Galdzicki, Kevin P Clancy, Ernst Oberortner, Matthew Pocock, Jacqueline Y Quinn, Cesar A Rodriguez, Nicholas Roehner, Mandy L Wilson, Laura Adam, J Christopher Anderson, et al. The synthetic biology open language (sbol) provides a community standard for communicating designs in synthetic biology. *Nature biotechnology*, 32(6):545–550, 2014.

المنارة للاستشارات
www.manaraa.com

[103] Timothy S Gardner, Charles R Cantor, and James J Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767):339, 2000.

[104] Alan Garny and Peter J Hunter. Opencor: a modular and interoperable approach to computational biology. *Frontiers in physiology*, 6:26, 2015.

[105] Zong Woo Geem, Joong Hoon Kim, and Gobichettipalayam Vasudevan Loganathan. A new heuristic optimization algorithm: harmony search. *simulation*, 76(2):60–68, 2001.

[106] Michael A Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The journal of physical chemistry A*, 104(9):1876–1889, 2000.

[107] Scott Gilbert. *Developmental Biology*. Sinauer Associates, 6th edition, 2000.

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

[109] Daniel T Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[110] Daniel T Gillespie and Linda R Petzold. Improved leap-size selection for accelerated stochastic simulation. *The journal of chemical physics*, 119(16):8229–8234, 2003.

[111] Padraig Gleeson, Sharon Crook, Robert C Cannon, Michael L Hines, Guy O Billings, Matteo Farinella, Thomas M Morse, Andrew P Davison, Subhasis Ray, Upinder S Bhalla, et al. Neuroml: a language for describing data driven models of neurons and networks with a high degree of biological detail. *PLoS computational biology*, 6(6):e1000815, 2010.

[112] Jeremy Goecks, Anton Nekrutenko, and James Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, 11(8):R86, 2010.

[113] Andreas O Griewank. Generalized descent for global optimization. *Journal of optimization theory and applications*, 34(1):11–39, 1981.

[114] Stanley Gu and J Kyle Medley. Notebook for converting sbml test cases to combine archives, 2017.

[115] Jeremy Gunawardena. Models in biology:accurate descriptions of our pathetic thinking. *BMC biology*, 12(1):29, 2014.

المنارة للاستشارات

www.manaraa.com

[116] Xin Guo, Honggui Li, Hang Xu, Shihlung Woo, Hui Dong, Fuer Lu, Alex J Lange, and Chaodong Wu. Glycolysis in the control of blood glucose homeostasis. *Acta Pharmaceutica Sinica B*, 2(4):358–367, 2012.

[117] Melinda Halasz, Boris N Kholodenko, Walter Kolch, and Tapesh Santra. Integrating network reconstruction with mechanistic modeling to predict cancer therapies. *Sci. Signal.*, 9(455):ra114–ra114, 2016.

[118] Leonard A Harris, Marco S Nobile, James C Pino, Alexander LR Lubbock, Daniela Besozzi, Giancarlo Mauri, Paolo Cazzaniga, and Carlos F Lopez. Gpu-powered model analysis with pysb/cupsoda. *Bioinformatics*, 33(21):3492–3494, 2017.

[119] Martin Haslbeck, Titus Franzmann, Daniel Weinfurtner, and Johannes Buchner. Some like it hot: the structure and function of small heat-shock proteins. *Nature structural & molecular biology*, 12(10):842, 2005.

[120] Janna Hastings, Paula de Matos, Adriano Dekker, Marcus Ennis, Bhavana Harsha, Namrata Kale, Venkatesh Muthukrishnan, Gareth Owen, Steve Turner, Mark Williams, et al. The chebi reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic acids research*, 41(D1):D456–D463, 2012.

[121] Janna Hastings, Gareth Owen, Adriano Dekker, Marcus Ennis, Namrata Kale, Venkatesh Muthukrishnan, Steve Turner, Neil Swainston, Pedro Mendes, and Christoph Steinbeck. Chebi in 2016: Improved services and an expanding collection of metabolites. *Nucleic acids research*, 44(D1):D1214–D1219, 2015.

[122] David Heckmann, Colton J Lloyd, Nathan Mih, Yuanchi Ha, Daniel C Zielinski, Zachary B Haiman, Abdelmoneim Amer Desouki, Martin J Lercher, and Bernhard O Palsson. Machine learning applied to enzyme turnover numbers reveals protein structural correlates and improves metabolic models. *Nature communications*, 9(1):5252, 2018.

[123] Domitille Heitzler, Guillaume Durand, Nathalie Gallay, Aurélien Rizk, Seungkirl Ahn, Jihee Kim, Jonathan D Violin, Laurence Dupuy, Christophe Gauthier, Vincent Piketty, et al. Competing g protein-coupled receptor kinases balance g protein and β-arrestin signaling. *Molecular systems biology*, 8(1):590, 2012.

[124] A. C. Hindmarsh. Odepack, a systematized collection of ode solvers.

[125] David D Ho, Avidan U Neumann, Alan S Perelson, Wen Chen, John M Leonard, and Martin Markowitz. Rapid turnover of plasma virions and cd4 lymphocytes in hiv-1 infection. *Nature*, 373(6510):123, 1995.

المنارة للاستشارات
www.manaraa.com

[126] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

[127] Stefan Hoops, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. Copasia complex pathway simulator. *Bioinformatics*, 22(24):3067–3074, 2006.

[128] Min-Chi Hsiao, Chiu-Hsien Chan, Vijay Srinivasan, Ashish Ahuja, Gopal Erinjippurath, Theodoros P Zanos, Ghassan Gholmieh, Dong Song, Jack D Wills, Jeff LaCoss, et al. Vlsi implementation of a nonlinear neuronal model: a" neural prosthesis" to restore hippocampal trisynaptic dynamics. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4396–4399. IEEE, 2006.

[129] Jun-yong Huang and Jordan W Raff. The dynamic localisation of the drosophila apc/c: evidence for the existence of multiple complexes that perform distinct functions and are differentially localised. *Journal of cell science*, 115(14):2847–2856, 2002.

[130] Daphne HEW Huberts, Bastian Niebel, and Matthias Heinemann. A flux-sensing mechanism could regulate the switch between respiration and fermentation. *FEMS yeast research*, 12(2):118–128, 2012.

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

[132] Terence Hwa. From physiology to molecules: a top-down approach towards predictive biology. Univ. Washington, 2016.

[133] Giacomo Indiveri, Bernabé Linares-Barranco, Tara Julia Hamilton, André Van Schaik, Ralph Etienne-Cummings, Tobi Delbruck, Shih-Chii Liu, Piotr Dudek, Philipp Häfliger, Sylvie Renaud, et al. Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5:73, 2011.

[134] Brian P Ingalls. A frequency domain approach to sensitivity analysis of biochemical networks. *The Journal of Physical Chemistry B*, 108(3):1143–1152, 2004.

[135] Dario Izzo, Marek Ruciński, and Francesco Biscani. The generalized island model. In *Parallel Architectures and Bioinspired Algorithms*, pages 151–169. Springer, 2012.

المنارة للاستشارات

www.manaraa.com

[136] WOLF Jana and Reinhart Heinrich. Effect of cellular interaction on glycolytic oscillations in yeast: a theoretical investigation. *Biochemical Journal*, 345(2):321–334, 2000.

[137] Khuloud Jaqaman and Gaudenz Danuser. Linking data to models: data regression. *Nature Reviews Molecular Cell Biology*, 7(11):813, 2006.

[138] Dave Johannsen. Bristle blocks: A silicon compiler. In *16th Design Automation Conference*, pages 310–313. IEEE, 1979.

[139] Kenneth A Johnson and Roger S Goody. The original michaelis constant: translation of the 1913 michaelis–menten paper. *Biochemistry*, 50(39):8264–8269, 2011.

[140] Eric Jones, Travis Oliphant, and Pearu Peterson. {SciPy}: open source scientific tools for {Python}. 2014.

[141] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. Parallel and hybrid models for multi-objective optimization: Application to the vehicle routing problem. In *International Conference on Parallel Problem Solving from Nature*, pages 271–280. Springer, 2002.

[142] Vanessa I. Jurtz, Alexander R. Johansen, Morten Nielsen, Jose J. A. Armenteros, Nielsen Henrik, Casper K. Sønderby, Ole Winther, and Sønderby Søren K. An introduction to deep learning on biological sequence data – examples and solutions. *Bioinformatics*, 33(16):In press, 2017.

[143] Nick Juty and Nicolas le Novère. Systems biology ontology. *Encyclopedia of Systems Biology*, pages 2063–2063, 2013.

[144] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471, 2007.

[145] Jonathan R Karr, Jayodita C Sanghvi, Derek N Macklin, Miriam V Gutschow, Jared M Jacobs, Benjamin Bolival, Nacyra Assad-Garcia, John I Glass, and Markus W Covert. A whole-cell computational model predicts phenotype from genotype. *Cell*, 150(2):389–401, 2012.

[146] Jonathan R Karr, Alex H Williams, Jeremy D Zucker, Andreas Raue, Bernhard Steiert, Jens Timmer, Clemens Kreutz, Simon Wilkinson, Brandon A Allgood, Brian M Bot, et al. Summary of the dream8 parameter estimation challenge: toward parameter identification for whole-cell models. *PLoS computational biology*, 11(5):e1004096, 2015.

المنارة للاستشارات
www.manaraa.com

[147] Stuart Kauffman, Carsten Peterson, Björn Samuelsson, and Carl Troein. Random boolean network models and the yeast transcriptional network. *Proceedings of the National Academy of Sciences*, 100(25):14796–14799, 2003.

[148] Kyle Kelley and nteract Developers. nteract Literate Coding Notebook. 11 2017.

[149] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*. IEEE.

[150] Boris N Kholodenko, Oleg V Demin, Gisela Moehren, and Jan B Hoek. Quantification of short term signaling by the epidermal growth factor receptor. *Journal of Biological Chemistry*, 274(42):30169–30181, 1999.

[151] Kyoung Ae Kim, Sabrina L Spencer, John G Albeck, John M Burke, Peter K Sorger, Suzanne Gaudet, et al. Systematic calibration of a cell signaling network model. *BMC bioinformatics*, 11(1):202, 2010.

[152] Kyung Hyuk Kim, Hong Qian, and Herbert M Sauro. Nonlinear biochemical signal processing via noise propagation. *The Journal of chemical physics*, 139(14):10B608_1, 2013.

[153] Zachary A King, Andreas Dräger, Ali Ebrahim, Nikolaus Sonnenschein, Nathan E Lewis, and Bernhard O Palsson. Escher: a web application for building, sharing, and embedding data-rich visualizations of biological pathways. *PLoS computational biology*, 11(8):e1004321, 2015.

[154] Zachary A King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A Lerman, Ali Ebrahim, Bernhard O Palsson, and Nathan E Lewis. Bigg models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic acids research*, 44(D1):D515–D522, 2015.

[155] Hiroaki Kitano. Perspectives on systems biology. *New Generation Computing*, 18(3):199–216, 2000.

[156] Anatoly Kiyatkin, Edita Aksamitiene, Nick I Markevich, Nikolay M Borisov, Jan B Hoek, and Boris N Kholodenko. Scaffolding protein grb2-associated binder 1 sustains epidermal growth factor-induced mitogenic and survival signaling by multiple positive feedback loops. *Journal of biological chemistry*, 281(29):19925–19938, 2006.

[157] D. E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

المنارة للاستشارات

www.manaraa.com

[158] Karl Kochanowski, Benjamin Volkmer, Luca Gerosa, Bart R Haverkorn van Rijsewijk, Alexander Schmidt, and Matthias Heinemann. Functioning of a metabolic flux sensor in escherichia coli. *Proceedings of the National Academy of Sciences*, 110(3):1130–1135, 2013.

[159] Walter Kolch, Melinda Halasz, Marina Granovskaya, and Boris N Kholodenko. The dynamic control of signal transduction networks in cancer cells. *Nature Reviews Cancer*, 15(9):515, 2015.

[160] Oliver Kotte, Judith B Zaugg, and Matthias Heinemann. Bacterial adaptation through distributed sensing of metabolic fluxes. *Molecular systems biology*, 6(1):355, 2010.

[161] TJ Kowalski, DJ Geiger, WH Wolf, and W Fichtner. The vlsi design automation assistant: From algorithms to silicon. *IEEE Design & Test of Computers*, 2(4):33–43, 1985.

[162] John R Koza et al. *Genetic programming II*, volume 17. MIT press Cambridge, MA, 1994.

[163] Nicolas Le Novere. Landing page for sbo term 0000290, 2006.

[164] Nicolas Le Novere, Benjamin Bornstein, Alexander Broicher, Melanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert Sauro, Maria Schilstra, Bruce Shapiro, et al. Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic acids research*, 34(suppl 1):D689–D691, 2006.

[165] Nicolas Le Novére, Nicolas Rodriguez, Finja Wrzodek, Florian Mittag, Sebastian Frhlich, Michael Hucka, Alex Thomas, Bernhard . Palsson, Nathan E. Lewis, Andreas Drger, Chris J. Myers, Leandro Watanabe, Ibrahim Y. Vazirabad, Victor Kofia, Harold F. Gmez, Alexander Diamantikos, Eugen Netz, Jakob Matthes, Johannes Eichner, Roland Keller, Jan Rudolph, and Clemens Wrzodek. JSBML 1.0: providing a smorgasbord of options to encode systems biology models. *Bioinformatics*, 31(20):3383–3386, 06 2015.

[166] Young Jun Lee, Jihyun Lee, Kyung Ki Kim, Yong-Bin Kim, and Joseph Ayers. Low power cmos electronic central pattern generator design for a biomimetic underwater robot. *Neurocomputing*, 71(1-3):284–296, 2007.

[167] Joanna Lewis, Charles E. Breeze, Jane Charlesworth, Oliver J. Maclaren, and Jonathan Cooper. Where next for the reproducibility agenda in computational biology? *BMC Systems Biology*, 10(1):52, Jul 2016.

المنارة للاستشارات
www.manaraa.com

[168] Chen Li, Marco Donizelli, Nicolas Rodriguez, Harish Dharuri, Lukas Endler, Vijayalakshmi Chelliah, Lu Li, Enuo He, Arnaud Henry, Melanie I Stefan, et al. Biomodels database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC systems biology*, 4(1):92, 2010.

[169] Genyuan Li, Herschel Rabitz, Jishan Hu, Zheng Chen, and Yiguang Ju. Regularized random-sampling high dimensional model representation (rs-hdmr). *Journal of Mathematical Chemistry*, 43(3):1207–1232, 2008.

[170] Han Li and James C Liao. Engineering a cyanobacterium as the catalyst for the photosynthetic conversion of co 2 to 1, 2-propanediol. *Microbial cell factories*, 12(1):4, 2013.

[171] Juliane Liepe, Chris Barnes, Erika Cule, Kamil Erguler, Paul Kirk, Tina Toni, and Michael PH Stumpf. Abc-sysbioapproximate bayesian computation in python with gpu support. *Bioinformatics*, 26(14):1797–1799, 2010.

[172] Juliane Liepe, Paul Kirk, Sarah Filippi, Tina Toni, Chris P Barnes, and Michael PH Stumpf. A framework for parameter estimation and model selection from experimental data in systems biology using approximate bayesian computation. *Nature protocols*, 9(2):439, 2014.

[173] Christian Lieven, Moritz Emanuel Beber, Brett G Olivier, Frank T Bergmann, Meric Ataman, Parizad Babaei, Jennifer A Bartell, Lars M Blank, Siddharth Chauhan, Kevin Correia, et al. Memote: A community-driven effort towards a standardized genome-scale metabolic model test suite. *BioRxiv*, page 350991, 2018.

[174] Catherine M Lloyd, James R Lawson, Peter J Hunter, and Poul F Nielsen. The cellml model repository. *Bioinformatics*, 24(18):2122–2123, 2008.

[175] Alba Lopez, Mtys Fodor, Anna Sirunian, Ali Kaafarani, Christian Lieven, Nikolaus Sonnenschein, Tala Azrak, and Moritz E. Beber. Dd-decaf/caffeine: Version 1. `https://doi.org/10.5281/zenodo.2616028`, March 2019.

[176] Carlos F Lopez, Jeremy L Muhlich, John A Bachman, and Peter K Sorger. Programming biological models in python using pysb. *Molecular systems biology*, 9(1):646, 2013.

[177] Ding Ma, Laurence Yang, Ronan MT Fleming, Ines Thiele, Bernhard O Palsson, and Michael A Saunders. Reliable and efficient solution of genome-scale models of metabolism and macromolecular expression. *Scientific reports*, 7:40863, 2017.

المنارة للاستشارات

www.manaraa.com

[178] Rainer Machne, Jana Wolf, , Reinhart Heinrich, and Hiroshi Kuriyama. Biomd0000000090, 2007.

[179] Aidan MacNamara, Camille Terfve, David Henriques, Beatriz Peñalver Bernabé, and Julio Saez-Rodriguez. State–time spectrum of signal transduction logic models. *Physical biology*, 9(4):045003, 2012.

[180] Curtis Madsen, Zhen Zhang, Nicholas Roehner, Chris Winstead, and Chris Myers. Stochastic model checking of genetic circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 11(3):23, 2014.

[181] Linus Magnusson and Erland Källén. Factors influencing skill improvements in the ecmwf forecasting system. *Monthly Weather Review*, 141(9):3142–3153, 2013.

[182] Shmoolik Mangan and Uri Alon. Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences*, 100(21):11980–11985, 2003.

[183] Marcus Märtens and Dario Izzo. The asynchronous island model and nsga-ii: study of a new migration operator and its performance. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 1173–1180. ACM, 2013.

[184] William A Martin. Determining the equivalence of algebraic expressions by hash coding. In *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, pages 305–310. ACM, 1971.

[185] Saroj P Mathupala, Young H Ko, and Peter L Pedersen. Hexokinase-2 bound to mitochondria: cancer's stygian link to the warburg effect and a pivotal target for effective therapy. In *Seminars in cancer biology*, volume 19, pages 17–24. Elsevier, 2009.

[186] MATLAB. *MATLAB version 7.10.0 (R2016b)*. The MathWorks Inc., Natick, Massachusetts, 2016.

[187] Robert A McDougal, Anna S Bulanova, and William W Lytton. Reproducibility in computational neuroscience models and simulations. *IEEE Transactions on Biomedical Engineering*, 63(10):2021–2035, 2016.

[188] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc., 2012.

[189] J Kyle Medley. Hill coefficient study based on wolf 2001.

المنارة للاستشارات

www.manaraa.com

[190] J Kyle Medley. Notebook for running tellurium sbml test suite, 2017.

[191] J Kyle Medley. Tellurium combine archive tests, 2017.

[192] J Kyle Medley. Tellurium sbml test suite notebook, 2017.

[193] J Kyle Medley. Tellurium sbml test suite notebooks, 2017.

[194] J Kyle Medley. Tellurium sbml test suite performance benchmark notebook, 2017.

[195] J Kyle Medley. Wolf 2001 oxygen–only plot, 2017.

[196] J Kyle Medley, Kiri Choi, Matthias König, Lucian Smith, Stanley Gu, Joseph Hellerstein, Stuart C Sealfon, and Herbert M Sauro. Tellurium notebooksan environment for reproducible dynamical modeling in systems biology. *PLoS computational biology*, 14(6):e1006220, 2018.

[197] J Kyle Medley, Arthur P Goldberg, and Jonathan R Karr. Guidelines for reproducibly building and simulating systems biology models. *IEEE Transactions on Biomedical Engineering*, 63(10):2015–2020, 2016.

[198] Nouredine Melab, El-Ghazali Talbi, et al. Gpu-based island model for evolutionary algorithms. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1089–1096. ACM, 2010.

[199] Pedro Mendes, Stefan Hoops, Sven Sahle, Ralph Gauges, Joseph Dada, and Ursula Kummer. Computational modeling of biochemical networks using copasi. *Methods Mol Biol.*, pages 17–59, 2009.

[200] Leonor Menten and MI Michaelis. Die kinetik der invertinwirkung. *Biochem Z*, 49(333-369):5, 1913.

[201] Pierre Millard, Kieran Smallbone, and Pedro Mendes. Metabolic regulation is sufficient for global and robust coordination of glucose uptake, catabolism, energy production and growth in escherichia coli. *PLoS computational biology*, 13(2):e1005396, 2017.

[202] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[203] Eshan D Mitra, Raquel Dias, Richard G Posner, and William S Hlavacek. Using both qualitative and quantitative data in parameter identification for systems biology models. *Nature communications*, 9(1):3901, 2018.

المنارة للاستشارات

www.manaraa.com

[204] Aaron Mobley, Suzanne K Linder, Russell Braeuer, Lee M Ellis, and Leonard Zwelling. A survey on data reproducibility in cancer research provides insights into our limited ability to translate findings from the laboratory to the clinic. *PLoS One*, 8(5):e63221, 2013.

[205] Ion I Moraru, James C Schaff, Boris M Slepchenko, ML Blinov, Frank Morgan, Anuradha Lakshminarayana, Fei Gao, Yuhua Li, and Leslie M Loew. Virtual cell modelling and simulation software environment. *IET Syst. Biol.*, 2(5):352–362, 2008.

[206] Douglas B Murray, Manfred Beckmann, and Hiroaki Kitano. Regulation of yeast oscillatory dynamics. *Proceedings of the National Academy of Sciences*, 104(7):2241–2246, 2007.

[207] Chris J Myers, Nathan Barker, Kevin Jones, Hiroyuki Kuwahara, Curtis Madsen, and Nam-Phuong D Nguyen. ibiosim: a tool for the analysis and design of genetic circuits. *Bioinformatics*, 25(21):2848–2849, 2009.

[208] Maxwell Lewis Neal, Matthias König, David Nickerson, Göksel Mısırlı, Reza Kalbasi, Andreas Dräger, Koray Atalag, Vijayalakshmi Chelliah, Michael T Cooling, Daniel L Cook, et al. Harmonizing semantic annotations for computational models in biology. *Briefings in bioinformatics*, 20(2):540–550, 2018.

[209] Alexander Neckar, Terrence C Stewart, Ben V Benjamin, and Kwabena Boahen. Optimizing an analog neuron circuit design for nonlinear function approximation. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018.

[210] David D. van Niekerk, Gerald P. Penkler, Francois du Toit, and Jacky L. Snoep. Targeting glycolysis in the malaria parasite plasmodium falciparum. *The FEBS Journal*, 283(4):634–646, 2 2016.

[211] Alec AK Nielsen, Bryan S Der, Jonghyeon Shin, Prashant Vaidyanathan, Vanya Paralanov, Elizabeth A Strychalski, David Ross, Douglas Densmore, and Christopher A Voigt. Genetic circuit design automation. *Science*, 352(6281):aac7341, 2016.

[212] Denis Noble. From the hodgkin–huxley axon to the virtual heart. *The Journal of physiology*, 580(1):15–22, 2007.

[213] Bela Novak, Vijayalakshmi Chelliah, PK Vinod, Paula Freire, Ahmed Rattani, Andrea Ciliberto, and Frank Uhlmann. Biomd0000000370, 2011.

المنارة للاستشارات
www.manaraa.com

[214] Bela Novak and John J Tyson. Numerical analysis of a comprehensive model of m-phase control in xenopus oocyte extracts and intact embryos. *Journal of cell science*, 106(4):1153–1168, 1993.

[215] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R Pocock, Anil Wipat, et al. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[216] B. G. Olivier and F. T. Bergmann. The Systems Biology Markup Language (SBML) Level 3 Package: Flux Balance Constraints. *J Integr Bioinform*, 12(2):660–690, Jun 2015.

[217] B. G. Olivier and F. T. Bergmann. SBML Level 3 Package: Flux Balance Constraints version 2. *J Integr Bioinform*, 15(1), Mar 2018.

[218] Brett G Olivier, Johann M Rohwer, and Jan-Hendrik S Hofmeyr. Modelling cellular systems with pysces. *Bioinformatics*, 21(4):560–561, 2005.

[219] Brett G Olivier and Jacky L Snoep. Web-based kinetic modelling using jws online. *Bioinformatics*, 20(13):2143–2144, 2004.

[220] Clare M OConnor, Jill U Adams, and Jennifer Fairman. Essentials of cell biology. *Cambridge, MA: NPG Education*, 1, 2010.

[221] PaGMO Developers. Pagmo and pygmo. `https://esa.github.io/pagmo2/`, 2019.

[222] Robert Palmér, Elin Nyman, Mark Penney, Anna Marley, Gunnar Cedersund, and Balaji Agoram. Effects of il-1β–blocking therapies in type 2 diabetes mellitus: A quantitative systems pharmacology modeling approach to explore underlying mechanisms. *CPT: pharmacometrics & systems pharmacology*, 3(6):1–8, 2014.

[223] Erika L Pearce, Maya C Poffenberger, Chih-Hao Chang, and Russell G Jones. Fueling immunity: insights into metabolism and lymphocyte function. *Science*, 342(6155):1242454, 2013.

[224] Peter L Pedersen. Warburg, me and hexokinase 2: Multiple discoveries of key molecular events underlying one of cancers most common phenotypes, the warburg effect, ie, elevated glycolysis in the presence of oxygen, 2007.

[225] Grace CY Peng. Moving toward model reproducibility and reusability. *IEEE Transactions on Biomedical Engineering*, 63(10):1997–1998, 2016.

المنارة للاستشارات

www.manaraa.com

[226] Roger D Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011.

[227] Gerald Penkler, Francois Du Toit, Waldo Adams, Marina Rautenbach, Daniel C Palm, David D Van Niekerk, and Jacky L Snoep. Construction and validation of a detailed kinetic model of glycolysis in plasmodium falciparum. *The FEBS journal*, 282(8):1481–1511, 2015.

[228] Cláudio MNA Pereira and Celso MF Lapa. Parallel island genetic algorithm applied to a nuclear power plant auxiliary feedwater system surveillance tests policy optimization. *Annals of Nuclear Energy*, 30(16):1665–1675, 2003.

[229] Martin Peters, Johann J. Eicher, David D. van Niekerk, Dagmar Waltemath, and Jacky L. Snoep. The jws online simulation database. *Bioinformatics*, 33(10):1589–1590, 2017.

[230] L Petzold and A LSODA. Computing and mathematics research division. *Lawrence Livermore National Laboratory*, 1989.

[231] Linda Petzold. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM journal on scientific and statistical computing*, 4(1):136–148, 1983.

[232] LR Petzold and AC Hindmarsh. Lsoda. *Computing and Mathematics Research Division, I-316 Lawrence Livermore National Laboratory, Livermore, CA*, 94550, 1997.

[233] Stephen R Piccolo and Michael B Frampton. Tools and techniques for computational reproducibility. *GigaScience*, 5(1):30, 2016.

[234] David W Piston and Gert-Jan Kremers. Fluorescent protein fret: the good, the bad and the ugly. *Trends in biochemical sciences*, 32(9):407–414, 2007.

[235] Sergei Kosakovsky Pond, Samir Wadhawan, Francesca Chiaromonte, Guruprasad Ananda, Wen-Yu Chung, James Taylor, Anton Nekrutenko, Galaxy Team, et al. Windshield splatter analysis with the galaxy metagenomic pipeline. *Genome research*, 19(11):2144–2153, 2009.

[236] MG Poolman. Scrumpy: metabolic modelling with python. *IEE Proceedings-Systems Biology*, 153(5):375–378, 2006.

[237] Mitchell A Potter and Kenneth A De Jong. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 249–257. Springer, 1994.

للاستشارات المنارة
www.manaraa.com

[238] Florian Prinz, Thomas Schlange, and Khusru Asadullah. Believe it or not: how much can we rely on published data on potential drug targets? *Nature reviews Drug discovery*, 10(9):712–712, 2011.

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

[240] Wei Quan, Bikun Chen, and Fei Shu. Publish or impoverish: An investigation of the monetary reward system of science in china (1999-2016). *Aslib Journal of Information Management*, just-accepted(just-accepted):00–00, 2017.

[241] Jacqueline Y Quinn, Robert Sidney Cox III, Aaron Adler, Jacob Beal, Swapnil Bhatia, Yizhi Cai, Joanna Chen, Kevin Clancy, Michal Galdzicki, Nathan J Hillson, et al. Sbol visual: a graphical language for genetic designs. *PLoS biology*, 13(12):e1002310, 2015.

[242] Jordan W Raff, Kim Jeffers, and Jun-yong Huang. The roles of fzy/cdc20 and fzr/cdh1 in regulating the destruction of cyclin b in space and time. *The Journal of cell biology*, 157(7):1139–1149, 2002.

[243] M Ragan-Kelley, F Perez, B Granger, T Kluyver, P Ivanov, J Frederic, and M Bussonnier. The jupyter/ipython architecture: a unified view of computational research, from interactive exploration to communication and publication. In *AGU Fall Meeting Abstracts*, 2014.

[244] LA Rastrigin. Systems of extremal control. *Nauka*, 1974.

[245] Muruhan Rathinam, Linda R Petzold, Yang Cao, and Daniel T Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *The Journal of Chemical Physics*, 119(24):12784–12794, 2003.

[246] Andreas Raue, Marcel Schilling, Julie Bachmann, Andrew Matteson, Max Schelke, Daniel Kaschek, Sabine Hug, Clemens Kreutz, Brian D Harms, Fabian J Theis, et al. Lessons learned from quantitative dynamical modeling in systems biology. *PloS one*, 8(9):e74335, 2013.

[247] Andreas Raue, Bernhard Steiert, Max Schelker, Clemens Kreutz, Tim Maiwald, Helge Hass, Joep Vanlier, C Tönsing, L Adlung, R Engesser, et al. Data2dynamics: a modeling environment tailored to parameter estimation in dynamical systems. *Bioinformatics*, 31(21):3558–3560, 2015.

[248] Jennifer L Reed and Bernhard Ø Palsson. Thirteen years of building constraint-based in silico models of escherichia coli. *Journal of bacteriology*, 185(9):2692–2699, 2003.

المنارة للاستشارات
www.manaraa.com

[249] Michael Reich, Thorin Tabor, Ted Liefeld, Helga Thorvaldsdóttir, Barbara Hill, Pablo Tamayo, and Jill P. Mesirov. The genepattern notebook environment. *Cell Systems*, 5(2):149–151.e1, 2017/08/23 2017.

[250] Michael Römer, Johannes Eichner, Andreas Dräger, Clemens Wrzodek, Finja Wrzodek, and Andreas Zell. Zbit bioinformatics toolbox: a web-platform for systems biology and expression data analysis. *PloS one*, 11(2):e0149263, 2016.

[251] HoHo Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

[252] Andreas Rossberg, Ben L. Titzer, Andreas Haas, Derek L. Schuff, Dan Gohman, Luke Wagner, Alon Zakai, J. F. Bastien, and Michael Holman. Bringing the web up to speed with webassembly. *Commun. ACM*, 61(12):107–115, November 2018.

[253] Marek Ruciński, Dario Izzo, and Francesco Biscani. On the impact of the migration topology on the island model. *Parallel Computing*, 36(10-11):555–571, 2010.

[254] Howard M Salis, Ethan A Mirsky, and Christopher A Voigt. Automated design of synthetic ribosome binding sites to control protein expression. *Nature Biotechnology*, 27(10):946–950, October 2009.

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

[256] Alejandro San Martín, Sebastián Ceballo, Iván Ruminot, Rodrigo Lerchundi, Wolf B Frommer, and Luis Felipe Barros. A genetically encoded fret lactate sensor and its use to detect the warburg effect in single cancer cells. *PloS one*, 8(2):e57712, 2013.

[257] BJ Sánchez, F Li, H Lu, EJ Kerkhoven, and J Nielsen. Sysbiochalmers/yeast-gem: yeast 8.2. 0, 2018.

[258] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten simple rules for reproducible computational research. *PLoS computational biology*, 9(10):e1003285, 2013.

[259] Debolina Sarkar, Thomas J Mueller, Deng Liu, Himadri B Pakrasi, and Costas D Maranas. A diurnal flux balance model of synechocystis sp. pcc 6803 metabolism. *PLoS computational biology*, 15(1):e1006692, 2019.

[260] Herbert Sauro. *Enzyme Kinetics for Systems Biology*. Ambrosius Publishing, 2012.

المنارة للاستشارات
www.manaraa.com

[261] Herbert Sauro. *Systems Biology: Introduction to Pathway Modeling*. Ambrosius Publishing, 2014.

[262] Herbert M Sauro and Frank T Bergmann. Standards and ontologies in computational systems biology. *Essays in biochemistry*, 45:211–222, 2008.

[263] Herbert S Sauro. Jdesigner: A simple biochemical network designer. *Available via the World Wide Web at http://members. tripod. co. uk/sauro/biotech. htm*, 2001.

[264] Michael A Savageau. Biochemical systems analysis. a study of function and design in molecular biology. In *ADDISON WESLEY PUBL.* 1976.

[265] James C Schaff, Dan Vasilescu, Ion I Moraru, Leslie M Loew, and Michael L Blinov. Rule-based modeling with virtual cell. *Bioinformatics*, 32(18):2880–2882, 2016.

[266] Martin Scharm and Vasundra Tour. COMBINE Archive Show Case. 6 2016.

[267] Martin Scharm and Dagmar Waltemath. A fully featured combine archive of a simulation study on syncytial mitotic cycles in drosophila embryos. *F1000Research*, 2016. [version 1; referees: 1 approved, 2 approved with reservations].

[268] Martin Scharm, Florian Wendland, Martin Peters, Markus Wolfien, Tom Theile, and Dagmar Waltemath. The combinearchiveweb application–a web based tool to handle files associated with modelling results. Technical report, PeerJ PrePrints, 2014.

[269] Jan Schellenberger, Richard Que, Ronan MT Fleming, Ines Thiele, Jeffrey D Orth, Adam M Feist, Daniel C Zielinski, Aarash Bordbar, Nathan E Lewis, Sorena Rahmanian, et al. Quantitative prediction of cellular metabolism with constraint-based models: the cobra toolbox v2. 0. *Nature protocols*, 6(9):1290, 2011.

[270] Patrick Schrauwen, Silvie Timmers, and Matthijs KC Hesselink. Blocking the entrance to open the gate. *Diabetes*, 62(3):703–705, 2013.

[271] Bruce E Shapiro and Eric Mjolsness. Pycellerator: an arrow-based reaction-like modelling language for biological simulations. *Bioinformatics*, page btv596, 2015.

[272] Helen Shen. Interactive notebooks: Sharing the code. *Nature*, 515(7525):151, 2014.

[273] Alexander A Shestov, Xiaojing Liu, Zheng Ser, Ahmad A Cluntun, Yin P Hung, Lei Huang, Dongsung Kim, Anne Le, Gary Yellen, John G Albeck, et al. Quantitative determinants of aerobic glycolysis identify flux through the enzyme gapdh as a limiting step. *Elife*, 3:e03342, 2014.

www.manaraa.com

[274] Tujin Shi, Mario Niepel, Jason E McDermott, Yuqian Gao, Carrie D Nicora, William B Chrisler, Lye M Markillie, Vladislav A Petyuk, Richard D Smith, Karin D Rodland, et al. Conservation of protein abundance patterns reveals the regulatory architecture of the egfr-mapk pathway. *Sci. Signal.*, 9(436):rs6–rs6, 2016.

[275] Yuhui Shi et al. Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 81–86. IEEE, 2001.

[276] C Sievert, C Parmer, T Hocking, S Chamberlain, K Ram, M Corvellec, and P Despouy. plotly: Create interactive web graphics via plotlys javascript graphing library [software], 2016.

[277] Larry A Sklar, Heinz Mueller, Geneva Omann, and Zenaida Oades. Three states for the formyl peptide receptor on intact cells. *Journal of Biological Chemistry*, 264(15):8483–8486, 1989.

[278] Kieran Smallbone, Hanan L Messiha, Kathleen M Carroll, Catherine L Winder, Naglis Malys, Warwick B Dunn, Ettore Murabito, Neil Swainston, Joseph O Dada, Farid Khan, et al. A model of yeast glycolysis based on a consistent kinetic characterisation of all its enzymes. *FEBS letters*, 587(17):2832–2841, 2013.

[279] Kieran Smallbone, Evangelos Simeonidis, Neil Swainston, and Pedro Mendes. Towards a genome-scale kinetic model of cellular metabolism. *BMC systems biology*, 4(1):6, 2010.

[280] Adam M Smith, Wen Xu, Yao Sun, James R Faeder, and G Elisabeta Marai. Rulebender: integrated modeling, simulation and visualization for rule-based intracellular biochemistry. *BMC bioinformatics*, 13(8):S3, 2012.

[281] Lucian P Smith, Frank T Bergmann, Deepak Chandran, and Herbert M Sauro. Antimony: a modular model definition language. *Bioinformatics*, 25(18):2452–2454, 2009.

[282] Michael W Sneddon, James R Faeder, and Thierry Emonet. Efficient modeling, simulation and coarse-graining of biological complexity with nfsim. *Nature methods*, 8(2):177, 2011.

[283] Endre T Somogyi, Jean-Marie Bouteiller, James A Glazier, Matthias König, J Kyle Medley, Maciej H Swat, and Herbert M Sauro. libroadrunner: a high performance sbml simulation and analysis library. *Bioinformatics*, 31(20):3315–3321, 2015.

المنارة للاستشارات
www.manaraa.com

[284] Bernhard Steiert, Clemens Kreutz, Andreas Raue, and Jens Timmer. Recipes for analysis of molecular networks using the data2dynamics modeling environment. In *Modeling Biomolecular Site Dynamics*, pages 341–362. Springer, 2019.

[285] Ulrich Stelzl, Uwe Worm, Maciej Lalowski, Christian Haenig, Felix H Brembeck, Heike Goehler, Martin Stroedicke, Martina Zenkner, Anke Schoenherr, Susanne Koeppen, et al. A human protein-protein interaction network: a resource for annotating the proteome. *Cell*, 122(6):957–968, 2005.

[286] Victoria Stodden, Jennifer Seiler, and Zhaokun Ma. An empirical analysis of journal policy effectiveness for computational reproducibility. *Proceedings of the National Academy of Sciences*, 115(11):2584–2589, 2018.

[287] Lewi Stone and Alan Roberts. Competitive exclusion, or species aggregation? *Oecologia*, 91(3):419–424, 1992.

[288] Lewi Stone, Daniel Simberloff, and Yael Artzy-Randrup. Network motifs and their origins. *PLOS Computational Biology*, 15(4):e1006749, April 2019.

[289] Lewis Stone. *Some problems of community ecology: processes, patterns and species persistence in ecosystems.* PhD thesis, Monash University, 1988.

[290] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[291] Andrew Stuart. The legacy of rudolph kalman. Boeing Distinguished Colloquia, Univ. Washington, 2019.

[292] Anandh Swaminathan, Victoria Hsiao, and Richard M Murray. Quantitative modeling of integrase dynamics using a novel python toolbox for parameter inference in synthetic biology. *bioRxiv*, 2017.

[293] Maciej H Swat, Gilberto L Thomas, Julio M Belmonte, Abbas Shirinifard, Dimitrij Hmeljak, and James A Glazier. Multi-scale modeling of tissues using compucell3d. *Methods in cell biology*, 110:325, 2012.

[294] Andrea Tangherloni, Marco S Nobile, Daniela Besozzi, Giancarlo Mauri, and Paolo Cazzaniga. Lassie: simulating large-scale models of biochemical systems on gpus. *BMC bioinformatics*, 18(1):246, 2017.

المنارة للاستشارات

www.manaraa.com

[295] Lukas Bahati Tanner, Alexander G Goglia, Monica H Wei, Talen Sehgal, Lance R Parsons, Junyoung O Park, Eileen White, Jared E Toettcher, and Joshua D Rabinowitz. Four key steps control glycolytic flux in mammalian cells. *Cell systems*, 7(1):49–62, 2018.

[296] E. T. Tatro, S. Hefler, S. Shumaker-Armstrong, B. Soontornniyomkij, M. Yang, A. Yermanos, N. Wren, D. J. Moore, and C. L. Achim. Modulation of bk channel by microrna-9 in neurons after exposure to hiv and methamphetamine. *J Neuroimmune Pharmacol*, 2013. Tatro, Erick T Hefler, Shannon Shumaker-Armstrong, Stephanie Soontornniyomkij, Benchawanna Yang, Michael Yermanos, Alex Wren, Nina Moore, David J Achim, Cristian L R03 DA031591/DA/NIDA NIH HHS/United States U19 AI096113/AI/NIAID NIH HHS/United States Journal article Journal of neuroimmune pharmacology : the official journal of the Society on NeuroImmune Pharmacology J Neuroimmune Pharmacol. 2013 Mar 19.

[297] Michael Bedford Taylor. The evolution of bitcoin hardware. *Computer*, 50(9):58–66, 2017.

[298] Bas Teusink, Jutta Passarge, Corinne A Reijenga, Eugenia Esgalhado, Coen C Van der Weijden, Mike Schepper, Michael C Walsh, Barbara M Bakker, Karel Van Dam, Hans V Westerhoff, et al. Can yeast glycolysis be understood in terms of in vitro kinetics of the constituent enzymes? testing biochemistry. *European Journal of Biochemistry*, 267(17):5313–5329, 2000.

[299] Matthew Theisen. Introducing py_emra: the python module for ensemble modeling robustness analysis. *bioRxiv*, 2016.

[300] Tianhai Tian and Kevin Burrage. Binomial leap methods for simulating stochastic chemical kinetics. *The Journal of chemical physics*, 121(21):10356–10364, 2004.

[301] Franck Toledo and Geoffrey M. Wahl. Regulating the p53 pathway: in vitro hypotheses, in vivo veritas. *Nature Reviews Cancer*, 6(12):909–923, December 2006.

[302] Mark K. Transtrum, Benjamin B. Machta, Kevin S. Brown, Bryan C. Daniels, Christopher R. Myers, and James P. Sethna. Perspective: Sloppiness and emergent theories in physics, biology, and beyond. *The Journal of Chemical Physics*, 143(1):010901, July 2015.

[303] Thomas E Turner, Santiago Schnell, and Kevin Burrage. Stochastic approaches for modelling in vivo reactions. *Computational biology and chemistry*, 28(3):165–178, 2004.

المنارة للاستشارات
www.manaraa.com

[304] Piet H Van der Graaf and Neil Benson. Systems pharmacology: bridging systems biology and pharmacokinetics-pharmacodynamics (pkpd) in drug discovery and development. *Pharmaceutical research*, 28(7):1460–1464, 2011.

[305] Jan van der Greef and Robert N McBurney. Rescuing drug discovery: in vivo systems pathology and systems pharmacology. *Nature Reviews Drug Discovery*, 4(12):961, 2005.

[306] Johan H van Heerden, Frank J Bruggeman, and Bas Teusink. Multi-tasking of biosynthetic and energetic functions of glycolysis explained by supply and demand logic. *BioEssays*, 37(1):34–45, 2015.

[307] David D Van Niekerk, Gerald P Penkler, Francois du Toit, and Jacky L Snoep. Targeting glycolysis in the malaria parasite plasmodium falciparum. *The FEBS journal*, 283(4):634–646, 2016.

[308] Natal AW van Riel. Dynamic modelling and analysis of biochemical networks: mechanism-based models and model-based experiments. *Briefings in bioinformatics*, 7(4):364–374, 2006.

[309] Guido Van Rossum. *Python Library Reference*. CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, 1995.

[310] Alejandro F Villaverde, Fabian Fröhlich, Daniel Weindl, Jan Hasenauer, and Julio R Banga. Benchmarking optimization methods for parameter estimation in large kinetic models. *Bioinformatics*, 35(5):830–838, 2018.

[311] Alejandro F Villaverde, David Henriques, Kieran Smallbone, Sophia Bongard, Joachim Schmid, Damjan Cicin-Sain, Anton Crombach, Julio Saez-Rodriguez, Klaus Mauch, Eva Balsa-Canto, et al. Biopredyn-bench: a suite of benchmark problems for dynamic modelling in systems biology. *BMC systems biology*, 9(1):8, 2015.

[312] PK Vinod, Paula Freire, Ahmed Rattani, Andrea Ciliberto, Frank Uhlmann, and Bela Novak. Computational modelling of mitotic exit in budding yeast: the role of separase and cdc14 endocycles. *Journal of The Royal Society Interface*, 8(61):1128–1141, 2011.

[313] Dagmar Waltemath, Richard Adams, Frank T Bergmann, Michael Hucka, Fedor Kolpakov, Andrew K Miller, Ion I Moraru, David Nickerson, Sven Sahle, Jacky L Snoep, et al. Reproducible computational biology experiments with sed-ml-the simulation experiment description markup language. *BMC systems biology*, 5(1):198, 2011.

[314] Dagmar Waltemath and Olaf Wolkenhauer. How modeling standards, software, and initiatives support reproducibility in systems biology and systems medicine. *IEEE Transactions on Biomedical Engineering*, 63(10):1999–2006, 2016.

المنارة للاستشارات

www.manaraa.com

[315] Junwei Wang, Jiajun Zhang, Zhanjiang Yuan, and Tianshou Zhou. Noise-induced switches in network systems of the genetic toggle switch. *BMC systems biology*, 1(1):50, 2007.

[316] Otto Warburg. On the origin of cancer cells. *Science*, 123(3191):309–314, 1956.

[317] Leandro H Watanabe and Chris J Myers. Hierarchical stochastic simulation algorithm for sbml models of genetic circuits. *Frontiers in bioengineering and biotechnology*, 2:55, 2014.

[318] Duncan J Watts and Steven H Strogatz. Collective dynamics of “small-world” networks. *nature*, 393(6684):440, 1998.

[319] Tomasz Adam Wesolowski and Arieh Warshel. Frozen density functional approach for ab initio calculations of solvated molecules. *The Journal of Physical Chemistry*, 97(30):8050–8053, 1993.

[320] Arthur T Winfree. Biological rhythms and the behavior of populations of coupled oscillators. *Journal of theoretical biology*, 16(1):15–42, 1967.

[321] Jana Wolf, Ho-Yong Sohn, Reinhart Heinrich, and Hiroshi Kuriyama. Mathematical analysis of a mechanism for autonomous metabolic oscillations in continuous culture of saccharomyces cerevisiae. *FEBS letters*, 499(3):230–234, 2001.

[322] Stephen Wolfram. *Mathematica*. Cambridge University Press, 1996.

[323] Sung Sik Woo. *Fast Simulation of Stochastic Biochemical Reaction Networks on Cytomorphic Chips*. PhD thesis, Massachusetts Institute of Technology, 2016.

[324] Sung Sik Woo, Jaewook Kim, and Rahul Sarpeshkar. A cytomorphic chip for quantitative modeling of fundamental bio-molecular circuits. *IEEE Transactions on Biomedical Circuits and Systems*, 9(4):527–542, 2015.

[325] Sung Sik Woo, Jaewook Kim, and Rahul Sarpeshkar. A digitally programmable cytomorphic chip for simulation of arbitrary biochemical reaction networks. *IEEE transactions on biomedical circuits and systems*, 12(2):360–378, 2018.

[326] Jamey D Young, Kristene L Henne, John A Morgan, Allan E Konopka, and Doraiswami Ramkrishna. Integrating cybernetic modeling with pathway analysis provides a dynamic, systems-level description of metabolic control. *Biotechnology and bioengineering*, 100(3):542–559, 2008.

المنارة للاستشارات

www.manaraa.com

[327] James T Yurkovich, Daniel C Zielinski, Laurence Yang, Giuseppe Paglia, Ottar Rolfsson, Ólafur E Sigurjónsson, Jared T Broddrick, Aarash Bordbar, Kristine Wichuk, Sigurur Brynjólfsson, et al. Quantitative time-course metabolomics in human red blood cells reveal the temperature dependence of human metabolic networks. *Journal of Biological Chemistry*, 292(48):19556–19564, 2017.

[328] Alon Zakai. Emscripten: an llvm-to-javascript compiler. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pages 301–312. ACM, 2011.

[329] Carolyn Zhang, Ryan Tsoi, Feilun Wu, and Lingchong You. Processing oscillatory signals by incoherent feedforward loops. *PLOS Computational Biology*, 12(9):e1005101, September 2016.

[330] Fengkai Zhang and Martin Meier-Schellersheim. Sbml level 3 package: multistate, multicomponent and multicompartment species, version 1, release 1. *Journal of integrative bioinformatics*, 15(1), 2018.

[331] Ende Zhao, Tomasz Maj, Ilona Kryczek, Wei Li, Ke Wu, Lili Zhao, Shuang Wei, Joel Crespo, Shanshan Wan, Linda Vatan, et al. Cancer mediates effector t cell dysfunction by targeting micrornas and ezh2 via glycolysis restriction. *Nature immunology*, 17(1):95, 2016.

[332] Qi Zhao, Arion I. Stettner, Ed Reznik, Ioannis Ch. Paschalidis, and Daniel Segrè. Mapping the landscape of metabolic goals of a cell. *Genome Biology*, 17(1), May 2016.

المنارة للاستشارات

www.manaraa.com

Appendix A

LIST OF SOFTWARE PROJECTS

Listed project based on authorship or major contributions.

- Distributed island model fitter.

  https://github.com/distrib-dyn-modeling/sabaody

- Cytomorphic compiler.

  https://github.com/0u812/mtt

- Tellurium.

  https://github.com/sys-bio/tellurium

- Tellurium examples.

  https://github.com/sys-bio/tellurium-examples

- Tellurium notebook examples.

  https://github.com/sys-bio/tellurium-notebooks

- libRoadRunner.

  https://github.com/sys-bio/roadrunner

- Network layout tool.

المنارة للاستشارات
www.manaraa.com

`https://github.com/sys-bio/sbnw`

- Python package from constructing SBML models.

  `https://github.com/sys-bio/simplesbml`

- Package for converting from SBML models to Matlab.

  `https://github.com/sys-bio/sbml2matlab`

المنـــارة للاستشارات

www.manaraa.com

VITA

J Kyle Medley

Table A.1: Peer–reviewed First–author Publications

A compiler for biological networks on silicon chips	*In preparation.*
Accelerated Biochemical Model Fitting via the Asynchronous, Generalized Island Method	*In preparation.*
libsbmljs — Enabling Web–Based SBML Tools	*In preparation.*
Tellurium notebooks — An environment for reproducible dynamical modeling in systems biology	Published (*PLoS Comp Biol.* 2018) [196]
Guidelines for reproducibly building and simulating systems biology models	Published (*IEEE Trans Biomed Eng.* 2016) [197]

Table A.2: Peer–reviewed Co–author Publications

Tellurium: An extensible Python-based modeling environment for systems and synthetic biology	Published (*Biosystems* 2018) [66]
Synthetic Biology: Engineering Living Systems from Biophysical Principles	Published (*Biophys J.* 2017) [31]
libRoadRunner: a high performance SBML simulation and analysis library	Published (*Bioinformatics* 2015) [283]

المنارة للاستشارات
www.manaraa.com

Table A.3: Conference Talks

Bringing Combine Standards to the Literate Notebook World	COMBINE 2018 http://co.mbine.org/events/COMBINE_2018/agenda
Tellurium 2.0: Revamped Notebook Interface & COMBINE Archive Support	Harmony 2017
High-performance Model Simulation with libRoadRunner	COMBINE 2016 http://co.mbine.org/events/COMBINE_2016/agenda

Table A.4: Posters

Building Reproducible Dynamical Models with Tellurium 2.0: A Case Study	*IMAG 10th Anniversary Conference* 2017
Distributed, Evolutionary Computing for Parameter Fitting of Stochastic Models	*Beacon* 2017
Building Hybrid/Hierarchical Models with Tellurium and libRoadRunner	*Whole Cell Summer School* 2016 http://www.wholecell.org/v2/school-2016/
LibRoadRunner: High-Performance Time-course Simulation and Model Fitting	*COMBINE* 2015
Tellurium : A Python Based Integrated Environment for Systems Biology	*Harmony* 2015

المنارة للاستشارات

www.manaraa.com

Table A.5: Preprints (First / Coauthor)

libsbmljs — Enabling Web–Based SBML Tools	*bioRxiv* 2018
Tellurium notebooks — An environment for reproducible dynamical modeling in systems biology	*bioRxiv* 2017
A portable library to support the SBML Layout Extension	*bioRxiv* 2016
SimpleSBML: A Python package for creating and editing SBML models	*bioRxiv* 2016

المنارة للاستشارات
www.manaraa.com