



Lan K. Nguyen *Editor*

Computational Modeling of Signaling Networks

METHODS IN MOLECULAR BIOLOGY

Series Editor

John M. Walker

School of Life and Medical Sciences
University of Hertfordshire Hatfield,
Hertfordshire, UK

For further volumes:
<http://www.springer.com/series/7651>

For over 35 years, biological scientists have come to rely on the research protocols and methodologies in the critically acclaimed *Methods in Molecular Biology* series. The series was the first to introduce the step-by-step protocols approach that has become the standard in all biomedical protocol publishing. Each protocol is provided in readily-reproducible step-by-step fashion, opening with an introductory overview, a list of the materials and reagents needed to complete the experiment, and followed by a detailed procedure that is supported with a helpful notes section offering tips and tricks of the trade as well as troubleshooting advice. These hallmark features were introduced by series editor Dr. John Walker and constitute the key ingredient in each and every volume of the *Methods in Molecular Biology* series. Tested and trusted, comprehensive and reliable, all protocols from the series are indexed in PubMed.

Computational Modeling of Signaling Networks

Edited by

Lan K. Nguyen

Biomedicine Discovery Institute, Monash University, Clayton, VIC, Australia



Editor

Lan K. Nguyen
Biomedicine Discovery Institute
Monash University
Clayton, VIC, Australia

ISSN 1064-3745

ISSN 1940-6029 (electronic)

Methods in Molecular Biology

ISBN 978-1-0716-3007-5

ISBN 978-1-0716-3008-2 (eBook)

<https://doi.org/10.1007/978-1-0716-3008-2>

© Springer Science+Business Media, LLC, part of Springer Nature 2023

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Humana imprint is published by the registered company Springer Science+Business Media, LLC, part of Springer Nature.

The registered company address is: 1 New York Plaza, New York, NY 10004, U.S.A.

Preface

A comprehensive understanding of how signalling networks behave in space and time to generate biological specificity is of paramount importance to biology and medicine. However, because of the marked complexity of signalling network interactions and significant heterogeneity in network state across different cellular contexts (cell types and patients), this goal is unlikely to be achieved through intuitive reasoning or experimentation alone. Mathematical and computational modelling have emerged as essential tools that synergise with biological experimentation to address these issues. Computational models of signalling networks provide valuable quantitative platforms that allow us to: (i) gain a systems-level understanding of network interactions and unravel emergent systems properties; (ii) integrate and explain complex, and often counterintuitive, datasets that are frequently of a heterogeneous nature; and (iii) make testable predictions about cellular responses to perturbations such as targeted therapies, thereby guiding the rational design of experimentation. Moreover, because of the nonlinearity and dynamic nature of the genotype–phenotype relationships, it is difficult to predict optimal therapeutic interventions from knowing a cancer’s mutation profile. In this regard, the power of computational models to identify key components and vulnerable nodes within the networks also lends itself to the identification of novel, non-trivial drug targets and therapeutic strategies, such as the optimisation of combination therapies against cancer.

With computational modelling taking the central stage in systems signalling research, this book volume highlights state-of-the-art developments in the areas of computational modelling, simulation and integrative analysis of cell signalling networks. The volume is organised into three main themes. The first comprises chapters describing new advances in computational techniques and methods that facilitate the modelling and model-based analysis of biochemical networks, ranging from small signalling circuits to large-sized systems. This theme also includes chapters that provide an overview and practical guide to key topics in network modelling, such as parameter estimation, model calibration and reproducible modelling, written in a comprehensive and accessible manner. Since a model is only as good as its input data, the true power of computational modelling is maximised when it is complemented by suitable and high-quality experimental data. With this in mind, the second theme of this book highlights new developments in a range of exciting experimental techniques that help generate dynamic, quantitative and systems-level measurements of cell signalling at the single-cell, bulk and even mixed-cell population levels, which are all highly relevant for computational modelling. These chapters are written by expert scientists whose research focuses not only on developing such techniques but also on applying them to the mathematical modelling of specific biological systems. The seamless integration between experimentation and modelling, a hallmark of the systems biology approach, is therefore expertly showcased by these chapters. The third and final theme emphasises practical applications of computational modelling and integrative analysis of signal transduction networks in diseases. Specifically, it focuses on understanding the impact of oncogenic mutations and targeted therapies on cancer signalling behaviour, and the discovery of new therapeutic regimens for cancer.

This book aims to benefit a wide range of readers, including researchers beyond the computational and biological systems biology communities. The majority of the chapters are

written in an accessible manner and pedagogical manner, and so are highly suitable for new learners in the field. Moreover, the depth and breadth of the chapters covering new developments in both the computational and experimental fronts also mean the book will be of interest to more advanced researchers, enabling them to learn how novel computational and integrative modelling approaches are currently being applied to research in signalling and cancer. While this book is by no means exhaustive, I do hope it will serve as a catalyst for achieving this goal.

I extend my sincere gratitude to Professor John Walker, the series editor, for entrusting me with the task of editing this volume. I am also grateful to Springer Nature and their team members, Patrick Marton, David Casey, and Anna Rakovsky for their unwavering support. The success of this book would not have been possible without the dedication and tireless efforts of all the contributing authors, especially considering the challenging circumstances of the COVID-19 pandemic. I wholeheartedly appreciate their expertise, valuable contributions, and remarkable patience throughout the process.

Clayton, VIC, Australia

Lan K. Nguyen

Contents

Preface	v
Contributors	ix

PART I ADVANCES IN COMPUTATIONAL MODELLING OF SIGNALLING NETWORKS

1 Design Principles Underlying Robust Adaptation of Complex Biochemical Networks	3
<i>Robyn P. Araujo and Lance A. Liotta</i>	
2 Multi-Dimensional Analysis of Biochemical Network Dynamics Using pyDYVIPAC	33
<i>Yunduo Lan and Lan K. Nguyen</i>	
3 A Practical Guide for the Efficient Formulation and Calibration of Large, Energy- and Rule-Based Models of Cellular Signal Transduction	59
<i>Fabian Fröhlich</i>	
4 Systems Biology: Identifiability Analysis and Parameter Identification via Systems-Biology-Informed Neural Networks	87
<i>Mitchell Danecker, Zhen Zhang, George Em Karniadakis, and Lu Lu</i>	
5 A Practical Guide to Reproducible Modeling for Biochemical Networks	107
<i>Veronica L. Porubsky and Herbert M. Sauro</i>	
6 Integrating Multi-Omics Data to Construct Reliable Interconnected Models of Signaling, Gene Regulatory, and Metabolic Pathways	139
<i>Krishna Kumar, Debaleena Bhowmik, Sapan Mandloi, Anupam Gautam, Abhishek Lahiri, Nupur Biswas, Sandip Paul, and Saikat Chakrabarti</i>	
7 Efficient Quantification of Extrinsic Fluctuations via Stochastic Simulations	153
<i>Tagari Samanta and Sandip Kar</i>	
8 Meta-Dynamic Network Modelling for Biochemical Networks	167
<i>Anthony Hart and Lan K. Nguyen</i>	
9 Rapid Particle-Based Simulations of Cellular Signalling with the FLAME-Accelerated Signalling Tool (FaST) and GPUs	191
<i>Gavin Fullstone</i>	

PART II ADVANCES IN INTEGRATIVE ANALYSIS OF SIGNALLING NETWORKS

10 Modeling Cellular Signaling Variability Based on Single-Cell Data: The TGF β -SMAD Signaling Pathway	215
<i>Uddipan Sarma, Lorenz Ripka, Uchenna Alex Anyaegbunam, and Stefan Legewie</i>	

11	Quantitative Imaging Analysis of NF-κB for Mathematical Modeling Applications	253
	<i>Johannes Nicolaus Wibisana, Takehiko Inaba, Yasushi Sako, and Mariko Okada</i>	
12	Resolving Crosstalk Between Signaling Pathways Using Mathematical Modeling and Time-Resolved Single Cell Data	267
	<i>Fabian Konrath, Alexander Loewer, and Jana Wolf</i>	
13	Live-Cell Sender-Receiver Co-cultures for Quantitative Measurement of Paracrine Signaling Dynamics, Gene Expression, and Drug Response	285
	<i>Michael Pargett, Abhineet R. Ram, Vaibhav Murthy, and Alexander E. Davies</i>	
14	Application of Optogenetics to Probe the Signaling Dynamics of Cell Fate Decision-Making	315
	<i>Heath E. Johnson</i>	

PART III APPLICATION OF INTEGRATIVE MODELLING AND ANALYSIS
OF SIGNALLING NETWORKS IN DISEASES

15	Computational Random Mutagenesis to Investigate RAS Mutant Signaling	329
	<i>Edward C. Stites</i>	
16	Mathematically Modeling the Effect of Endocrine and Cdk4/6 Inhibitor Therapies on Breast Cancer Cells.....	337
	<i>Wei He, Ayesha N. Shajahan-Haq, and William T. Baumann</i>	
17	SynDISCO: A Mechanistic Modeling-Based Framework for Predictive Prioritization of Synergistic Drug Combinations Targeting Cell Signalling Networks.....	357
	<i>Sung-Young Shin and Lan K. Nguyen</i>	
	<i>Index</i>	383

Contributors

- UCHENNA ALEX ANYAEGBUNAM • *Institute of Molecular Biology (IMB), Mainz, Germany; Department of Systems Biology, Institute for Biomedical Genetics, University of Stuttgart, Stuttgart, Germany*
- ROBYN P. ARAUJO • *School of Mathematical Sciences, Queensland University of Technology, Brisbane, QLD, Australia; Institute of Health and Biomedical Innovation (IHBI), Kelvin Grove, QLD, Australia*
- WILLIAM T. BAUMANN • *Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA*
- DEBALEENA BHOWMIK • *Structural Biology and Bioinformatics Division, Council for Scientific and Industrial Research (CSIR) - Indian Institute of Chemical Biology (IICB), Kolkata, West Bengal, India; Academy of Scientific and Innovative Research (AcSIR), Ghaziabad, Uttar Pradesh, India*
- NUPUR BISWAS • *Structural Biology and Bioinformatics Division, Council for Scientific and Industrial Research (CSIR) - Indian Institute of Chemical Biology (IICB), Kolkata, West Bengal, India*
- SAIKAT CHAKRABARTI • *Structural Biology and Bioinformatics Division, Council for Scientific and Industrial Research (CSIR) - Indian Institute of Chemical Biology (IICB), Kolkata, West Bengal, India; Academy of Scientific and Innovative Research (AcSIR), Ghaziabad, Uttar Pradesh, India*
- MITCHELL DANEKER • *Department of Chemical and Biomolecular Engineering, University of Pennsylvania, Philadelphia, PA, USA*
- ALEXANDER E. DAVIES • *Department of Veterinary Biosciences, College of Veterinary Medicine, The Ohio State University, Columbus, OH, USA; Knight Cancer Institute, Cancer Early Detection Advanced Research Center, Oregon Health & Science University, Portland, OR, USA*
- FABIAN FRÖHLICH • *Department of Systems Biology, Harvard Medical School, Boston, MA, USA*
- GAVIN FULLSTONE • *Institute for Cell Biology and Immunology, University of Stuttgart, Stuttgart, Germany; Stuttgart Research Center Systems Biology, University of Stuttgart, Stuttgart, Germany*
- ANUPAM GAUTAM • *Algorithms in Bioinformatics, Institute for Bioinformatics and Medical Informatics, University of Tübingen, Tübingen, Germany; International Max Planck Research School “From Molecules to Organisms,” Max Planck Institute for Biology Tübingen, Tübingen, Germany; Cluster of Excellence: EXC 2124: Controlling Microbes to Fight Infection, University of Tübingen, Tübingen, Germany*
- ANTHONY HART • *Department of Biochemistry and Molecular Biology, School of Biomedical Sciences, Monash University, Clayton, VIC, Australia*
- WEI HE • *Program in Genetics, Bioinformatics, and Computational Biology, VT BIOTRANS, Virginia Tech, Blacksburg, VA, USA*
- TAKEHIKO INABA • *Cellular Informatics Laboratory, RIKEN Cluster for Pioneering Research, Hirosawa, Wako, Japan*
- HEATH E. JOHNSON • *School of Biological Sciences, The University of Hong Kong, Hong Kong Special Administrative Region, Hong Kong, China*

- SANDIP KAR • *Department of Chemistry, IIT Bombay, Mumbai, India*
- GEORGE EM KARNIADAKIS • *Division of Applied Mathematics, Brown University, Providence, RI, USA; School of Engineering, Brown University, Providence, RI, USA*
- FABIAN KONRATH • *Mathematical Modelling of Cellular Processes, Max Delbrueck Center for Molecular Medicine, Berlin, Germany*
- KRISHNA KUMAR • *Structural Biology and Bioinformatics Division, Council for Scientific and Industrial Research (CSIR) - Indian Institute of Chemical Biology (IICB), Kolkata, West Bengal, India*
- ABHISHAKE LAHIRI • *Structural Biology and Bioinformatics Division, Council for Scientific and Industrial Research (CSIR) - Indian Institute of Chemical Biology (IICB), Kolkata, West Bengal, India; Academy of Scientific and Innovative Research (AcSIR), Ghaziabad, Uttar Pradesh, India*
- YUNDUO LAN • *Department of Biochemistry and Molecular Biology, School of Biomedical Sciences, Monash University, Clayton, VIC, Australia; Biomedicine Discovery Institute, Monash University, Clayton, VIC, Australia*
- STEFAN LEGEWIE • *Institute of Molecular Biology (IMB), Mainz, Germany; Department of Systems Biology, Institute for Biomedical Genetics, University of Stuttgart, Stuttgart, Germany; Stuttgart Research Center for Systems Biology, University of Stuttgart, Stuttgart, Germany*
- LANCE A. LIOTTA • *Center for Applied Proteomics and Molecular Medicine, George Mason University, Manassas, VA, USA*
- ALEXANDER LOEWER • *Systems Biology of the Stress Response, Department of Biology, Technische Universität Darmstadt, Darmstadt, Germany*
- LU LU • *Department of Chemical and Biomolecular Engineering, University of Pennsylvania, Philadelphia, PA, USA*
- SAPAN MANDLOI • *Structural Biology and Bioinformatics Division, Council for Scientific and Industrial Research (CSIR) - Indian Institute of Chemical Biology (IICB), Kolkata, West Bengal, India*
- VAIBHAV MURTHY • *Department of Veterinary Biosciences, College of Veterinary Medicine, The Ohio State University, Columbus, OH, USA; Knight Cancer Institute, Cancer Early Detection Advanced Research Center, Oregon Health & Science University, Portland, OR, USA*
- LAN K. NGUYEN • *Department of Biochemistry and Molecular Biology, School of Biomedical Sciences, Monash University, Clayton, VIC, Australia; Biomedicine Discovery Institute, Monash University, Clayton, VIC, Australia*
- MARIKO OKADA • *Institute for Protein Research, Osaka University, Suita, Osaka, Japan; Center for Drug Design and Research, National Institutes of Biomedical Innovation, Health and Nutrition, Ibaraki, Osaka, Japan*
- MICHAEL PARGETT • *Department of Molecular and Cellular Biology, University of California, Davis, Davis, CA, USA*
- SANDIP PAUL • *JIS Institute of Advanced Studies and Research, JIS University, Kolkata, India*
- VERONICA L. PORUBSKY • *University of Washington, Department of Bioengineering, Seattle, WA, USA*
- ABHINEET R. RAM • *Department of Molecular and Cellular Biology, University of California, Davis, Davis, CA, USA*

LORENZ RIPKA • *Institute of Molecular Biology (IMB), Mainz, Germany; Department of Systems Biology, Institute for Biomedical Genetics, University of Stuttgart, Stuttgart, Germany*

YASUSHI SAKO • *Cellular Informatics Laboratory, RIKEN Cluster for Pioneering Research, Hirosawa, Wako, Japan*

TAGARI SAMANTA • *Department of Chemistry, IIT Bombay, Mumbai, India*

UDDIPAN SARMA • *Institute of Molecular Biology (IMB), Mainz, Germany*

HERBERT M. SAURO • *University of Washington, Department of Bioengineering, Seattle, WA, USA*

AYESHA N. SHAJAHAN-HAQ • *Department of Oncology, Lombardi Comprehensive Cancer Center, Georgetown University Medical Center, Washington, DC, USA*

SUNG-YOUNG SHIN • *Department of Biochemistry and Molecular Biology, School of Biomedical Sciences, Monash University, Clayton, VIC, Australia; Biomedicine Discovery Institute, Monash University, Clayton, VIC, Australia*

EDWARD C. STITES • *Integrative Biology Laboratory, Salk Institute for Biological Studies, La Jolla, CA, USA*

JOHANNES NICOLAUS WIBISANA • *Institute for Protein Research, Osaka University, Suita, Osaka, Japan*

JANA WOLF • *Mathematical Modelling of Cellular Processes, Max Delbrueck Center for Molecular Medicine, Berlin, Germany; Mathematical Modelling of Cellular Processes, Department of Mathematics and Computer Science, Free University Berlin, Berlin, Germany*

ZHEN ZHANG • *Division of Applied Mathematics, Brown University, Providence, RI, USA*

Part I

Advances in Computational Modelling of Signalling Networks



Chapter 1

Design Principles Underlying Robust Adaptation of Complex Biochemical Networks

Robyn P. Araujo and Lance A. Liotta

Abstract

Biochemical networks are often characterized by tremendous complexity—both in terms of the sheer number of interacting molecules (“nodes”) and in terms of the varied and incompletely understood interactions among these molecules (“interconnections” or “edges”). Strikingly, the vast and intricate networks of interacting proteins that exist within each living cell have the capacity to perform remarkably robustly, and reproducibly, despite significant variations in concentrations of the interacting components from one cell to the next and despite mutability over time of biochemical parameters. Here we consider the ubiquitously observed and fundamentally important signalling response known as robust perfect adaptation (RPA). We have recently shown that all RPA-capable networks, even the most complex ones, must satisfy an extremely rigid set of design principles, and are modular, being decomposable into just two types of network building-blocks—opposer modules and balancer modules. Here we present an overview of the design principles that characterize all RPA-capable network topologies through a detailed examination of a collection of simple examples. We also introduce a diagrammatic method for studying the potential of a network to exhibit RPA, which may be applied without a detailed knowledge of the complex mathematical principles governing RPA.

Key words Robust perfect adaptation, Complexity, Chemical reaction networks, Robustness, Network topology

1 Introduction

The coexistence of both complexity and robustness in the self-organizing, self-regulating networks arising in nature represents an extraordinary paradox [1–3]. Indeed, since robustness in the face of changing and unpredictable environments is one of the most fundamental requirements for any living system, this raises a deep question about nature’s most basic design principles: How must biological complexity be organized to accommodate the exacting demands of robust performance?

In this chapter we consider this question in the light of the keystone biological function known as robust perfect adaptation (RPA). RPA is the process whereby a system resets selected internal

components to their respective pre-stimulus baseline levels (or “set points”) following a disturbance or altered input, with no need for fine-tuning of system parameters [4–6]. The capacity for RPA is widely considered to be an essential characteristic of all evolvable and self-regulating systems [5]. It has been ubiquitously observed in biology at all levels of organization, from intracellular networks comprising genes, metabolites, and/or proteins to signalling networks at the whole-organism level, including calcium or hormone regulation, vision, olfaction, touch, and embryonic morphogenesis [7–14]. Dysregulation of RPA is also thought to play a central role in human disease, since maladaptation—the establishing of harmful, disease-promoting RPA set points—is thought to be a signalling feature underpinning disorders such as drug addiction, chronic pain, cancer progression, and metabolic syndrome (e.g., obesity, high blood pressure, and insulin resistance) [15–20].

Early theoretical work in the RPA field focused on a number of relatively simple RPA-capable network designs, which have attracted widespread attention from the scientific community and have been analyzed extensively. Ma et al. [21], for instance, determined that for networks comprising just three interacting nodes, all network arrangements capable of exhibiting RPA to a persistent change in network input can be divided into two well-defined classes—negative feedback loop with buffer node (NFBLB) and incoherent feedforward loop with proportioner node (IFLPN). A highly influential four-node RPA motif, proposed independently of the work of Ma et al., is the antithetic integral control model [22, 23]. Recent work has also considered simple RPA motifs in models that account for the dilution of cellular constituents during the cell’s growth phase in order to extend the applicability of existing RPA theory in synthetic biology settings [24, 25].

While the study of these small and very specific RPA-promoting network designs has given rise to many important applications of RPA theory in the synthetic biology field [23, 25], such simple models are ill equipped to provide insight into the highly complex network designs through which RPA may be realized in nature—that is, in complex *self-organizing*, *self-regulating*, and highly mutable systems such as cellular signal transduction networks. It is clear that complex biosystems differ in fundamental ways from engineering control systems and are comprised of elements that must serve both as the transmitted signals *and their own controllers*. Unlike their designed counterparts in engineering control systems, bionetworks do not have the luxury of employing specially designed, dedicated components whose purpose is to sense or control biochemical signals. Moreover, these vast and intricate biosystems are typically called upon to solve a large number of “cognitive problems” in parallel—processing and interpreting a high-dimensional space of biochemical and mechanical stimuli and making decisions as to appropriate systems-level responses. In this context, signalling

proteins that play a *regulatory role* (reduplicating a model of the dynamic structure of the input signal) in a *feedback* path can also simultaneously play a transmissive role (in a “route” leading from input node to output node). Such ambiguous roles for signalling componentry may not be needed, and may not even be appropriate, in an engineering design context.

An understanding of how RPA could be implemented in complex bionetworks amid such strenuous cognitive demands requires access to a “complete” design space for *all possible* RPA-capable network topologies. We recently developed a new mathematical approach to solve this general “RPA problem” [5], and as a consequence, the full set of all possible RPA-capable network configurations is now known—for any-sized network, for any degree of complexity (network size and interconnectedness), for any network “type” (e.g., protein network, gene regulatory network, intercellular communication network, neuronal network), and any adaptive timescale. In particular, we have demonstrated two classes of network topologies—the opposer module and the balancer module—that can truly be considered topological basis elements, along with a general way to combine those elements, so as to span the complete solution space to the RPA problem.

From this general RPA framework, it has now been established conclusively that RPA-capable networks must be *modular*. The opposer module is a rich and potentially complex generalization of the negative feedback integral control that has been well-known to control engineers since the 1970s [5, 26, 27]. The balancer module, on the other hand, generalizes the simple incoherent feedforward motif that occurs repeatedly in bacterial transcription networks. It is now clear that the NFBBL motif identified by Ma et al. [21] is a special case of an opposer module, while the IFFLP motif [21] is the smallest possible balancer module that incorporates an independent “balancer node.” The antithetic integral control motif [22] is also a simple opposer module. RPA basis modules may be far more complex than these well-studied examples, as we shall see in this chapter, and may be interconnected in well-defined ways to construct arbitrarily large RPA-capable networks [5].

The goal of this chapter is to present the essence of general RPA theory through the discussion and analysis of a selection of examples. Each example is chosen to highlight one or more of the essential design principles that characterize general RPA networks. In each case, we illustrate the use of a diagrammatic method which captures the mathematical content of the “RPA equation”—the fundamental algebraic condition which must be satisfied by all RPA-capable networks—thus providing a tool for exploring the deep principles of RPA without detailed mathematical knowledge of the underlying theory. It is our hope that this diagrammatic method will offer an accessible overview of RPA theory to

mathematicians and nonmathematicians alike and will stimulate broad scientific interest in this vital topic in biocomplexity theory.

2 The Mathematics of RPA

In the interests of a self-contained presentation, we begin with a brief overview of mathematical RPA theory to provide a backdrop for the examples we explore in Subheading 4. In Subheading 2.1 we present a brief discussion of some of the earliest known models of RPA networks, which were almost entirely limited to just three or four interacting elements. In Subheading 2.2, we discuss the topological framework [5] that allows more general RPA-capable topologies to be identified, in networks that contain arbitrarily large numbers of interacting elements and interconnections. Readers with limited interest in the mathematical details of RPA in a general network setting may omit this section and can proceed to the more pragmatic discussion in Subheadings 3 and 4 without loss of continuity. Subheading 3 will be devoted to the development of a diagrammatic representation of the “RPA equation”—the defining algebraic constraint that must be satisfied by all RPA-capable networks. We will employ this diagrammatic method to represent the flow of biochemical information in our example networks later in Subheading 4, illustrating the essential RPA-promoting mechanisms at play in each case.

2.1 Early Models of RPA in Biology

Early approaches to studying RPA may be grouped into three overarching methodologies [28]:

1. *Development of models corresponding to well-known adaptive systems in biology.* One of the best known cases of this strategy is the Barkai-Leibler model of bacterial chemotaxis [4]. Barkai and Leibler’s work on chemotaxis in *E. coli* has exerted a strong and long-lasting influence on RPA theory for many reasons. It was the first study to show, for a specific molecular signalling network, that RPA could be a systems-level property, independent of parameter fine-tuning. In fact, the use of the word *robustness* to describe the independence of the adaptive performance from system parameters may be traced back to this seminal work. But the signalling events that regulate bacterial chemotaxis, the subject of Barkai and Leibler’s work, are somewhat unusual in that the functionality (in this case, RPA of tumbling frequency) can be captured by only a very small number of signalling molecules. This has reinforced the notion of “motif” in signalling networks—the concept that particular network functionalities such as RPA may be engendered at the level of a relatively small functional unit within a larger network.

2. *Ad-hoc modelling.* Modelling of small systems of interactions, generally from a nonlinear dynamics perspective [29], has also made significant contributions to early RPA models, often drawing on the possibility of strong parallels between biochemical network structures and engineering design principles. This perspective has produced, among other functional response units, the “sniffer” motif, which is simple two-component RPA model. This modelling framework has also suggested a two-component feedback model for RPA (homeostasis) that relies on an embedded ultrasensitive response element [29], thereby highlighting the deep relationship between ultrasensitivity and RPA. The antithetical integral control motif [22, 23] is a more recent example of a small motif structure that produces RPA. This motif has been genetically engineered as a synthetic controller in living cells and has been applied to growth-rate control in *E. coli* [23].
3. *Computational searches.* Comprehensive computational searching strategies consider RPA from the point of view of an inverse problem: Given that the output of the network must return to the same fixed baseline value, irrespective of the input magnitude, what is the space of all possible motifs that could achieve this? This approach is not about asking which solutions are actually observed in particular organisms or signalling contexts, or even about which solutions might be “best”; it is purely a matter of delineating the complete space of biochemically plausible ways to implement the required function. The computational demands of this approach *severely* restrict the network sizes that can be investigated by this method, and often impose a level of “coarse-graining” of the parameter space to be sampled. The most comprehensive computational search for solutions to the RPA problem at the present time remains the study by Ma et al. [21], which was limited to small networks of just three nodes. This influential study revealed that for three-node networks, just two types of motif are capable of exhibiting RPA.

Schematic representations of the main classes of small RPA motifs described above are depicted in Fig. 1.

2.2 The Mathematical Basis for RPA in General Interaction Networks

We now turn our attention to the more challenging problem of how robust perfect adaptation (RPA) may be achieved in complex networks that are self-organizing, self-regulating, and evolvable—in other words, networks that arise in biology. To understand how complexity is organized into robustly adapting systems in nature, it is not a simple matter of looking for *more* solutions to the RPA problem, given that we already know *some* solutions in very small, simple networks (as summarized in Subheading 2.1). Nor do we seek a method for testing particular network topologies to see if

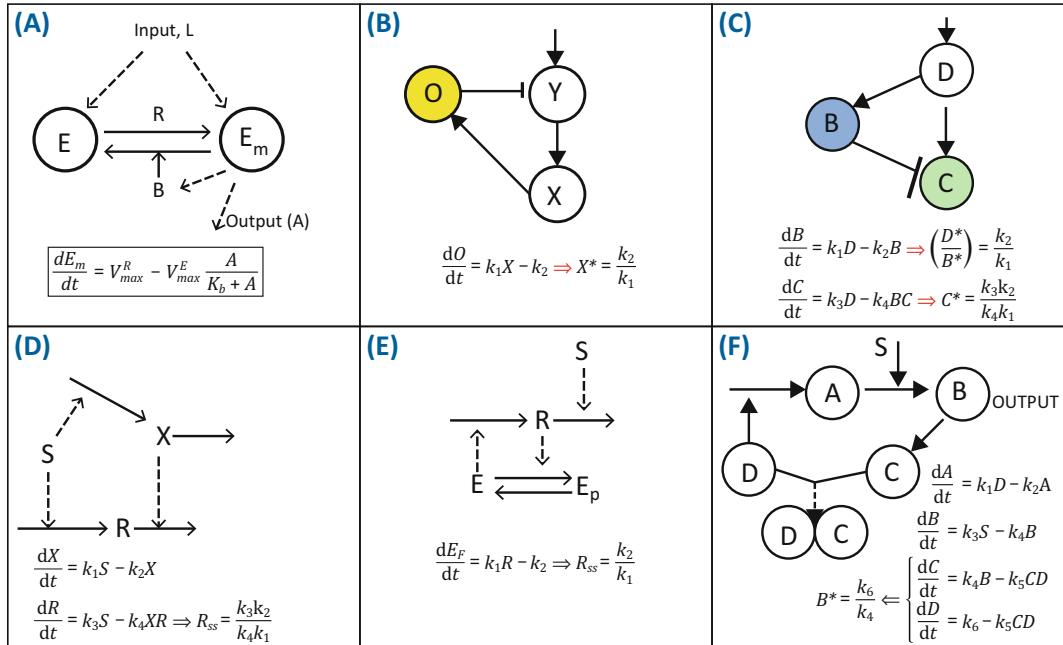


Fig. 1 Schematic representations of early RPA models. **(a)** Barkai-Leibler model of bacterial chemotaxis [4]. **(b)** Negative feedback loop with buffer node (NFLBN) identified by Ma et al. [21]. **(c)** Incoherent feedforward loop with proportioner node (IFLPN) identified by Ma et al. [21]. **(d)** ‘Sniffer’ model of RPA [29]. **(e)** Homeostasis model with embedded ultrasensitivity [29]. **(f)** Antithetic integral control model of RPA [22, 23]

they are capable of RPA. Rather, we seek *the set of all possible RPA network topologies*—that is, all the possible arrangements of nodes that are capable of exhibiting RPA, along with the constraints on reaction mechanisms at those nodes.

Our recent work [5] has demonstrated that two—and only two—classes of network topology (modules) can truly be considered *topological basis elements* for RPA in arbitrarily complex networks. These topological basis modules, along with a general way to combine those modules, *span* the complete solution space to the RPA problem. RPA-capable networks are thereby constrained to be modular. This fundamental property of robustly performing complex networks suggests that they may now be studied from the point of view of their unexpected simplicity—that is, as decompositions into basis modules. Since the set of all possible network arrangements grows factorially with network size, the very limited set of RPA-capable network topologies defined by this basis effectively represents the “needles in the haystack” that evolution must seek out over and over again as networks that require the RPA property grow and change over time.

The essential foundation for the development of this RPA theory may be summarized briefly as follows (*see* Ref. [5] for full details):

Suppose we have n interacting “nodes.” Typically, “nodes” refer to the interacting physical components in the system, although they could refer to more complicated mathematical expressions involving these interactions [5]. Here, for the sake of concreteness, it will be helpful to consider the “nodes” to be the interacting proteins in an intracellular signalling network. We denote the respective concentrations (or abundances) of these interacting proteins as P_1, P_2, \dots, P_n . One of these proteins, the “input node,” will receive a signal from outside the network, while one node (often, but not necessarily, distinct from the input node) is the endpoint of interest, which is expected to exhibit RPA in response to the signal received at the input node, and is given the status of the “output node.” Let us assume that the n nodes are ordered so that the input node is P_1 and the output node is P_n . For a network with single input/output node, we will place this node first in the ordering, P_1 .

Now, to each node P_i in the set, we associate a reaction rate $f_i = dP_i/dt$, which allows us to account for the nature and strength of the interactions among the n nodes of the network. The rate f_i will always be regulated by at least one *other* node, say P_j . It will almost always also depend on the node’s own concentration, P_i , except in the case of an exceptional reaction mechanism of central importance to the RPA problem—the “opposition mechanism”—which we discuss in Subheading 3.2. The input node, P_1 , will also be regulated by the external stimulus, S . In general, f_i could depend on any subset of the n network nodes. Thus, f_i has the general form

$$f_i = dP_i/dt = f_i(P_1, P_2, \dots, P_n)$$

for all nodes not receiving an external stimulus, and

$$f_1 = dP_1/dt = f_1(P_1, P_2, \dots, P_n, S)$$

for the input node, P_1 .

It is a straightforward algebraic problem to determine that there are two key mathematical quantities that codify the sensitivity of the output node’s steady-state to the level of the externally delivered stimulus. These are (1) the system Jacobian, J_n , and (2) the “input-output minor,” M_{1n} , associated to J_n . We depict the structures of these quantities in Fig. 2. As shown, the input-output minor M_{ij} is obtained from J_n by eliminating the row associated with the network’s input node, P_i , and the column associated with the output node, P_j .

The ratio of the determinants of these two key matrices gives rise to the special algebraic condition which must be satisfied by all RPA-capable networks *at steady-state*, for all possible parameter

The diagram illustrates the derivation of the RPA equation from the Jacobian matrix J_n . The Jacobian matrix J_n is shown as a grid of partial derivatives. A red bracket on the left indicates the "Row for INPUT node, P_i ", and a red bracket at the top indicates the "Column for OUTPUT node, P_j ". An orange arrow points from J_n to the minor M_{ij} . Below M_{ij} , two red arrows point downwards, labeled "Input Row Removed" and "Output Column Removed". The minor M_{ij} is a square matrix with dashed red borders.

$$J_n = \begin{pmatrix} \frac{\partial f_1}{\partial P_1} & \dots & \frac{\partial f_1}{\partial P_{j-1}} & \frac{\partial f_1}{\partial P_j} & \frac{\partial f_1}{\partial P_{j+1}} & \dots & \frac{\partial f_1}{\partial P_n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_{i-1}}{\partial P_1} & \dots & \frac{\partial f_{i-1}}{\partial P_{j-1}} & \frac{\partial f_{i-1}}{\partial P_j} & \frac{\partial f_{i-1}}{\partial P_{j+1}} & \dots & \frac{\partial f_{i-1}}{\partial P_n} \\ \frac{\partial f_i}{\partial P_1} & \dots & \frac{\partial f_i}{\partial P_{j-1}} & \frac{\partial f_i}{\partial P_j} & \frac{\partial f_i}{\partial P_{j+1}} & \dots & \frac{\partial f_i}{\partial P_n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial P_1} & \dots & \frac{\partial f_n}{\partial P_{j-1}} & \frac{\partial f_n}{\partial P_j} & \frac{\partial f_n}{\partial P_{j+1}} & \dots & \frac{\partial f_n}{\partial P_n} \end{pmatrix}$$

$$M_{ij} = \begin{pmatrix} \frac{\partial f_1}{\partial P_1} & \dots & \frac{\partial f_1}{\partial P_{j-1}} & \frac{\partial f_1}{\partial P_{j+1}} & \dots & \frac{\partial f_1}{\partial P_n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_{i-1}}{\partial P_1} & \dots & \frac{\partial f_{i-1}}{\partial P_{j-1}} & \frac{\partial f_{i-1}}{\partial P_{j+1}} & \dots & \frac{\partial f_{i-1}}{\partial P_n} \\ \frac{\partial f_{i+1}}{\partial P_1} & \dots & \frac{\partial f_{i+1}}{\partial P_{j-1}} & \frac{\partial f_{i+1}}{\partial P_j} & \frac{\partial f_{i+1}}{\partial P_{j+1}} & \dots & \frac{\partial f_{i+1}}{\partial P_n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial P_1} & \dots & \frac{\partial f_n}{\partial P_{j-1}} & \frac{\partial f_n}{\partial P_{j+1}} & \dots & \frac{\partial f_n}{\partial P_n} \end{pmatrix}$$

Fig. 2 The RPA equation—the fundamental algebraic condition that must be satisfied by all RPA-capable networks—is derived from the “input-output” minor, M_{ij} , associated to the input node (P_i) and output node (P_j) for a network. As indicated, M_{ij} is obtained from the Jacobian matrix, J_n , of the system by eliminating the row of J_n associated with the input node, P_i , and the column associated with the output node, P_j .

sets, and all possible stimulus levels delivered to the system. In particular,

$$\text{Det}(M_{1n})/\text{Det}(J_n) = 0,$$

or, equivalently,

$$\text{Det}(M_{1n}) = 0, \text{ with } \text{Det}(J_n) \neq 0,$$

is a necessary condition for RPA in all networks, regardless of size or complexity, and must be satisfied by all system steady states, for all possible parameter choices, and all stimulus levels.

We refer to the equation $\text{Det}(M_{1n}) = 0$ hereafter as *the RPA equation*, with the condition.

$\text{Det}(J_n) \neq 0$ denotes *the RPA constraint*. Rigorous analysis of these algebraic conditions [5] has shown that the RPA equation, in particular, induces tremendous structure on network designs that accommodate the RPA property.

Note that both these mathematical conditions can contain an extraordinarily large number of terms in networks containing more than just a handful of nodes. Indeed, the RPA equation can contain up to $(n - 1)!$ terms, depending on how extensively the n nodes are interconnected. The RPA constraint can contain up to $n!$ terms. Fortunately, a topological approach to the structure of the RPA equation [5] has simplified this problem tremendously and allowed the set of all possible RPA-capable network configurations to be determined—not by enumeration, of course, but through the identification of a suitable basis.

With a view to describing the essential nature of this (topological) basis, along with the rules for combining (interconnecting) basis elements in general RPA-capable networks, we devote the next section to a new diagrammatic method for presenting the mathematical content of the RPA equation.

3 Methods for RPA Analysis

3.1 A Diagrammatic Method for Analyzing the RPA Capacity of a Network

As is evident from the discussion in the previous section, we require remarkably little biochemical information about a particular network to analyze its capacity for RPA. The RPA equation comprises a summation of terms, each a product of $(n - 1)$ factors of the form $\partial f_i / \partial P_j$ or the form $\partial f_k / \partial P_k$. Each factor of the form $\partial f_i / \partial P_j$ corresponds to a network interconnection $P_j \rightarrow P_i$, while $\partial f_k / \partial P_k$ may be referred to as the “kinetic multiplier” [5] associated to the node P_k . As we will see, kinetic multipliers play a pivotal role in “solving” the RPA equation since these are the only class of factors that can be identically zero while still preserving the configuration of the underlying network. In particular, a zero value for a factor of this type places constraints on the reaction mechanism occurring at the node in question, without removing any network linkages impinging on the node.

The mathematical content of each term in the RPA equation thus suggests a diagrammatic representation: Replacing the factor $\partial f_i / \partial P_j$ with the symbol $P_j \rightarrow P_i$ and representing $\partial f_k / \partial P_k$ with the symbol (P_k) , we see that the terms of the RPA equation correspond to successions of linkages, along with kinetic multipliers, that exist in the underlying network. Together, these form “routes” leading from the input node to the output node, feedback loops (multi-node cycles), and products of kinetic multipliers (one-node cycles).

In fact, we have shown [5] that the special algebraic structure of the determinant map will constrain each term of the RPA equation to assume the following form:

$$\underbrace{(+/-)}_{\text{native sign}} \underbrace{(+/-)}_{\text{influence sign}} \left(\underbrace{P_1 \rightarrow \cdots \rightarrow P_n}_{\text{route component}} \right) \\ \times \left(\underbrace{(P_m \rightarrow \cdots \rightarrow P_k)(P_x)(P_y) \cdots (P_z)}_{\text{cycle component}} \right).$$

In the special case of a network with a single input/output node, the route component will reduce to unity, producing an RPA equation with only cycle components:

$$\underbrace{(+/-)}_{\text{native sign}} \underbrace{(+/-)}_{\text{influence sign}} \left(\underbrace{(P_m \rightarrow \dots \rightarrow P_k)(P_x)(P_y) \dots (P_z)}_{\text{cycle component}} \right).$$

In fact, each and every “route” in a particular network (leading from the input node through to the output node) will be represented as a route component $P_1 \rightarrow \dots \rightarrow P_n$ in the associated RPA equation. Likewise, each and every feedback loop that exists in the network will appear in the cycle component of the RPA equation. In particular, the full set of cycle components appended to a particular route component will be generated by *all possible combinations of cycles* (i.e., feedback loops and kinetic multipliers) formed by the nodes that are *disjoint from* (not occurring in) that route component. For clarity and convenience, we represent a feedback loop of the form $P_a \rightarrow P_b \rightarrow P_c \rightarrow P_a$ with the diagrammatic representation ($P_a \rightarrow P_b \rightarrow P_c$)—see Fig. 3. The RPA equation may thus be interpreted as a summation of all possible route-cycle combinations for its network.

Note that each term in the RPA equation is prefixed with a positive or negative sign. There are two contributing factors to this sign, so we represent it as the product of two component signs—the *native sign* and the *influence sign*, as indicated above. The native sign is inherited from the term’s contribution to the determinant

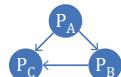
FACTOR appearing in a TERM of the RPA equation	Equivalent diagrammatic representation of FACTOR	Network schematic representation of FACTOR	FACTOR’s contribution to Network Topology
$\frac{\partial f_B}{\partial P_A} \frac{\partial f_C}{\partial P_B} \frac{\partial f_D}{\partial P_C}$	$P_A \rightarrow P_B \rightarrow P_C \rightarrow P_D$	$P_A \rightarrow P_B \rightarrow P_C \rightarrow P_D$	ROUTE SEGMENT
$\frac{\partial f_B}{\partial P_A} \frac{\partial f_C}{\partial P_B} \frac{\partial f_A}{\partial P_C}$	$(P_A \rightarrow P_B \rightarrow P_C)$		FEEDBACK LOOP
$\frac{\partial f_A}{\partial P_A} \frac{\partial f_B}{\partial P_B} \frac{\partial f_C}{\partial P_C}$	$(P_A)(P_B)(P_C)$	no contribution to network interconnections	BIOCHEMICAL REACTION MECHANISMS

Fig. 3 Collections of factors appearing in a single term of the RPA equation may be represented diagrammatically as route segments, feedback loops or one-node cycles (“kinetic multipliers”). Of these three types of network contributions resident in the terms of the RPA equation, only kinetic multipliers can assume a zero value without affecting the interconnectivity of nodes in the network, since these reflect the nature of the reaction mechanisms at individual nodes

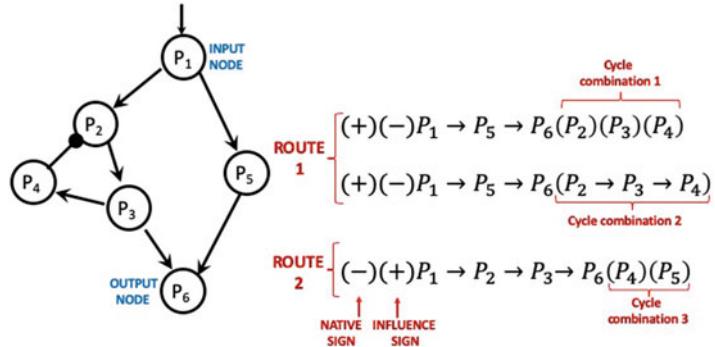


Fig. 4 A simple six-node network example, illustrating the diagrammatic method for deducing the RPA equation for a network. As shown, this network comprises two distinct routes from input (P_1) to output (P_6). The cycle combination appended to each route comprises all possible combinations of cycles composed of nodes that are disjoint from that route

expansion. It is straightforward to show (see Proposition 1 in [5]) that the native sign for the term is given by $(-1)^{z+n+1}$, where z is the number of distinct cycles (feedback loops plus kinetic multipliers) present in the term and n is the number of nodes in the network. The influence sign, on the other hand, takes account of how many inhibitory interactions occur in the sequences of node-node linkages comprising routes and feedback loops represented in the term. An odd number of inhibitory interactions attracts a negative sign, while a positive number of such interactions attracts a positive sign. Note that kinetic multipliers always attract a negative sign [5].

In Fig. 4 we demonstrate the application of these simple principles by deducing the diagrammatic representation of the RPA equation for a simple six-node network. For reasons that will become clear through the analysis of the examples in Subheading 4, the particular network configuration depicted in Fig. 4 is *not* capable of exhibiting RPA. Note that here, and in all subsequent network schematics, we adopt the convention that the symbol \rightarrow represents an *activating* interaction, while the symbol \dashrightarrow denotes an *inhibitory* interaction. For this network, P_1 is selected as the input node, while P_6 is the output node.

In this particular network, there are two distinct routes from P_1 to P_6 : $P_1 \rightarrow P_5 \rightarrow P_6$ and $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_6$. For the route $P_1 \rightarrow P_5 \rightarrow P_6$, containing no inhibitory interactions, there is just one feedback loop disjoint from the route, namely, $(P_2 \rightarrow P_3 \rightarrow P_4)$. There are thus two possible cycle combinations appended to this route: $(P_2 \rightarrow P_3 \rightarrow P_4)$ and $(P_2)(P_3)(P_4)$. Both combinations contribute a negative interaction sign to the term (a *negative* feedback loop and an *odd* number of kinetic multipliers). The native sign contributed by the first combination is $(-1)^{1+6+1} = (+1)$; for the second combination, the native sign is $(-1)^{3+6+1} = (+1)$.

For the route $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_6$, also containing no inhibitory interactions, there are no disjoint feedback loops; the route is therefore only appended to the cycle combination $(P_4)(P_5)$. This combination contributes a positive interaction sign (an *even* number of kinetic multipliers), with native sign equal to $(-1)^{2+6+1} = (-1)$.

3.2 Fundamental Design Principles of RPA Networks

Equipped with an understanding of the mathematical content of the RPA equation, we may now consider the general properties of network configurations that satisfy the RPA equation. At the broadest level, identifying the full space of all RPA-capable network topologies relies on a general method for partitioning the RPA equation to “independently adapting subsets.” This framework interprets the terms of the RPA equation as elements of a topological space, with the independently adapting sets serving as the closed sets of that space [5]. Such partitions of the RPA equation impose severe constraints on the design principles of networks capable of exhibiting RPA.

With the aim of presenting a nontechnical exposition of the essential design principles of RPA-capable networks uncovered by this approach, we proceed in two steps: First, we present (below) a summary of the network design principles that may be deduced from the properties of a mathematically valid partition of the RPA equation. Then, in the remainder of this chapter (Subheading 4), we illustrate the application of these core RPA design principles through the discussion and analysis of five simple network examples, exploiting the use of the diagrammatic method we presented in Subheading 3.1. The design summary below and the set of illustrative examples together give a comprehensive overview of the essential design features that characterize all RPA-capable networks.

Summary of Design Principles of RPA-Capable Networks

A. General Principles

- (a) There exist two, and only two, distinct types of independently adapting subsets in a mathematically valid partition of the RPA equation: S-sets and M-sets. All terms contained in an S-set are identically zero due to the presence of a “special” kinetic multiplier ($P_k \equiv \partial f_k / \partial P_k = 0$ (at steady-state) for all stimulus levels, and for all network parameters. All terms contained in an M-set are strictly non-zero, and exactly “balance”—for all stimulus levels, and for all network parameters.
- (b) In a valid partition, terms are assigned to independently adapting subsets *by route*. That is, once a term has been assigned to a particular set in the partition, all terms with the same route component must also be assigned to that set.

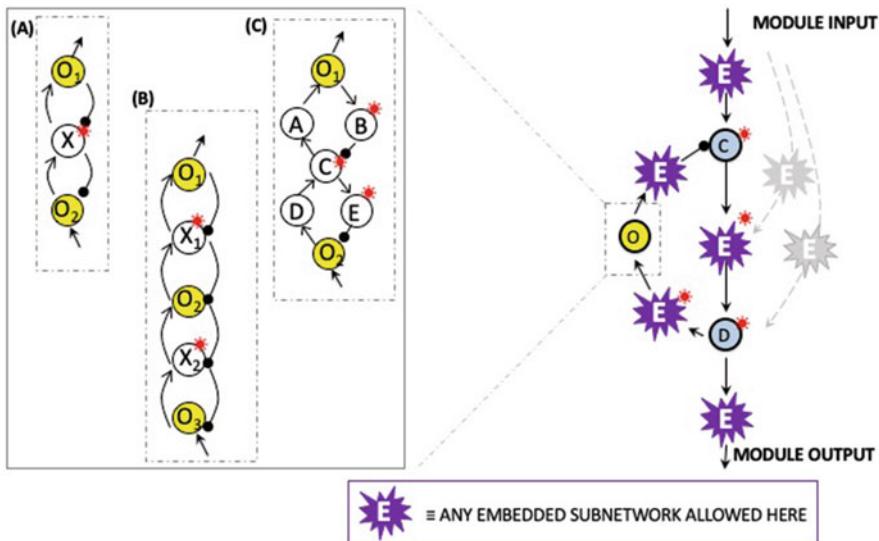


Fig. 5 Essential design features of an opposer module. Opposer modules are characterized by a feedback architecture delineated by two nodes—a connector node, C, at the apex of the module, and a diverter node, D, at the base of the module. Arbitrary subnetworks may be embedded at the positions indicated without affecting the RPA-promoting function of the module. At least one computational node (an opposer node) is embedded into the feedback segment of the module. All nodes that exhibit the RPA property due to the activities of the opposer node(s) are indicated with a red asterisk. Collections of opposer nodes may be organized in intricate topological arrangements within the feedback segments of opposer modules, and work together collaboratively to confer RPA on the diverter node of the module. Several possible opposer node arrangements are illustrated in the inset to the figure (*see Ref. [5]* for a more complete description of “opposing sets” and their associated “master sets”): (a) a two-node opposing set with three-node master set; (b) a three-node opposing set with five-node master set; and (c) a two-node opposing set with seven-node master set

- (c) The terms of an S-set correspond to the presence of an opposer module in the associated network. The special reaction mechanism ($P_k \equiv 0$) referred to as an opposer mechanism or opposer kinetics. The node P_k is referred to as an opposer node. Opposer nodes can *only* exist within a feedback loop (*see Theorem 1 in [5]*). The essential structure of an opposer module is illustrated in Fig. 5. The feedback architecture is defined by a diverter node (D) at the base of the module and a connector node (C) at the apex of the module.
- (d) The terms of an M-set correspond to the presence of a balancer module in the associated network. As illustrated in Fig. 6, a balancer module is defined by a diverter node (D) at the apex, and a connector node (C) at the base, with one or more route segments connecting these two nodes. Subnetworks of arbitrary complexity may be embedded into these route segments. All nodes downstream of the diverter node and upstream of the connector node are referred to as balancer nodes.

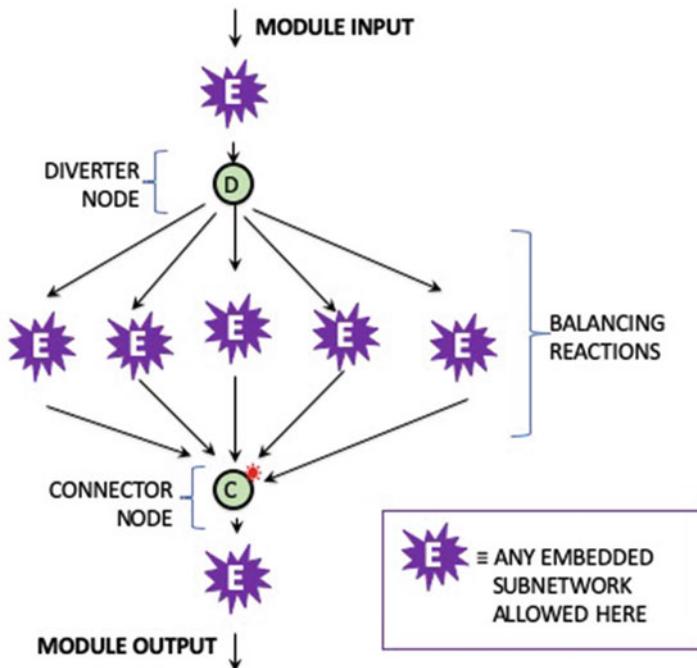


Fig. 6 Essential design features of a balancer module. Balancer modules possess a “feedforward” topology, which may incorporate any number of parallel pathways between the two defining nodes of the module—the diverter node, D, at the apex, and the connector node, C, at the base. Feedback loops may also be embedded into these parallel pathways, as indicated. Only the connector node exhibits the RPA property in a balancer module, as indicated by the red asterisk

- (e) A network for which the input and output nodes are *not distinct* (i.e., a network with a single input/output node) has an RPA equation which may *only* contain S-set(s). Equivalently, a network with a single input/output node can *only* achieve RPA through a decomposition into one or more opposer modules. It cannot contain balancer modules (*see Example Network 3 in Subheading 4*).

B. Opposer Principles

- (a) An opposer module can function with a single opposer node, but can also employ a collection of opposer nodes which work collaboratively to “solve” the RPA problem for the network. Such a collection of opposer nodes is known as an “opposing set.” Opposing sets must work within a very well-defined subnetwork topology, embedded into a feedback loop (*see Theorem 3 in [5]*). Some simple configurations for valid opposing sets are depicted in Fig. 5. Interested readers are referred to [5] for a more extensive description of opposing set topologies.

- (b) An opposer node requires a *unique* regulator node. If, in practice, multiple nodes regulate an opposer node, then this unique regulator is the nearest upstream branchpoint of these multiple regulatory pathways (*see* Ref. [5]). For clarity, Subheading 4 only considers examples where the opposer node is regulated directly by a single upstream node.
- (c) An opposer node exhibits opposer kinetics (*see* point *c* under “General Principles” in part A above) via a mathematical principle known as *kinetic pairing* (*see* Ref. [5] for further details). If P_o is the concentration of the opposer node, and P_R is the concentration of its unique regulator, then kinetic pairing results in the vanishing of a function of the following form on the steady-state locus of the system: $\mathcal{J}(P_o).(P_R - k)$, where k is a function of network parameters. The simplest possible function form for opposer kinetics is a rate equation that is zero-order in P_o , which is the functional form we adopt in all our examples in Subheading 4. Much more complicated mechanisms for orchestrating opposer kinetics are possible however (*see* Ref. [5] for a more extensive discussion of this point).
- (d) An opposer node never exhibits the RPA property. Rather, an opposer node uses kinetic pairing (*see* point *c* above) to compute an “error integral” [5] which confers the RPA property on its unique regulator node. This, in turn, confers the RPA property on various other nodes within the opposer module (*see* Fig. 5).
- (e) An opposer node can never directly regulate another opposer node. This fact places severe constraints on the topologies of opposing sets (*see* point *a* above).

C. Balancer Principles

- (a) In order to generate a balancer module, each M-set requires at least one term with negative sign, and at least one term with positive sign.
- (b) Balancer nodes and connector nodes are subject to special reaction mechanisms—the balancer mechanism and the connector mechanism, respectively. If P_B is the concentration of a balancer node, and P_{RB} is the concentration of its unique regulator, then kinetic pairing results in the vanishing of a function of the following form on the steady-state locus of the system: $(P_B/P_{RB} - k_B)$, where k is a function of network parameters. If P_C is the concentration of a Connector node and P_D is the concentration of the Diverter node of the module, then kinetic pairing results in the vanishing of a function of the following form

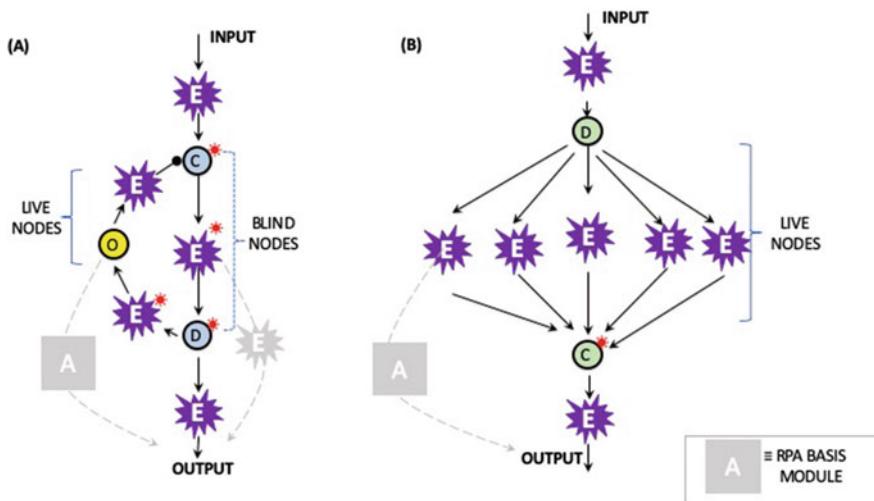


Fig. 7 In the context of intermodular connectivities, nodes that exhibit the RPA property within an RPA module are referred to as “blind nodes,” while nodes that do not exhibit the RPA property are known as “live nodes.” Any live nodes that contribute to a route of the network must be connected downstream to another RPA module (as indicated schematically by the symbol A above). Blind nodes can contribute to additional routes without affecting the RPA capacity of the network

on the steady-state locus of the system: $\mathcal{J}(P_D).(P_C - k_C)$, where k_C is a function of network parameters.

- (c) Balancer nodes exhibit balancer kinetics, and connector nodes exhibit connector kinetics, via the mathematical principle known as kinetic pairing (*see* point *c* under “Opposer Principles” in part B above).
- (d) The presence of the requisite term signs (*see* point *a* above) and the requisite reaction mechanisms at balancer/connector nodes enables the M-set to achieve independent adaptation through a “balancing act” [5].

D. Principles of Intermodular Connections

- (a) As noted schematically in Figs. 5, 6, and 7, some nodes with an RPA module exhibit RPA, and some do not. Connector nodes and opposer regulators are examples of nodes that do exhibit RPA, while balancer nodes and opposer nodes are examples of nodes that do not. Outgoing regulations from nodes that exhibit RPA are referred to as “blind” regulations. Any new network linkages that emanate from these nodes, leading ultimately to the output node, merely supply additional terms to the existing subset of the RPA equation partition associated with the module in question. The new network connections are thus to be interpreted as part of the exist module. By contrast, outgoing regulations from nodes that do not exhibit RPA are referred to as “live” regulations. Any

new network linkages that emanate from live nodes supply new terms that must be assigned to a different subset in the partition of the RPA equation. These new “routes” must therefore be connected downstream to a separate RPA module. The two modules act independently and are said to be connected “in parallel” (*see* Network Example 1 in Subheading 4).

- (b) It is possible for a single subset in a partition of the RPA equation to be associated with multiple potential modular structures in the underlying network. In this case, the multiple modules are said to be connected “in series” (*see* Network Example 2 in Subheading 4).

A final comment on stability is in order before proceeding to our network examples. Identifying RPA-exhibiting steady states from the solution of the RPA equation is not, in itself, a guarantee that the associated “set point” is a stable one. System stability is, in general, a complex matter that is beyond the scope of the present chapter, and for which we provide some guidance in [5]. We do note here that in the context of feedback regulation, positive feedback is highly destabilizing, while negative feedback, although no guarantee of stability, is stability-promoting. For this reason, all feedback loops chosen in the following examples are negative feedback loops, for which we show that stable RPA performance readily obtains.

4 Applications of the Diagrammatic Method to Example Networks

In this subheading, we illustrate these topological principles through an examination of five simple networks, each having an RPA equation containing only a small number of terms. In each case, we examine the conditions under which the network could exhibit RPA, identifying the computational nodes that are subject to special constraints on reaction mechanism in order to solve the RPA equation.

4.1 Example Network 1: An Opposer Module in Parallel with a Balancer Module

In Fig. 8 we depict a simple five-node network that receives an external stimulus at P_1 . Nominating the node P_5 as the output, this network contains a number of parallel pathways, as well as a negative feedback loop between nodes P_1 and P_2 . Appealing to the diagrammatic method introduced earlier, we readily conclude that the RPA equation for this particular network configuration contains three terms, as shown in Fig. 8.

A simple partition of the RPA equation for this network may be achieved as follows: A two-node opposer module is formed through the feedback interactions between P_1 and P_2 . The node

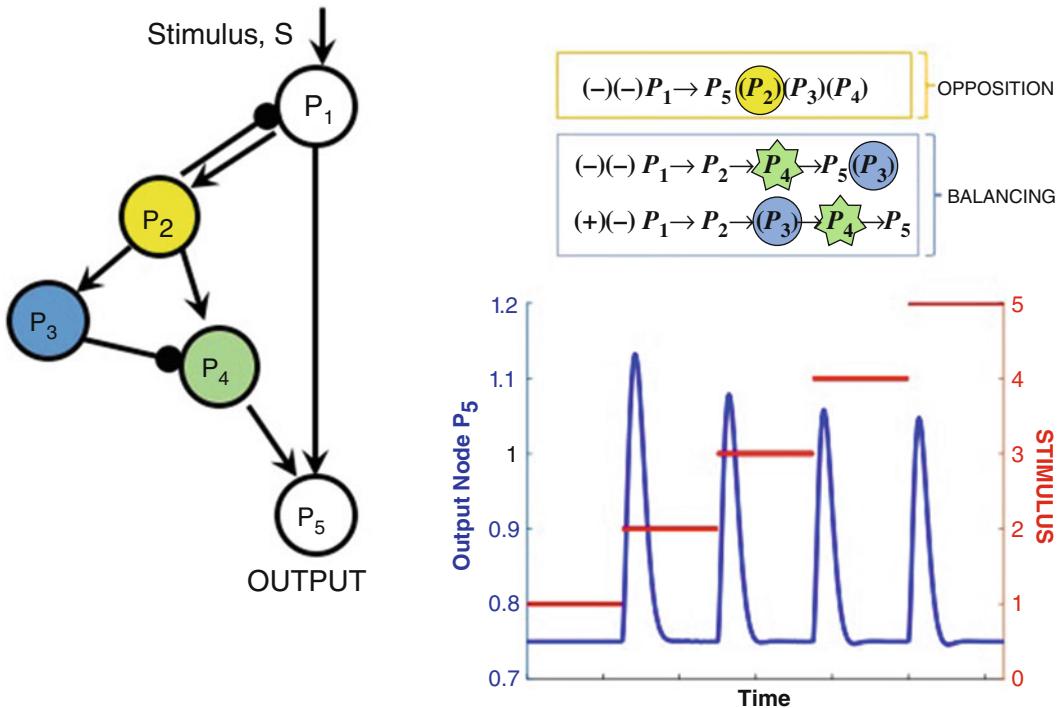


Fig. 8 Network Example 1: a two-node opposer module *in parallel* with a three-node balancer module. Parameters: $k_1 = k_3 = k_4 = k_{10} = 1$; $k_2 = k_5 = k_6 = k_7 = k_8 = k_9 = k_{11} = k_{12} = 0.5$

\$P_2\$ plays the role of the opposer node (noted in yellow) and is subject to opposer kinetics $\frac{\partial f_2}{\partial P_2} = 0$. The single regulator of this opposer node, \$P_1\$, will thus exhibit RPA under these conditions, and the first term of the RPA equation noted in Fig. 8 will be assigned to an S-set (noted in yellow).

But the opposer node, \$P_2\$, also appears in a *route* of the network: The network connections \$P_2 \rightarrow P_3\$ and \$P_2 \rightarrow P_4\$ are therefore “live” regulations, and the terms containing these regulations in their route components must be assigned to a separate subset in the partition. We see that two different pathways emanating from \$P_2\$ reconnect at node \$P_5\$, and noting that the two remaining terms of the RPA equation have different signs, it follows that assigning \$P_3\$ the role of a balancer node (noted in blue), and \$P_4\$ the role of a connector node (noted in green), will allow \$P_4\$ to exhibit RPA. These terms are thereby assigned to an M-set (noted in blue in the diagrammatic representation). Under these conditions, the output node, \$P_5\$, is regulated by two RPA-exhibiting nodes (\$P_1\$ and \$P_4\$) and must, itself, exhibit RPA.

Thus, for this RPA solution, there are three nodes subject to special reaction mechanisms: \$P_2\$ requires opposer kinetics, \$P_3\$ requires balancer kinetics, and \$P_4\$ requires connector kinetics. The remaining nodes (\$P_1\$ and \$P_5\$) are not subject to any constraints; the

RPA capacity of this network is unaffected by the reaction kinetics at these nodes. For illustrative purposes, we choose the simplest possible functional forms for the rate equations at P_2 , P_3 , and P_4 as follows:

For opposer node, P_2 :

$$\frac{dP_2}{dt} = k_4 P_1 - k_5 \quad (1)$$

Since this reaction rate is zero-order in P_2 , it is clear that the opposer condition $\frac{\partial f_2}{\partial P_2} = 0$ will obtain at steady state, for any choice of parameters. In fact, it readily follows from this equation that the unique regulator node, P_1 , exhibits the RPA property with set point k_5/k_4 .

For balancer node, P_3 :

$$\frac{dP_3}{dt} = k_6 P_2 - k_7 P_3 \quad (2)$$

This rate equation ensures that P_3 will always be proportional to P_2 at steady state:

$$P_3 = (k_6/k_7)P_2.$$

For Connector node, P_4 :

$$\frac{dP_4}{dt} = k_8 P_2 - k_9 P_3 P_4 \quad (3)$$

Connector kinetics are ensured by this equation, since both terms are “balanced” by the diverter node, since at steady state, $P_3 = (k_6/k_7)P_2$ due to the upstream balancer kinetics. It follows that P_4 exhibits the RPA property with set point $k_8 k_7 / k_9 k_6$.

As noted above, any functional forms are allowed for the remaining nodes, provided the reaction rates accurately reflect the regulations noted in the network schematic. We choose the following simple forms:

$$\frac{dP_1}{dt} = k_1 S - k_2 P_2 - k_3 P_1 \quad (4)$$

$$\frac{dP_5}{dt} = k_{11} P_1 + k_{12} P_4 - k_{10} P_5 \quad (5)$$

This network thus achieves RPA through the actions of two independently functioning modules—an opposer module and a balancer module. Since the modules contribute independently, each associated to its own subset in the partition of the RPA equation, the modules are said to be *in parallel* (see Ref. [5] for further discussion).

We solve Eqs. (1) through (5) for a range of different stimulus strengths, with results as shown in Fig. 8 (see figure caption for parameter values). Notice that each time the stimulus (red curve) undergoes a step increase, the output node (blue curve) exhibits a

rapid initial increase, but then diminishes back to the baseline level, even though the larger stimulus magnitude persists. Since the output always returns to the same value, regardless of the stimulus level being received at the input node, the system exhibits RPA.

4.2 Example

Network 2: An Opposer Module in Series with a Balancer Module

Figure 9 presents an alternative combination of the two module types considered in the previous example. In this configuration, the two-module structure is, in fact, *redundant*. In particular, this network will exhibit RPA as a single opposer module if the node P_2 acts as an opposer node (i.e., if it operates with opposer kinetics $\frac{\partial f_2}{\partial P_2} = 0$). If P_2 does not act as an opposer, the network can still exhibit RPA as a single balancer module, provided P_3 exhibits balancer kinetics and P_4 exhibits connector kinetics. In other words, this particular configuration can operate *either* as an opposer module *or* as a balancer module. For this reason, we say that the two potential modules—the opposer and the balancer—are topologically *in series*.

This series connection of the two modules becomes clear when we examine the two-term RPA equation for this network (see diagrammatic representation in Fig. 9).

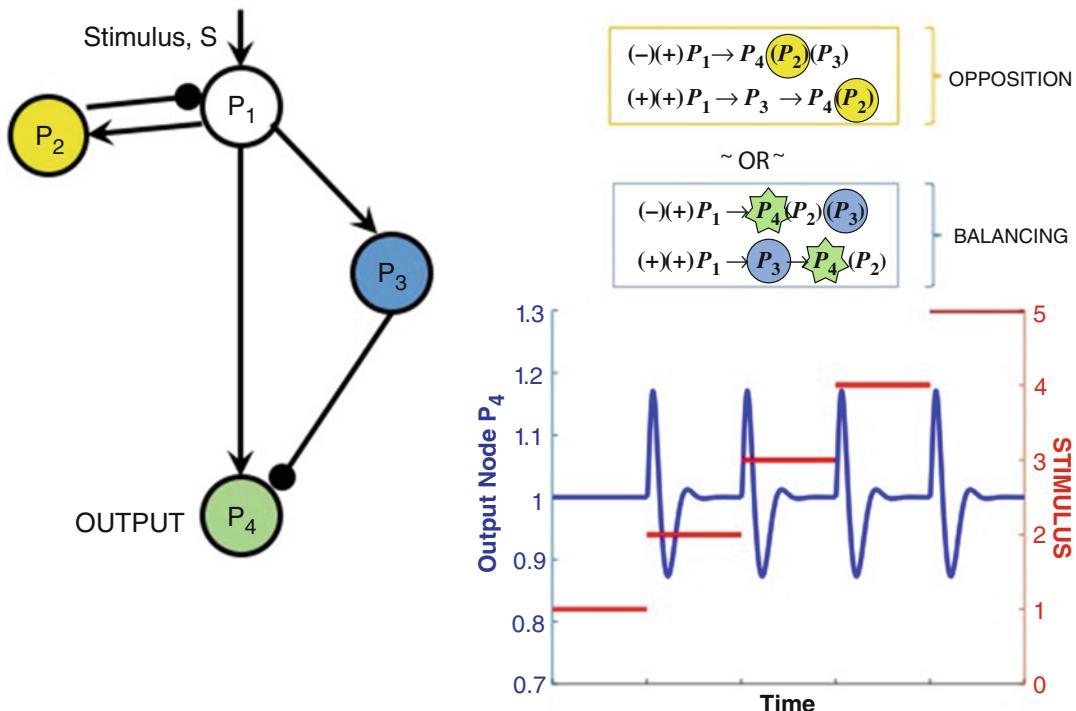


Fig. 9 Network Example 2: a two-node opposer module *in series* with a three-node balancer module. Parameters: $k_1 = k_2 = k_3 = k_4 = k_5 = k_6 = k_7 = k_8 = k_9 = 1$

We simulate a set of model equations for this network, corresponding to the configuration as a single opposer module (with diagrammatic solution highlighted in yellow as an opposition mechanism). The only node subject to a special reaction constraint in this scenario is P_2 , for which we again choose the simple zero-order form to confer opposer kinetics:

$$\frac{dP_2}{dt} = k_4 P_1 - k_5 \quad (6)$$

For the remaining three nodes, any reaction mechanism is allowed. We choose the following very simple rate equations:

$$\frac{dP_1}{dt} = k_1 S - k_2 P_2 - k_3 P_1 \quad (7)$$

$$\frac{dP_3}{dt} = k_6 P_1 - k_7 P_3 \quad (8)$$

$$\frac{dP_4}{dt} = k_8 P_1 - k_9 P_3 P_4 \quad (9)$$

We note that the rate equations for P_3 and P_4 happen to be consistent with balancer kinetics and connector kinetics, respectively. But any other functional forms for these equations would also be suitable, since the opposer module is active as long as P_2 operates with opposer kinetics, as reflected in Eq. (6).

Simulation outputs for Eqs. (6) through (9) are given in Fig. 9. Note that for this particular choice of parameters (see figure caption), the system exhibits damped oscillations. Indeed, as the stimulus undergoes a step increase, the output first increases rapidly, but then decreases and undershoots the set point. The signal then increases again and slightly overshoots the set point, before finally approaching the set point asymptotically. Since the set point is independent of the stimulus magnitude, the system exhibits RPA. The time-dependent response of the system, on the other hand, and its propensity for oscillations, depends on the choice of parameters. Notice also that the time-dependent response exhibits a great deal of symmetry: as long as the difference in stimulus magnitude is preserved from one step increase to the next, the time-dependent response of the system remains identical. This latter property is a consequence of the special (simple) choices of reaction kinetics used in this example.

4.3 Example

Network 3: A Single Opposer Module with Two-Node Opposing Set and a Single Input-Output Node

In the simple four-node network example in Fig. 10, we illustrate two interesting features that can characterize RPA-capable networks: (1) The network need not have distinct input-output nodes—a valid topological possibility rarely considered in the RPA literature to date—and (2) a single opposer module may contain a whole constellation of opposer nodes, connected together in an orchestrated collection of interconnected feedback

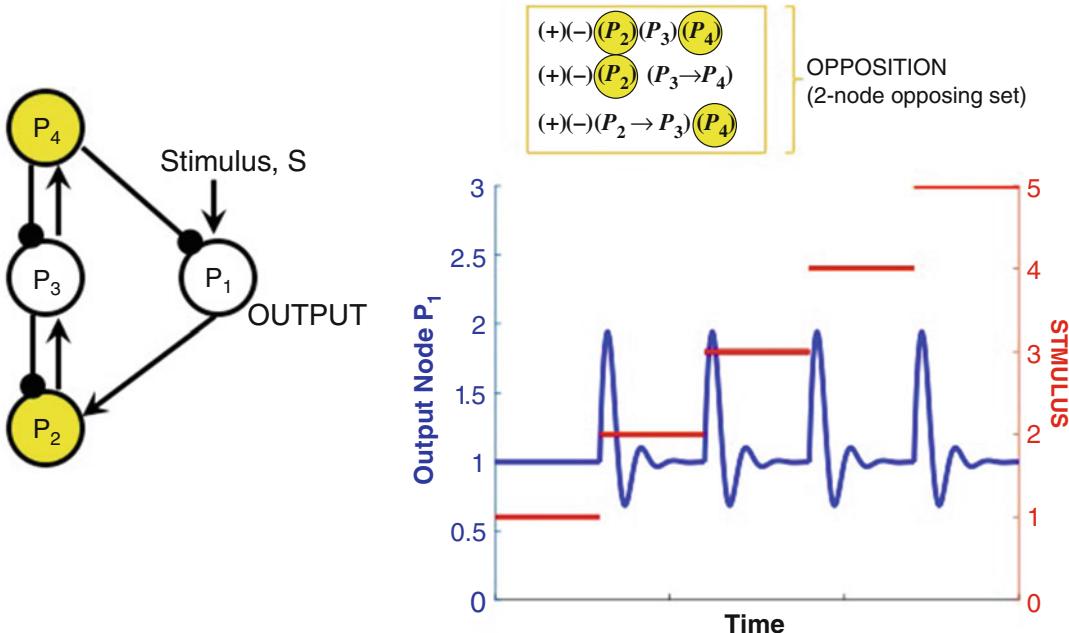


Fig. 10 Network Example 3: a two-node opposing set (a single opposer module with two collaborating opposer nodes). The network has a single input/output node (P_1). Parameters: $k_1 = k_2 = k_6 = k_7 = k_8 = 1$; $k_3 = k_4 = k_5 = k_9 = k_{10} = 0.8$

loops. For simplicity of illustration, we choose to examine a set of just two opposer nodes collaborating to confer RPA on the output node, P_1 (which is also the node receiving the external stimulus, S).

Networks with a single input/output node have an RPA equation with a slightly different structure insofar as each term in the equation contains only the cycle component. There are no “routes” in networks with a single input/output node. Thus, as noted in the summary of RPA network design features in Subheading 3.2, a network of this type can only achieve RPA through an opposer module.

There are two (negative) feedback loops disjoint from the input/output node in this network, giving rise to three different cycle combinations as indicated in the diagrammatic representation in Fig. 10. All three of the remaining nodes exist in feedback loops, and it is clear from the diagrammatic representation that the opposer mechanisms $\frac{\partial f_2}{\partial P_2} = \frac{\partial f_4}{\partial P_4} = 0$ are sufficient to satisfy the RPA equation for this network.

We solve the following simple rate equations for this RPA network solutions for a range of increasing steps in the stimulus, S :

$$\frac{dP_1}{dt} = k_1 S - k_2 P_4 - k_3 P_1 \quad (10)$$

$$\frac{dP_2}{dt} = k_4 P_1 - k_5 P_3 \quad (11)$$

$$\frac{dP_3}{dt} = k_6 P_2 - k_7 P_4 - k_8 P_3 \quad (12)$$

$$\frac{dP_4}{dt} = k_9 P_3 - k_{10} \quad (13)$$

These solutions are depicted in Fig. 10. Notice that for these simple reaction kinetics, and for the chosen parameters (see figure caption), the time-dependent response exhibits damped oscillations as well as a high level of symmetry for fixed step increases in stimuli.

4.4 Example

Network 4: A Single RPA-Capable Network with Two Possible RPA Solutions

In the example depicted in Fig. 11, we highlight two crucial properties that characterize many RPA-capable networks:

1. RPA may be achieved in large networks with constraints at only a single node of the network. In other words, all but one of the vast array of chemical reactions comprising a complex network may be perturbed (mutationally, pharmacologically, or otherwise) while exerting no influence whatsoever on that network's ability to exhibit RPA. Only the single computational node of the network (e.g., the opposer node P_4 in the case of the network in Fig. 11) is subject to a strict constraint on its reaction mechanism (here, an opposer mechanism).

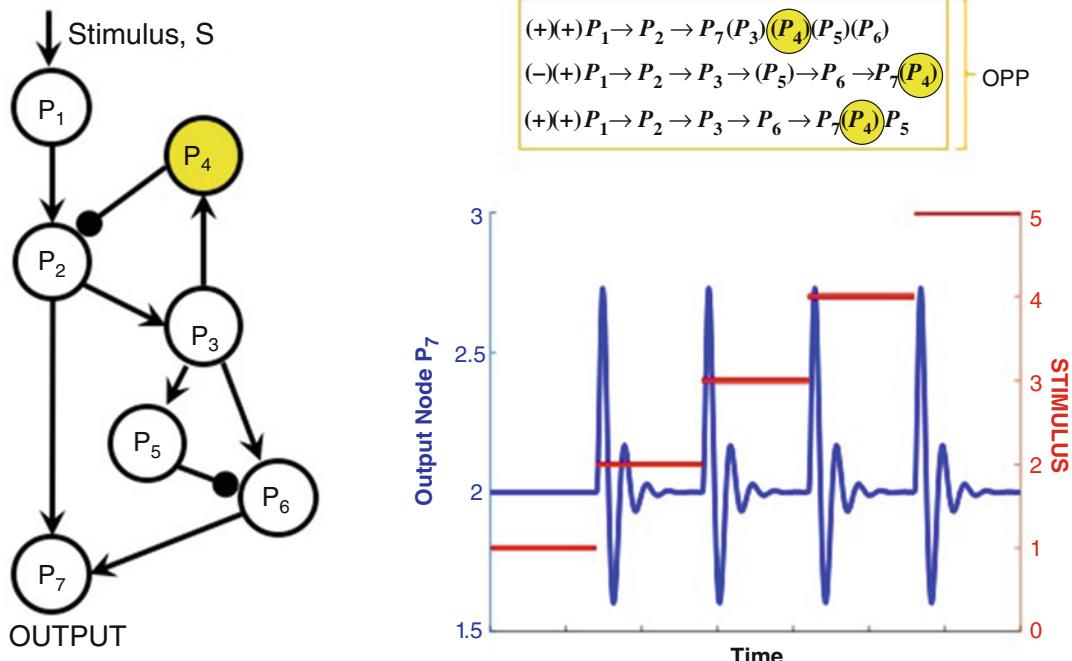


Fig. 11 Network Example 4: an RPA-capable network with two possible decompositions into basis modules. Here we depict the network in its realization as a single opposer module. Parameters: $k_1 = k_2 = k_3 = k_4 = k_5 = k_{10} = k_{11} = k_{12} = k_{13} = k_{14} = k_{15} = k_{16} = 1$. $k_6 = k_7 = k_8 = k_9 = 0.5$

2. For networks of sufficient complexity, there may be multiple ways for the network to be decomposed into RPA modules.

Although the network in Fig. 11 contains seven interacting nodes, it's very simple architecture and sparse input-output minor endow it with a small RPA equation: Three distinct routes from P_1 to P_7 exist in this network ($P_1 \rightarrow P_2 \rightarrow P_7$, $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_6 \rightarrow P_7$, and $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_5 \rightarrow P_6 \rightarrow P_7$), with the single network feedback loop ($P_2 \rightarrow P_3 \rightarrow P_4$) contiguous with all three. The RPA equation is limited to the three terms noted in Fig. 11. Since each term contains the kinetic multiplier (P_4) in its cycle component and the node P_4 occurs in a feedback loop, this entire network can achieve RPA through the activity of the single opposer node P_4 alone.

For illustration, we simulate a set of model equations for this RPA possibility, again choosing zero-order kinetics for the reaction rate at P_4 to guarantee opposer kinetics for this node:

$$\frac{dP_4}{dt} = k_8 P_3 - k_9 \quad (14)$$

Once again, an arbitrary choice can be made for the remaining reaction rates, so we choose the following very simple forms:

$$\frac{dP_1}{dt} = k_1 S - k_2 P_1 \quad (15)$$

$$\frac{dP_2}{dt} = k_3 P_1 - k_4 P_4 - k_5 P_2 \quad (16)$$

$$\frac{dP_3}{dt} = k_6 P_2 - k_7 P_3 \quad (17)$$

$$\frac{dP_5}{dt} = k_{10} P_3 - k_{11} P_5 \quad (18)$$

$$\frac{dP_6}{dt} = k_{12} P_3 - k_{13} P_5 P_6 \quad (19)$$

$$\frac{dP_7}{dt} = k_{14} P_2 + k_{15} P_6 - k_{16} P_7 \quad (20)$$

Solutions to these equations are given in Fig. 11 (with parameter choices listed in the figure caption). Here again, for the simple reaction kinetics we've employed, and for this particular parameter set, the time-dependent response exhibits damped oscillations as well as a high level of symmetry for fixed step increases in stimuli.

Now, the node P_3 also occurs in a feedback loop in this network and is thus a candidate opposer node. Nevertheless, its kinetic multiplier only appears in one of the three terms owing to the fact that P_3 also occurs in a route of this network (as reflected in the remaining two terms). With P_3 acting as an opposer node, the network may still achieve RPA through the simultaneous activity of a balancer module, with P_5 as a balancer node and P_6 as a

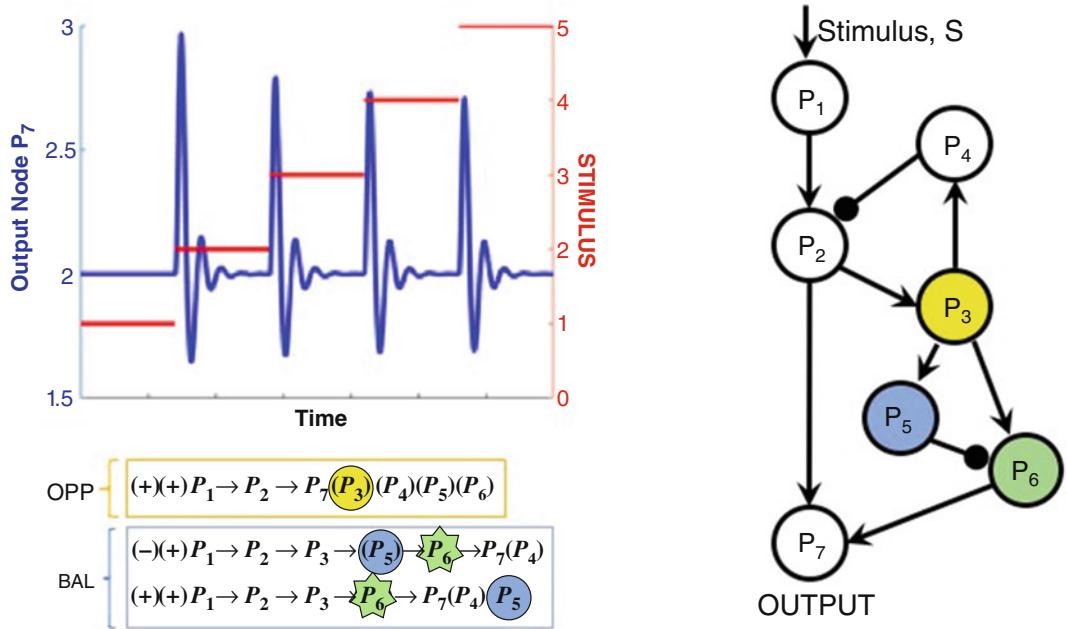


Fig. 12 Network Example 4 with an alternative RPA solution. Here the network is shown as a three-node opposer module in parallel with a three-node balancer module. Parameters: $k_1 = k_2 = k_3 = k_4 = k_5 = k_{10} = k_{11} = k_{12} = k_{13} = k_{14} = k_{15} = k_{16} = 1$. $k_6 = k_7 = k_8 = k_9 = 0.5$

connector node. This gives rise to the two-module structure depicted in Fig. 12, in which an opposer module and a balancer module coexist in parallel.

For this alternative RPA solution, with P_3 acting as an opposer node, the node P_4 may no longer possess opposer kinetics. We thus replace the rate equation for this node with the following simple form:

$$\frac{dP_4}{dt} = k_8 P_3 - k_9 P_4 \quad (21)$$

We also replace the rate equation for P_3 with a zero-order rate law for which opposer kinetics are assured:

$$\frac{dP_3}{dt} = k_6 P_2 - k_7 \quad (22)$$

Moreover for this two-module RPA solution, P_5 must operate with balancer kinetics, and P_6 with connector kinetics. We observe that Eqs. (18) and (19) already conform to these requisite reaction mechanisms (although no special reaction kinetics were required at these nodes for the previous RPA solution noted above). We solve the new set of model equations (incorporating the new rate Eqs. (21) and (22) above) and display the results in Fig. 12. Notice

that the symmetry that previously characterized the time-dependent solutions (Fig. 11) has now been lost in this two-module version of the network due to the alterations in reaction mechanisms.

4.5 Example

Network 5: A 12-Node RPA-Capable Network with Two Opposer Modules in Parallel

In our final example, we emphasize again that RPA may be achieved in larger networks with only very few constraints on individual nodes. By the diagrammatic method we've demonstrated throughout this chapter, it is straightforward to show that the simple 12-node network configuration illustrated in Fig. 13 has an RPA equation with fourteen terms. As we highlight in the diagrammatic representation of this RPA equation, the kinetic multiplier (P_9) occurs in eight of these terms, while the kinetic multipliers (P_5) and (P_6) occur in the remaining six terms. Nodes P_5 , P_6 , and P_9 are all contained in feedback loops, so they are suitable choices for opposer nodes. By proposing a set of simple modeling equations that endow, with P_9 subjected to an opposer reaction mechanism,

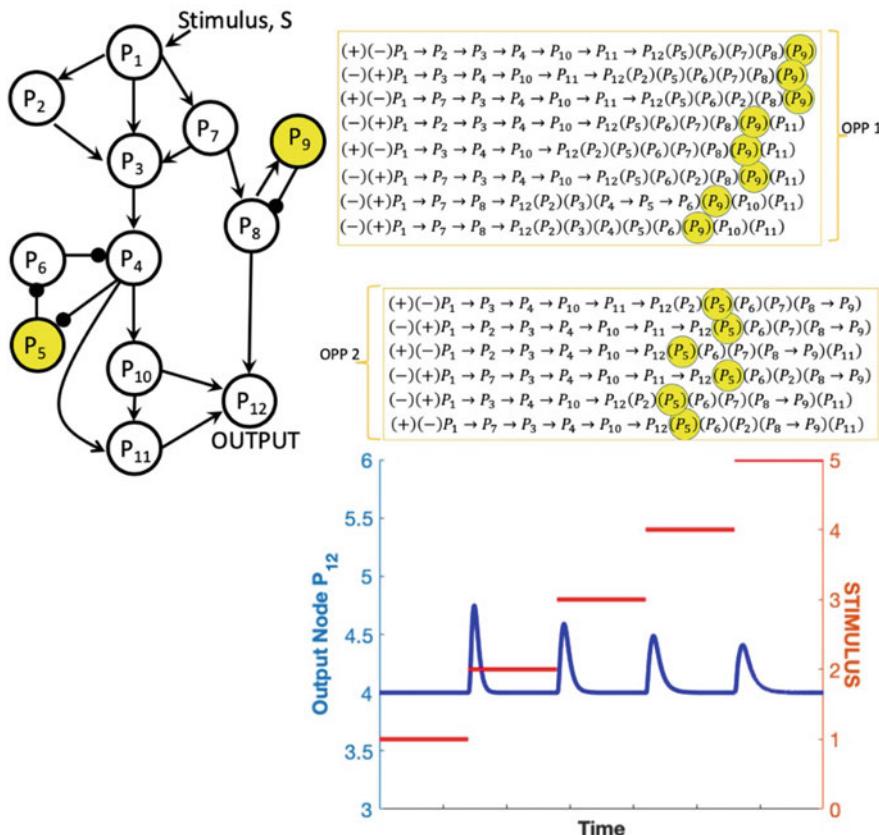


Fig. 13 Network Example 5: a 12-node RPA-capable network with two opposer modules in parallel. Parameters: $k_1 = k_2 = k_3 = k_4 = k_5 = k_6 = k_7 = k_8 = k_{15} = k_{16} = k_{22} = k_{23} = k_{24} = k_{25} = k_{26} = k_{27} = k_{28} = k_{29} = k_{30} = 1$; $k_9 = k_{10} = k_{11} = k_{12} = k_{13} = k_{14} = k_{17} = k_{18} = k_{19} = k_{20} = k_{21} = 1$

and either P_5 or P_6 also subjected to an opposer mechanism, we show in Fig. 13 that this network readily exhibits RPA through this decomposition into two parallel opposer modules.

We choose the following simple set of equations for the model simulations in Fig. 13:

$$\frac{dP_1}{dt} = k_1 S - k_2 P_1 \quad (23)$$

$$\frac{dP_2}{dt} = k_3 P_1 - k_4 P_2 \quad (24)$$

$$\frac{dP_3}{dt} = k_5 P_1 + k_6 P_2 + k_7 P_7 - k_8 P_3 \quad (25)$$

$$\frac{dP_4}{dt} = k_9 P_3 - k_{10} P_4 P_6 \quad (26)$$

$$\frac{dP_5}{dt} = k_{11} - k_{12} P_4 \quad (27)$$

$$\frac{dP_6}{dt} = k_{13} P_3 - k_{14} (P_5 + P_6) \quad (28)$$

$$\frac{dP_7}{dt} = k_{15} P_1 - k_{16} P_7 \quad (29)$$

$$\frac{dP_8}{dt} = k_{17} P_7 - k_{18} P_9 - k_{19} P_8 \quad (30)$$

$$\frac{dP_9}{dt} = k_{20} P_8 - k_{21} \quad (31)$$

$$\frac{dP_{10}}{dt} = k_{22} P_4 - k_{23} P_{10} \quad (32)$$

$$\frac{dP_{11}}{dt} = k_{24} P_{10} + k_{25} P_4 - k_{26} P_{11} \quad (33)$$

$$\frac{dP_{12}}{dt} = k_{27} P_{10} + k_{28} P_{11} + k_{29} P_8 - k_{30} P_{12} \quad (34)$$

Notice that for this particular choice of parameters, no oscillations are observed in the output signal in response to each step increase in stimulus. Indeed, the output decreases monotonically to the fixed stimulus-independent baseline level after its initial transient increase.

5 Concluding Remarks: Future Directions for RPA Theory

The analysis of the complex networks arising in nature confronts us with extraordinary technical challenges. In many biological contexts—cellular signal transduction and cellular metabolism, for instance—the underlying signaling networks are so complex and high-dimensional that their detailed topologies and reaction mechanisms are essentially unknowable. Without access to the

full-network design space for a particular functional requirement, proposing an accurate and predictive mathematical model for a particular network, or testing such a network to see if it meets a particular mathematical criterion, would require us to know (a) *all the molecules* involved in the cascades of chemical reactions that are initiated once a “stimulus” of some kind is delivered to the system (including all scaffolds, adaptor proteins, second messengers, and other ancillary molecules) and (b) the reaction mechanisms (e.g., rate laws and kinetics) of all reactions, including the role of intermediate complexes in all enzyme-catalyzed reactions, and the specific mechanisms of multisite covalent modification (which could be processive or distributive in nature, for example). In practice, we would also require information on the abundances of all reactant molecules, since the kinetics of a given reaction can vary in important qualitative ways as the involved enzymes become “saturated” by their substrates.

The topological method we discuss here for solving the RPA problem is thus an important step forward in our quest to determine how biological complexity is organized. Indeed, the strict mathematical requirement for robust networks to exist in well-defined modular structures suggests the tantalizing concept that complex networks are like snowflakes: while each is unique, all individual instances are alike in their essential structure. Our hope is that the pragmatic discussion of the general RPA solution presented here will stimulate broad interest in these new approaches, and promote the discovery of new robustness-preserving network designs for other functional requirements of biological systems.

Before closing, we wish to highlight a frontier research problem in RPA theory: The deep question of how integral control is implemented at the microscale of complex networks through intricate intermolecular interactions. Although we have employed very simple reaction rate laws for illustrative purposes in our example networks here, the molecular implementation of integral control in intracellular compartments is almost certainly vastly more complex. Significant progress has been made in this direction through the study of a special case of RPA known as absolute concentration robustness (ACR). It is now clear [5] that the Shinar-Feinberg model of ACR [30] is a simple balancer module, although the connector kinetics required for this model are buried much more deeply into the reactions of this model than the simple connector mechanisms employed in this chapter. Even the opposer node in the antithetical integral control model [22] requires a transformation of coordinates to achieve opposer kinetics [31]. In addition, the deep connection between RPA and ultrasensitivity [5] suggests a wealth of future avenues for identifying new mechanisms for the construction of robustness-promoting integrals from amid intricate biochemical reactions. The study of robustness at network micro-scales along these lines will offer us fresh insights into the biological

signalling mechanisms underpinning human disease states, along with new possibilities for pharmacological disruption, and will spawn a host of new opportunities in synthetic biology.

Acknowledgments

Robyn Araujo is the recipient of an Australian Research Council (ARC) Future Fellowship (project number FT190100645) funded by the Australian Government.

References

- Wagner A (2013) Robustness and evolvability in living systems, vol 24. Princeton University Press
- Araujo RP, Liotta LA, Petricoin EF (2007) Proteins, drug targets and the mechanisms they control: the simple truth about complex networks. *Nat Rev Drug Discov* 6(11): 871–880
- Whitacre JM (2012) Biological robustness: paradigms, mechanisms, and systems principles. *Front Genet* 3:67
- Barkai N, Leibler S (1997) Robustness in simple biochemical networks. *Nature* 387(6636): 913–917
- Araujo RP, Liotta LA (2018) The topological requirements for robust perfect adaptation in networks of any size. *Nat Commun* 9(1):1–12
- Yi TM, Huang Y, Simon MI, Doyle J (2000) Robust perfect adaptation in bacterial chemotaxis through integral feedback control. *Proc Natl Acad Sci* 97(9):4649–4653
- Hoeller O, Gong D, Weiner OD (2014) How to understand and outwit adaptation. *Dev Cell* 28(6):607–616
- Alon U, Surette MG, Barkai N, Leibler S (1999) Robustness in bacterial chemotaxis. *Nature* 397(6715):168–171
- Muzzey D, Gómez-Uribe CA, Mettetal JT, van Oudenaarden A (2009) A systems-level analysis of perfect adaptation in yeast osmoregulation. *Cell* 138(1):160–171
- El-Samad H, Goff JP, Khammash M (2002) Calcium homeostasis and parturient hypocalcemia: an integral feedback perspective. *J Theor Biol* 214(1):17–29
- Kaupp UB (2010) Olfactory signalling in vertebrates and insects: differences and commonalities. *Nat Rev Neurosci* 11(3):188–200
- Yau KW, Hardie RC (2009) Phototransduction motifs and variations. *Cell* 139(2):246–264
- Ben-Zvi D, Barkai N (2010) Scaling of morphogen gradients by an expansion-repression integral feedback control. *Proc Natl Acad Sci* 107(15):6924–6929
- Eldar A, Dorfman R, Weiss D, Ashe H, Shilo BZ, Barkai N (2002) Robustness of the BMP morphogen gradient in *Drosophila* embryonic patterning. *Nature* 419(6904):304–308
- Yadid G, Overstreet DH, Zangen A (2001) Limbic dopaminergic adaptation to a stressful stimulus in a rat model of depression. *Brain Res* 896(1–2):43–47
- Medina-Gomez G, Yetukuri L, Velagapudi V, Campbell M, Blount M, Jimenez-Linan M, Ros M, Orešić M, Vidal-Puig A (2009) Adaptation and failure of pancreatic β cells in murine models with different degrees of metabolic syndrome. *Dis Model Mech* 2(11–12):582–592
- Sturgeon JA, Zautra AJ (2010) Resilience: a new paradigm for adaptation to chronic pain. *Curr Pain Headache Rep* 14(2):105–112
- Fodale V, Pierobon M, Liotta L, Petricoin E (2011) Mechanism of cell adaptation: when and how do cancer cells develop chemoresistance? *Cancer J* 17(2):89–95
- Araujo RP, Petricoin EF, Liotta LA (2007) Mathematical modeling of the cancer cell's control circuitry: paving the way to individualized therapeutic strategies. *Curr Signal Transduction Ther* 2(2):145–155
- Geho DH, Petricoin EF, Liotta LA, Araujo RP (2005) Modeling of protein signaling networks in clinical proteomics. In: Cold Spring Harbor symposia on quantitative biology, vol 70. Cold Spring Harbor Laboratory Press, pp 517–524
- Ma W, Trusina A, El-Samad H, Lim WA, Tang C (2009) Defining network topologies that can achieve biochemical adaptation. *Cell* 138(4): 760–773
- Briat C, Gupta A, Khammash M (2016) Antithetic integral feedback ensures robust perfect adaptation in noisy biomolecular networks. *Cell Syst* 2(1):15–26
- Aoki SK, Lillacci G, Gupta A, Baumschlager A, Schweingruber D, Khammash M (2019) A

- universal biomolecular integral feedback controller for robust perfect adaptation. *Nature* 570(7762):533–537
- 24. Qian Y, Del Vecchio D (2018) Realizing ‘integral control’ in living cells: how to overcome leaky integration due to dilution? *J R Soc Interface* 15(139):20170902
 - 25. Del Vecchio D, Dy AJ, Qian Y (2016) Control theory meets synthetic biology. *J R Soc Interface* 13(120):20160380
 - 26. Francis BA, Wonham WM (1975) The internal model principle for linear multivariable regulators. *Appl Math Optim* 2(2):170–194
 - 27. Francis BA, Wonham WM (1976) The internal model principle of control theory. *Automatica* 12(5):457–465
 - 28. Ferrell JE Jr (2016) Perfect and near-perfect adaptation in cell signaling. *Cell Syst* 2(2): 62–67
 - 29. Tyson JJ, Chen KC, Novak B (2003) Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Curr Opin Cell Biol* 15(2):221–231
 - 30. Shinar G, Feinberg M (2010) Structural sources of robustness in biochemical reaction networks. *Science* 327(5971):1389–1391
 - 31. Xiao F, Doyle JC (2018, December) Robust perfect adaptation in biomolecular reaction networks. In: 2018 IEEE conference on decision and control (CDC). IEEE, pp 4345–4352



Chapter 2

Multi-Dimensional Analysis of Biochemical Network Dynamics Using pyDYVIPAC

Yunduo Lan and Lan K. Nguyen

Abstract

Biochemical networks are dynamic, nonlinear, and high-dimensional systems. Realistic kinetic models of biochemical networks often comprise a multitude of kinetic parameters and state variables. Depending on the specific parameter values, a network can display one of a variety of possible dynamic behaviors such as monostable fixed point, damped oscillation, sustained oscillation, and/or bistability. Understanding how a network behaves under a particular parametric condition, and how its behavior changes as the model parameters are varied within the multidimensional parameter space are imperative for gaining a holistic understanding of the network dynamics. Such knowledge helps elucidate the parameter-to-dynamics mapping, uncover how cells make decisions under various pathophysiological contexts, and inform the design of biological circuits with desired behavior, where the latter is critical to the field of synthetic biology. In this chapter, we will present a practical guide to the multidimensional exploration, analysis, and visualization of network dynamics using pyDYVIPAC, which is a tool ideally suited to these purposes implemented in Python. The utility of pyDYVIPAC will be demonstrated using specific examples of biochemical networks with differing structures and dynamic properties via the interactive Jupyter Notebook environment.

Key words ODE modelling, Systems dynamics analysis, Oscillation, Bistability, DYVIPAC, High-dimensional parameter space, Parallel coordinates

1 Introduction

Living cells have to adapt to constantly changing environments while maintaining homeostasis. Consequently, the biochemical networks within cells have evolved an impressive capacity to display a rich repertoire of dynamic behaviors, which enable them to undertake a range of biological tasks efficiently and robustly [1]. These include systems-level dynamics, for example, stable fixed point, sustained oscillation, switch-like, and bistable responses that have been shown to underpin important biological processes [2]. For instance, oscillations underscore the rhythmic once-per-second pulse of the cardiac pacemaker, the once-per-25-min pulse of the *Xenopus* embryonic cell cycle [3], or the once-per-day pulse

of the mammalian circadian clock [4]. On the other hand, bistability is key in generating the switch-like, all-or-none responses that underpin critical decision-making in an array of cellular processes, including cell division [5], mitotic control [6], ubiquitin elongation [7], actin cytoskeleton dynamics, and cell motility [8–10]. Thus, understanding the dynamic properties of biochemical networks, including when and how they exhibit particular dynamic behaviors, represents important goals in computational systems biology.

Quantitative and systemic analyses of network dynamics require mathematical modelling. Once a mathematical model, such as an ordinary differential equation (ODE)-based model, of the network under study is established, dynamical analysis then involves studying the relationships between the model’s parameters and the consequential dynamic behaviors of the system. Depending on specific parameter values, a network model can display a single or mixture of possible dynamic behaviors [11, 12]. To examine the parameter-to-dynamics relationship, a number of powerful computational tools have been developed, among which AUTO, XPPAUT [13], and MATCONT [14] are probably the most popular. However, a shared limitation of these tools is that they only allow dynamical analysis and visualization of the results to be performed at very low dimensions. That is, one can typically only interrogate the effects of varying one or two model parameters on the systems dynamics, while fixing all the remaining parameters at constant values; and the result is often projected onto two-dimensional (2D) plots [11, 15, 16]. However, due to the nonlinearity and multifactorial determinant of network dynamics, such results may differ substantially when the fixed parameters are allowed to vary themselves, i.e., taking on new values, making low-dimensional analysis prone to inaccurate generalization at higher dimensions [17]. Furthermore, the low-dimensional perspective also restricts our ability to derive global parametric conditions (i.e., involving multiple control parameters) that govern a particular dynamic behavior(s). Hence, to gain a holistic understanding of the causal linkages between parameters and dynamics, and to discover useful governing patterns between the parameters, the ability to interrogate and visualize network behaviors in the parameter hyperspace becomes essential.

In this chapter, we present a practical and interactive guide to the multidimensional exploration, analysis, and visualization of network dynamics using pyDYVIPAC, an updated Python implementation of DYVIPAC (Dynamics Visualisation based on Parallel Coordinates). We will first provide an overview of pyDYVIPAC, followed by detailed instructions of how to set up and run pyDYVIPAC using the interactive Jupyter Notebook environment. Finally, we will demonstrate the key features and results of

pyDYVIPAC through specific examples of biochemical networks with different dynamical properties.

2 Overview of pyDYVIPAC

pyDYVIPAC is a tool for causally linking system parameters with dynamic behaviors through multidimensional analysis and visualization. pyDYVIPAC is a Python 3 implementation and an updated re-implementation of DYVIPAC [17], which provides an integrated and interactive analysis environment through Jupyter Notebook. pyDYVIPAC is also parallel-computing enabled via joblib, which allows multicore processing of large parameter space analyses.

The overall workflow of pyDYVIPAC is given in Fig. 1. Given a biochemical network of interest, pyDYVIPAC starts with an ordinary differential equation (ODE) model of the network in the Systems Biology Markup Language (SBML) format as the input. It facilitates global interrogation of network behaviors by allowing one to systematically sample a large number of parameter sets in the high-dimensional parameter space, by varying multiple model parameters at the same time within defined ranges. Unlike other tools, the number of varying parameters in pyDYVIPAC can be significantly greater than two and in theory has no upper limit other than the size of the model (i.e., as many as the number of free parameters in the model). Then, for each sampled parameter set, pyDYVIPAC performs local stability analysis to determine the possible dynamic behavior(s) of the system at that parameter values. This enables partitioning of the parameter space into regions with specific dynamic behaviors, e.g., monostability, oscillations, bistability, excitability, etc. Finally, pyDYVIPAC visualizes the results using powerful parallel coordinate (PC) plots, which provides a powerful way to display complex high-dimensional data in 2D and to pinpoint hidden relational patterns between the sampled parameters. The PC visualization can be done within pyDYVIPAC using in-built PC functions or by employing a free and specialized tool called XDAT (<https://www.xdat.org/>), where the latter provides richer features (Fig. 1).

3 Requirements for Running pyDYVIPAC

3.1 Installation Requirements

- *Python 3.8.10*: Python is required for running pyDYVIPAC. It is available at <https://www.python.org/>. Python 3.8 comes with pip, which is a package management system that allows for easy installations of useful external libraries written in Python. You will need to install the following python packages either via pip

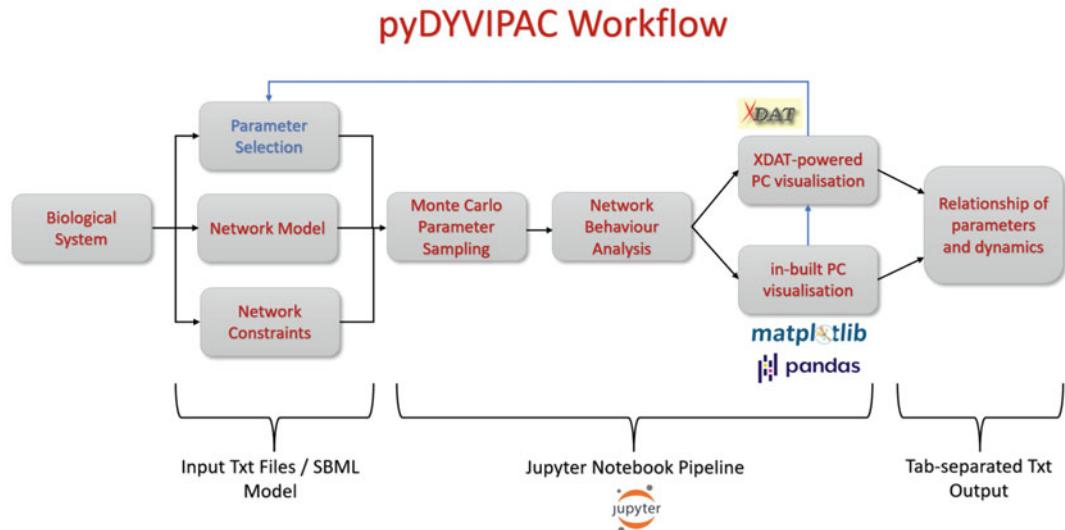


Fig. 1 Workflow of pyDYVIPAC. The pyDYVIPAC workflow takes three files derived from a biological system: network model, parameter space selection, and network constraints, as inputs. Then, the selected parameters are processed via Monte Carlo sampling, and at each parameter set, local stability analysis is performed to determine the network dynamic behavior, which are categorized by different numeric output codes. The results containing values of sampled parameters and corresponding dynamic outputs are generated as .txt files. The parameter values matching a specific behavior (output code) can then be visualized using either the in-built visualization module or an external software, XDAT. PC parallel coordinates

or alternative package managers such as anaconda for running the pyDYVIPAC workflow.

- *libRoadRunner*: A high-performance and portable simulation engine for systems biology (<https://www.libroadrunner.org/>). pyDYVIPAC uses the Python API of libRoadRunner to load SBML files of ODE models and perform stability analysis by computing the eigenvalues of the Jacobian matrix at specific parameter sets. pyDYVIPAC has been tested using libRoadRunner 2.2.2, so we recommend this version. You can install libroadrunner in the command line via the following:

```
pip install libroadrunner==2.2.2
```

- *Jupyter Notebook*: A web-based interactive computing platform, which is the preferred method for running pyDYVIPAC. Jupyter Notebook can be installed using the following command:

```
pip install notebook==6.4.12
```

Note that if this installation method becomes outdated, the installation of the latest version can always be found on the official website: <https://jupyter.org/install>.

- *Pandas*: A powerful Python library for data manipulation and analysis. However, here we will be using a specific Pandas function call to plot parallel coordinates in the Jupyter interactive window. Pandas can be installed using the following command:

```
pip install pandas==1.4.3
```

- *[Optional] Matplotlib*: A Python library for data visualization and graphical plotting. Matplotlib is the underlying plot-rendering engine for pandas parallel coordinates. This library should be automatically installed upon the installation of pandas. However, if it was not installed, it can be installed using the following command:

```
pip install matplotlib==3.5.2
```

- *[Optional] Numpy*: A high-performance library for scientific computing. It is not necessary to install numpy separately, since this package will be installed along with the installation of libRoadRunner. pyDYVIPAC was tested using the following numpy version:

```
pip install numpy==1.23.1
```

- *[Optional] Joblib*: A pipeline optimization Python package that comes with parallel computing features. Since pyDYVIPAC is configured in Jupyter, which has conflicts with Python's default multiprocessing library, joblib is a simple alternative solution. The tested version of joblib is as follows:

```
pip install joblib==1.1.0
```

- *[Optional] Antimony*: A model definition language which can be compiled into SBML format, the input model file format for pyDYVIPAC. Antimony has a user-friendly syntax compared to SBML. You can install this package if you need to create a custom model:

```
pip install antimony==2.13.2
```

- *[Optional] Tellurium*: If you would like to simulate your model using pyDYVIPAC-generated parameter sets, then Tellurium provides an integrated environment for this purpose. Tellurium can directly read Antimony syntax and use the same underlying ODE solving engine as pyDYVIPAC (libRoadRunner). You can install Tellurium via pip as follows:

```
pip install tellurium==2.2.3
```

- *XDAT*: A powerful software for creating parallel coordinate plots of high-dimensional with rich features. We recommend this tool for displaying and visually exploring the output results from pyDIVIPAC. XDAT can be installed at <https://www.xdat.org/index.php?ref=download>.

3.2 Description of Key Input Files for pyDYVIPAC

In order to run pyDYVIPAC, the following input files are required (see workflow in Fig. 1).

- *Input model file:* This is the file describing an ODE model of the network being studied. It must be in the standard SBML format (.xml file, Level 2, Version 5, or above). We recommend to build the model using Antimony and then convert it to SBML format using a feature provided by Antimony.
- *Input parameter file:* This .txt file contains a list of selected model parameters to be explored, and the corresponding range with lower and upper bounds for each parameter (see Notes 1 and 2 for a guide on setting these parameters and ranges).

This is an example of the parameter file:

```
k1 0 1
k2 1 10
k3 10 100
k4 10 100
S 0 2000
```

Each row contains the parameter name, lower bound, and upper bound separated by a space. In this example, kinetic parameter $k1$ will be randomly explored (sampled) within the range $[0, 1]$, $k2$ within the range $[1, 10]$, etc., while S represents concentration of a model species that will be explored within the range $[0, 2000]$.

- *Input constraint file:* This .txt file contains user-defined constraints related to the model's state variables.

Depending on the model, this file is required or not. For example, if the model contains conservation laws, these have to be specified in the constraint file. Two types of constraints are possible: equal = and less than <, or for more details see Note 4 on Model Reduction. An example of a conservation law is as follows:

```
S + pS = Stot
```

Here, S represents a model species. S and pS are the unphosphorylated and phosphorylated moieties of protein S in the model, respectively. Their total concentration is conserved: $Stot$ is a constant within the model (and can be subject to sampling as the kinetic parameters).

Moreover, if needed, one can use the constraint file to fix the maximum value of the model species within which pyDYVIPAC

searches for steady state (or from which to start searching the steady state). If a model species is not constrained using this file, then the default search range will be set to [0, 1000]. An example of these constraints is as follows:

```
s < Stot
s + pS < Stot
```

4 Running pyDYVIPAC

4.1 Setting Up and Running pyDYVIPAC Jupyter Notebook

In this section, we will provide a step-by-step guide for running pyDYVIPAC on your computer. You should be able to download all the necessary files from the “pyDYVIPAC” repository on the Nguyen lab github page (<https://github.com/ NguyenLabNetworkModeling/pyDYVIPAC>). For beginners, we recommend downloading them to a default user file location, such as the “Downloads” folder, and then uncompressing the package there. This is for easier access of the package from the Jupyter local server.

After downloading and uncompressing (as well as ensuring all requirements have been installed), we launch Jupyter Notebook by typing the following in the command line:

```
jupyter notebook
```

This should create a Jupyter homepage on your web browser.



The screenshot shows the Jupyter Notebook interface with the following details:

- Header:** The title bar says "jupyter". On the right are "Quit" and "Logout" buttons.
- Menu Bar:** "Files", "Running", "Clusters".
- Toolbar:** "Select items to perform actions on them.", "Upload", "New", "File size".
- File List:** A table showing the contents of the current directory:

	Name	Last Modified	File size
<input type="checkbox"/>	/	15 minutes ago	
<input type="checkbox"/>	Videos	7 hours ago	
<input type="checkbox"/>	Zotero	8 hours ago	
<input type="checkbox"/>	Downloads	2 days ago	
<input type="checkbox"/>	Qsync	9 months ago	
<input type="checkbox"/>	OneDrive	9 months ago	
<input type="checkbox"/>	Links	9 months ago	
<input type="checkbox"/>	Saved Games	9 months ago	
<input type="checkbox"/>	Music	9 months ago	
<input type="checkbox"/>	Searches	9 months ago	
<input type="checkbox"/>	Contacts	9 months ago	
<input type="checkbox"/>	Favorites	9 months ago	
<input type="checkbox"/>	Desktop	9 months ago	
<input type="checkbox"/>	package-lock.json	a year ago	19.2 kB
<input type="checkbox"/>	node_modules	a year ago	
<input type="checkbox"/>	3D Objects	2 years ago	
<input type="checkbox"/>	git	2 years ago	

DYVIPAC Jupyter Notebook Process Chain

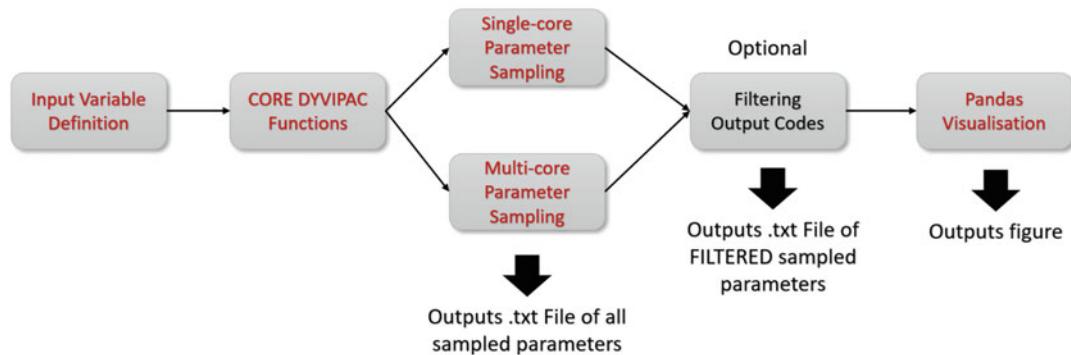


Fig. 2 The pyDYVIPAC workflow process chain. Each block of text represents one or a set of joint code blocks to be run in sequence. Users may choose either single-core or multi-core processing of parameter sampling, and then optionally filter out specific output codes. The thickened black arrows represent outputs produced by the pipeline

Navigate to the directory where you have unzipped the pyDYVIPAC package, and enter the “pyDYVIPAC-main” folder. The entire pyDYVIPAC workflow is encapsulated in the Jupyter notebook file called *pc_2.ipynb* in this folder, which you should be able to see.

The general process of running the pipeline is (see also Fig. 2):

1. Define the model input parameters.
2. Run the core pyDYVIPAC code block(s).
3. Decide if result filtering is required.
4. Visualize the output data within pyDYVIPAC notebook or externally using XDAT.

Upon entering the notebook, you should first see the “input data” section. This is where we define the input parameters for pyDYVIPAC:

input_model_file	str a filename specifying the input SBML model file path. <i>See Subheading 3.2</i> for more explanation of this file
input_parameter_file	str a filename specifying the parameter file path. <i>See Subheading 3.2</i> for more explanation of this file
input_constraint_file	str a filename specify the file with the species constraint file path. <i>See Subheading 3.2</i> for more explanation of this file
random_seed	int an integer number specifying a random seed (e.g., 0)

(continued)

number_of_samples	int an integer number specifying how many parameter sets to sample within the ranges included in the parameter file (see Note 3 for a guide on how to set this number)
log_search	bool search the natural logarithm of the parameter space. Also note that the parameter values in the output file will be the natural logarithm of the actual parameters
silence_program	bool this option is to silence the command line output of the program
output_file_name	str filename specifying the name of the output file of the analysis. The default name is output.txt followed by the random seed number, if specified

There are a few examples of how to define the inputs in the “Example {number}” sections in the notebook. We will explore these examples in the next section (Subheading 5) and illustrate the specific procedure of running pyDYVIPAC under a variety of biochemical network contexts.

4.2 Displaying and Analyzing pyDYVIPAC Outputs

The output of pyDYVIPAC (stored in the output.txt file) will be a table, delimited by spaces, where the columns (from the second column) indicate the selected parameters and the rows are the randomly sampled parameter values (see Fig. 3 for an example). The first column “group” contains for each row an integer code (0 to 8) that indicates the result from local stability analysis of the model at that parameter set. Specifically, 0 = no steady state could be found; 1 = one stable steady state found (indicating fixed point);

group	k1	k2	k3	kd1	kd2	kd3
1	0.5488135	0.67781654	0.31179588	0.45327775	0.20062975	0.15519041
1	0.71518937	0.27000797	0.69634349	0.38702367	0.46464571	0.18651743
1	0.60276338	0.73519402	0.37775184	0.16657258	0.04980747	0.26248522
2	0.54488318	0.96218855	0.17960368	0.04055069	0.47265077	0.37529751
1	0.4236548	0.24875314	0.02467873	0.20362059	0.43474427	0.16675373
6	0.64589411	0.57615733	0.06724963	0.11611707	0.2270812	0.46207938
2	0.43758721	0.59204193	0.67939277	0.06624382	0.16335044	0.43115927
2	0.891773	0.57225191	0.45369684	0.02671359	0.11637206	0.02434515
6	0.96366276	0.22308163	0.53657921	0.36279718	0.30723235	0.12682126
0	0.38344152	0.95274901	0.89667129	0.00571373	0.0165373	0.22306776
1	0.79172504	0.44712538	0.99033895	0.38529037	0.00780303	0.05231394
6	0.52889492	0.84640867	0.21689698	0.07347332	0.21439786	0.17423799
6	0.56804456	0.69947928	0.6630782	0.03976104	0.03403704	0.37004876
1	0.92559664	0.29743695	0.26332238	0.04480152	0.12597049	0.34025724
1	0.07103606	0.81379782	0.020651	0.3360239	0.11058046	0.31119221

Fig. 3 An illustrative example of the output file of pyDYVIPAC

2 = one unstable steady state found (indicating oscillations); 3 = two stable steady states found; 4 = one stable and one unstable steady states found; 5 = two unstable steady states found; 6 = two stable and one unstable steady states found (indicating bistability); 7 = one stable and two unstable steady states found; and 8 = unclassified. Based on experience, code values 1, 2, and 6 are more often encountered while exploring common signalling networks with feedback loops.

In principle, the output data can be displayed using any suitable visualization techniques, from simple scatter plots to more sophisticated heatmaps. However, here we emphasize on the application of parallel coordinate (PC) plots as a particularly useful way to visualize these multidimensional data because it facilitates identification of patterns within the data, as we will see in the examples in Subheading 5.

5 Demonstration of pyDYVIPAC Using Specific Examples

In this section, we will carry out analysis using pyDYVIPAC and demonstrate its features in a more interactive manner using two specific examples of biochemical networks encountered in cells. These are the (i) the negative-feedback Goodwin system and (ii) the signalling cascade with mixed negative and positive feedback loops.

5.1 Example 1: The Negative Feedback Goodwin System

5.1.1 Setting Up

We will use the negative feedback Goodwin model as a first detailed example for running pyDYVIPAC (Fig. 4a). In *pc_2.ipynb*, navigate to “Example 1: The negative feedback Goodwin model.” Here we can see all of the input parameters for pyDIVYPAC for this example:

```
input_model_file = 'models//NFB//NFB.xml'
input_parameter_file = 'models//NFB//NFB_params_simple.txt'
input_constraint_file = ''
random_seed = 0
number_of_samples = 1000
log_search = False
silence_program = False
add_header_to_output = True
output_file_name = f'output_example1_1_{random_seed}.txt'
```

Before we execute the code block, let us take a closer look at the specific files that were passed onto the notebook. The `input_model_file` variable describes the location of the model file. In the Jupyter Notebook server, we can navigate to the models folder

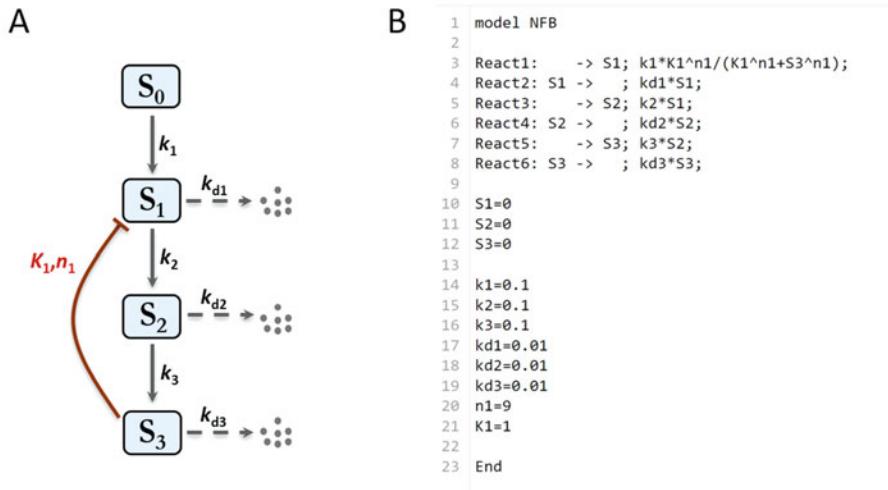
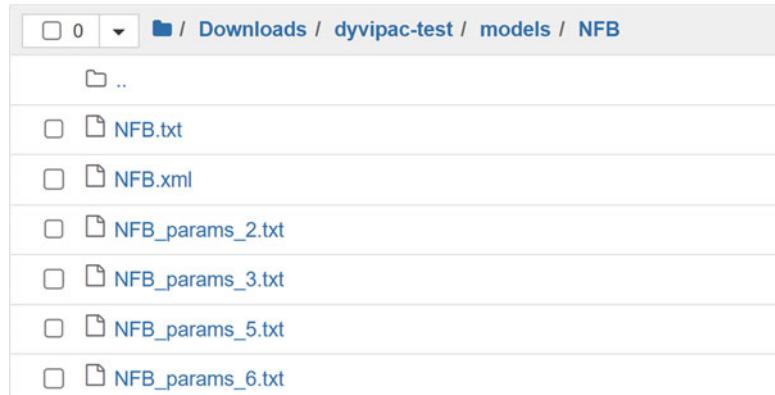


Fig. 4 The negative feedback Goodwin network. (a) A schematic diagram of the network. (b) Model definition of the network in Antimony, including model species, reactions and rates, parameters, and their nominal values. The structure of this particular file is that the model species and reactions are defined with names such as “React1.” Then, initial conditions are set. Finally, the parameters are defined last

in the pyDYVIPAC stored directory and enter the “NFB” folder.

Select items to perform actions on them.



We can see that *NFB.xml* and *NFB_params_simple.txt* are the two files we defined in the notebook. Let us first decipher the model definition. However, we are not going to explore the *NFB.xml* file, since it is quite complicated to decipher raw SBML text. Instead, we are going to view the Antimony definition of the model in the file *NFB.txt*, shown in Fig. 4b.

5.1.2 Parameter Sampling and Dynamics Processing

In pyDYVIPAC, any free model parameters, e.g., kinetic parameters or parameters representing conserved total concentrations of species, can be sampled. The parameters that are not sampled will be fixed at the nominal values as defined in the model definition file

(Fig. 4b). For instance, we are sampling two degradation kinetic parameters “kd1” and “kd2” in this specific example via the parameter file *NFB_params_2.txt*.

```
1 kd1 0 0.05
2 kd2 0 0.05
3
```

This will mean all other parameters remain fixed to the values defined within *NFB.xml* and its precompiled antimony version *NFB.txt*. In the parameter file above, we can see that both “kd1” and “kd2” are sampled from the range of 0 and 0.05.

Now that we are familiar with the model input files, let us execute this input code block. We should see the block is now “In [1].” This means that it is the first input block for this notebook.

Example 1 - The negative feedback Goodwin model

Example 1.1 - finding oscillations by samplings kd1 and kd2

```
In [1]: M input_model_file = 'models//NFB//NFB.xml'
input_parameter_file = 'models//NFB//NFB_params_2.txt'
input_constraint_file = ''
random_seed = 0
number_of_samples = 1000
log_search = False
silence_program = False
add_header_to_output = True
output_file_name = f'output_example1_1_{random_seed}.txt'
```

Next, we will run the code block under “The pyDYVIPAC program” to define all necessary functions needed for pyDYVIPAC. Note that “In [2]” is now next to the code block.

The pyDYVIPAC program

Run the block below after running the following input block

```
In [2]: # -----
# "Dynamics Visualization based on Parallel Coordinates using Python (pyDYVIPAC)"
# -----"
```

We now move on to the actual processing part. pyDYVIPAC offers two versions of processing: “single-core” and “multi-core.” Single-core processing is good for lightweight tasks that can be run in the background, while multi-core processing utilizes all CPU cores in the machine to crunch computationally heavy jobs. Since this example is relatively lightweight, we can use single-core processing to proceed.

Run the code block right under “Single-core processing of DYVIPAC.”

Single-core processing of DYVIPAC

Following running the block above, run either this block or the block below

```
In [*]: M #-----DONE READING FILES-----
#generate the parameter sets and then test them
nOfSamples = numerofsamples
```

We should see an “In [*]” next to the code block, indicating that it is still running. When it is complete, “In [*]” will turn into “In [3],” and the following message will be displayed at the bottom of the code block output:

```
Done. Results printed in the file output_example1_1_0.txt
```

Congratulations, you have just completed the core processes of running pyDYVIPAC. The output file *output_example1_1_0.txt* should be located under the DYVIPAC directory. It should contain 1000 sampled sets of the parameters “kd1” and “kd2” for the *NFB*.*xml* model with an integer code indicating the dynamic property attached to each set (as in *see Fig. 3*).

5.1.3 Visualization of Results

We can perform a quick visualization of the output data by going to the “Visualisation” section of the notebook. Under “Visualisation data input,” change the **plotting_file_name** to “output_exam- ple1_1_0.txt.” Then run the code block.

Visualisation Data Input

```
In [34]: M plotting_file_name = 'output_example1_1_0.txt'
normalise = False
initial_params_file = '' # necessary if normalise == True
figure_size = (12, 6)
image_quality_dpi = 80
```

Finally, we run the code block underneath the heading “Pandas Parallel Coordinate Plot.” This generates a PC lot of the results using the Python *pandas* library (Fig. 5). We can see that stable fixed point (code 1) and oscillations (code 2) are the major dynamics in this Goodwin system, as expected. Importantly, we can discern interesting patterns between kd1 and kd2 that give rise to oscillation (purple lines, Fig. 5), that is, (i) oscillations only occur over a fairly restricted range of both parameters (roughly between 0.005 and 0.02), and (ii) comparable kd1 and kd2 values tend to facilitate oscillations, consistent with previous findings [18, 19]. This is in contrast to the parameter conditions that give rise to stable fixed points (green lines, Fig. 5). The parallel coordinate plot produced by the *pandas* library provides a quick visualization that would be useful for preliminary assessment of the results; however, for deeper analysis, we recommend to use XDAT, described next.

To visualize the data in XDAT, you need to have it installed (*see Subheading 3.1*). Then the output file from pyDYVIPAC could be imported to XDAT. To do this, launch XDAT, under “Data - import data with headers,” navigate to the output file “output_exam- ple1_1_0.txt,” and open. The data should be loaded in and appear like so in Fig. 6.

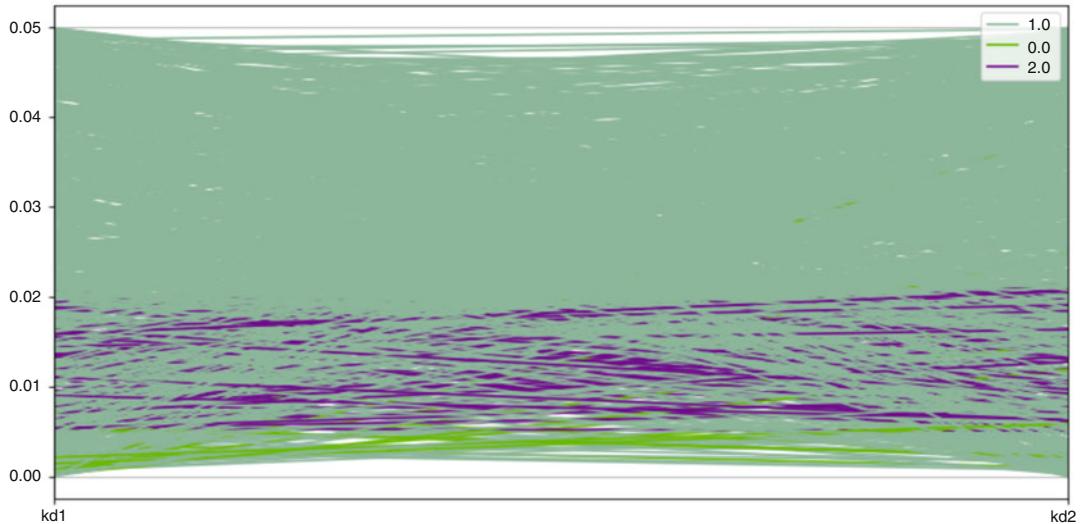


Fig. 5 A pandas-based parallel coordinate plot displaying pyDYVIPAC results from sampling kd1 and kd2. One thousand parameter sets were sampled. Different systems dynamics are indicated by the corresponding output codes (0, 1, and 2), as explained in Subheading 4.2. The number of sets returning 0 (numerical error) is very low (22 in our run). Of highlight, the purple lines indicate parameter sets giving rise to oscillations

5.1.4 Explore the Hyperspace in Higher Dimensions

Next, to create the parallel coordinate plot in XDAT, go to “chart - create parallel coordinates chart.” This will create a new window with a parallel coordinate plot. To display different output codes with different color, right click on any blank space in the plot, and select “apply colour gradient.” The result should look like Fig. 7. We can see now the output codes are represented by a separate vertical axis (“group”), and it is easier to distinguish parameter sets belonging to different dynamic behaviors. We recommend the readers to explore other useful features of XDAT.

For practical purposes, we started the demonstration of pyDYVIPAC in the previous section by exploring the parameter space in 2D (varying kd1 and kd2). However, a key strength of pyDYVIPAC is that it allows dynamical analysis in essentially any higher dimensions. Here, we demonstrate the power of DYVIPAC in multidimensional analysis by sampling six parameters at a time using the Goodwin system. So let us navigate back to the beginning of the notebook and run the code block underneath “Example 1.2 - more than two parameters.”

```
In [1]: M input_model_file = 'models/NFB/NFB.xml'
input_parameter_file = 'models/NFB/NFB_params_6.txt'
input_constraint_file = ''
random_seed = 0
number_of_samples = 5000
log_search = False
silence_program = False
add_header_to_output = True
output_file_name = f'output_example1_2_{random_seed}.txt'
```

#	group	kd1	kd2
1	1.0	0.02744068	0.02964401
2	1.0	0.03575947	5.0318E-4
3	1.0	0.03013817	0.02379131
4	1.0	0.02724416	0.03543852
5	1.0	0.02118274	0.00219877
6	1.0	0.03229471	0.04397607
7	1.0	0.02187936	0.02600407
8	1.0	0.04458865	0.00153305
9	1.0	0.04818314	0.01122068
10	1.0	0.01917208	0.04768378
11	1.0	0.03958625	0.02911599
12	1.0	0.02644475	0.00537363
13	1.0	0.02840223	0.01437723
14	1.0	0.04627983	0.02283518
15	0.0	0.0035518	0.0010475
16	1.0	0.00435646	0.02058078
17	1.0	0.00101092	0.02447293
18	1.0	0.04163099	0.01218389
19	1.0	0.03890784	0.02943195
20	1.0	0.04350061	0.03766201
21	1.0	0.04893092	0.01179171
22	1.0	0.03995793	0.031025
23	1.0	0.02307397	0.03198111

Fig. 6 Screenshot of a table displaying imported data in XDAT. The data was imported from the output file “output_example_1_0.txt”

As you can see, we are still using the same input model file, but this time we are using a different parameter file while generating 5000 samples instead of 1000. If we have a closer look at the input parameter file, we can see that we are sampling six parameters in *NFB_params_6.txt*:

```

1 k1 0 1
2 k2 0 1
3 k3 0 1
4 kd1 0 0.5
5 kd2 0 0.5
6 kd3 0 0.5
7

```

After running the code block, we can follow the same procedure as in the previous example (Subheadings 5.1.1, 5.1.2, and 5.1.3) by running the pyDYVIPAC program code block. However,

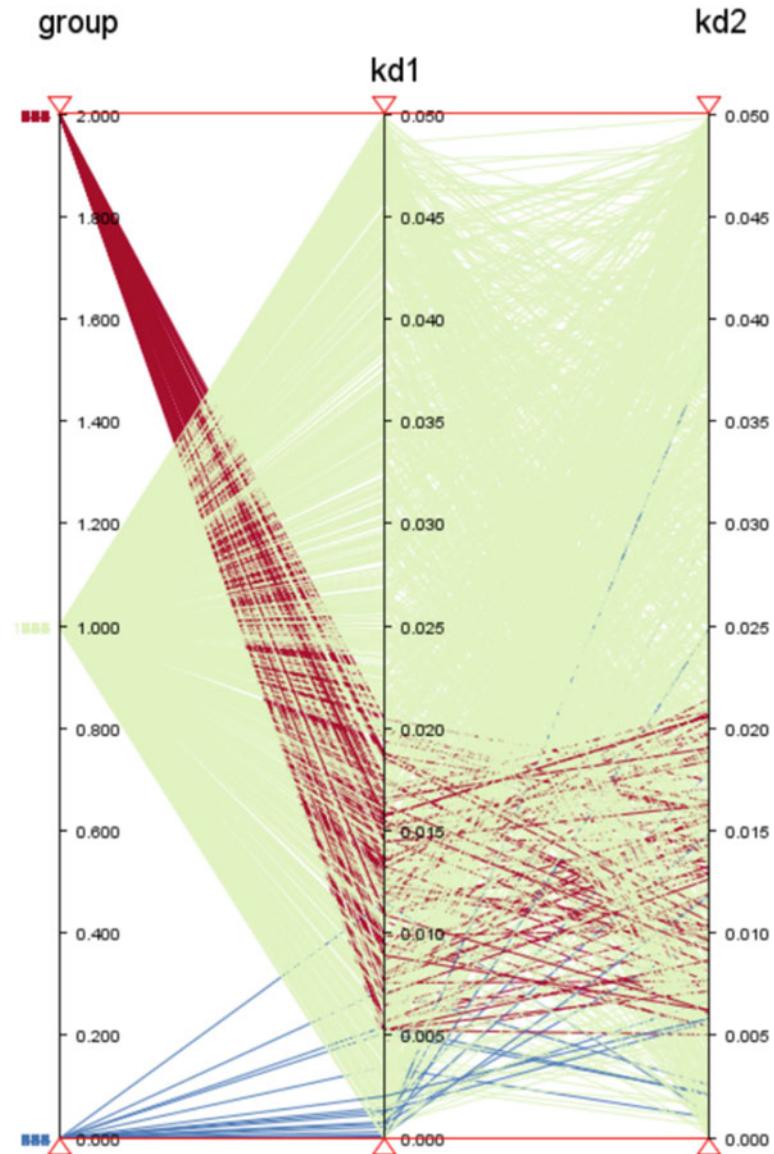


Fig. 7 Parallel coordinate plot displaying the results in XDAT. The data imported from the output file “output_example_1_0.txt” was used

when it comes to processing, you may consider running the “Multi-core processing of pyDYVIPAC” instead of the single-processing one to speed up computation time. The multiprocessing code block is right underneath the single-processing code block. You will need to install *joblib* to enable multi-core processing (*see Subheading 3.1* for more information). Follow the code block run, you should be able to produce the following result:

```

Model models//NFB//NFB.xml loaded correctly
Output file: output_example1_2_0.txt
Parameter search ranges file: models//NFB//NFB_params_6.txt
Number of parameters to search: 6
Random seed used: 0
Number of samples to test: 5000 Done. Results printed in the
file output_example1_2_0.txt

```

The output file `output_example1_2_0.txt` can then be loaded onto the embedded pandas visualization section for a quick visualization. Since we are sampling k_1 , k_2 , and k_3 at a different range compared to k_{d1} , k_{d2} , and k_{d3} (0 to 1 vs. 0 to 0.5), we recommend setting normalization to True in the visualization inputs. The normalization of output data is more meaningful if we restrict the normalization to the initial sampling minimum and maximum of each parameter. Therefore, we will also need to provide the file path for `initial_params_file` as “models//NFB//NFB_params_6.txt.” Ensure `normalise` is set to True.

Visualisation Data Input

```

In [38]: plotting_file_name = 'output_example1_2_0.txt'
normalise = True
initial_params_file = 'models//NFB//NFB_params_6.txt' # necessary if normalise == True
figure_size = (12, 6)
image_quality_dpi = 80

```

We obtained the plot as shown in Fig. 8 after running the visualization code block.

It is indeed difficult to see any patterns in the plot in Fig. 8, mainly due to the default color mapping of pandas’ plotting function, as well as its lack of support for line opacity. In situations like this, we should export the data to XDAT, which has a markedly

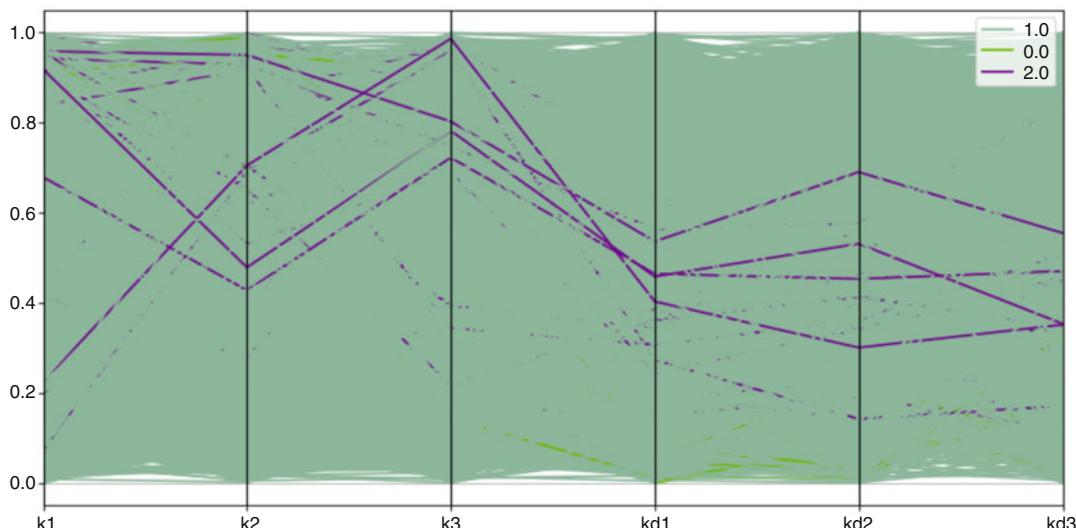


Fig. 8 Pandas-based parallel coordinate plot displaying pyDYVIPAC results. The data from the output file “output_example1_2_0.txt” was used

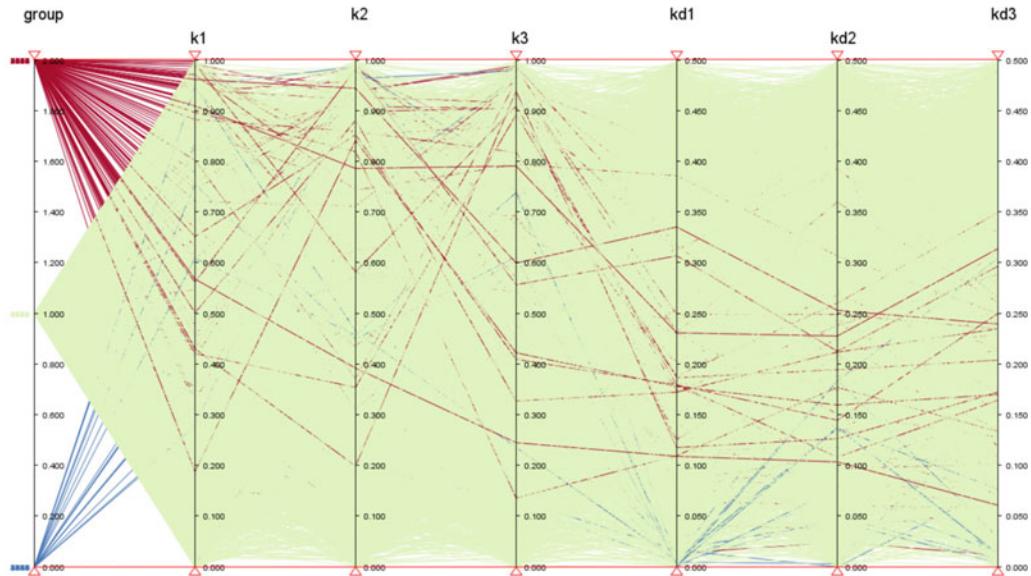
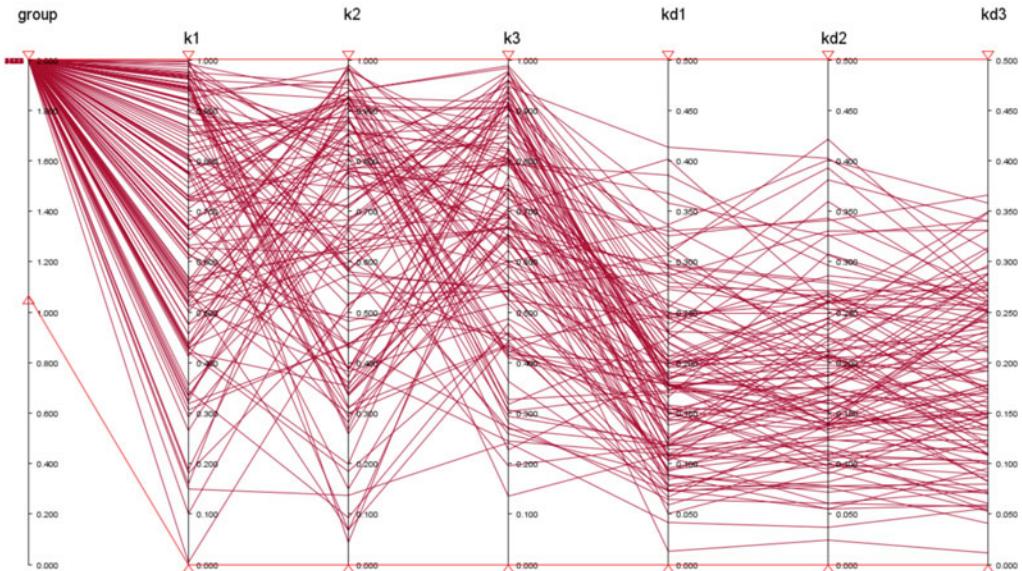
A**B**

Fig. 9 Parallel coordinate plot displaying the results in XDAT. The data imported from the output file “output_example1_2_0.txt” was used. **(a)** PC plot displaying all the dynamics groups. **(b)** PC plot displaying only parameter sets leading to oscillations by using the filtering feature in XDAT

better rendering backend as well as automatic normalization. Figure 9a displays the PC plot for the same data in XDAT.

We can see the oscillation pattern slightly better compared to *pandas* due to better rendering of the parallel coordinate lines for each parameter. We can do more, however, by toggling the triangle beneath the “group” parameter upward. By doing this, we can filter

out all of the output codes 0 and 1, leaving only output code 2, which represents potential oscillation behaviors of the model (Fig. 9b). This PC plot captures the salient features of the parameter space which produces oscillation behaviors in the *NFB* model in a visually appealing way. Importantly, we observe that even in the six-dimensional space, oscillations are most likely when the degradation rates (k_{d1-3}) take comparable values within rather restricted ranges of the initially sampled values, regardless of what happened to k_{1-3} , indicating this is a robust condition for oscillations. Another way to interpret this is that degradation rates that are either too low or too high are unlikely to lead to oscillations.

5.2 Example 2: The Mixed Feedback Phosphorylation Cascade

Previously, we used the Goodwin model to illustrate the pyDYVIPAC workflow in detail. In this example, we will use a slightly more complicated biochemical network model, the mixed-feedback phosphorylation cascade (Fig. 10a), to demonstrate some additional features of pyDYVIPAC.

Example 2 - The mixed feedback signalling cascade

```
In [3]: # The mixed model
input_model_file = 'models\Mixed\mixed.xml'
input_parameter_file = 'models\Mixed\mixed_p2.txt'
input_constraint_file = 'models\Mixed\mixed_con.txt'
random_seed = 0
number_of_samples = 20000
log_search = False
silence_program = False
add_header_to_output = True
output_file_name = f'output_example3_2_{random_seed}.txt'
```

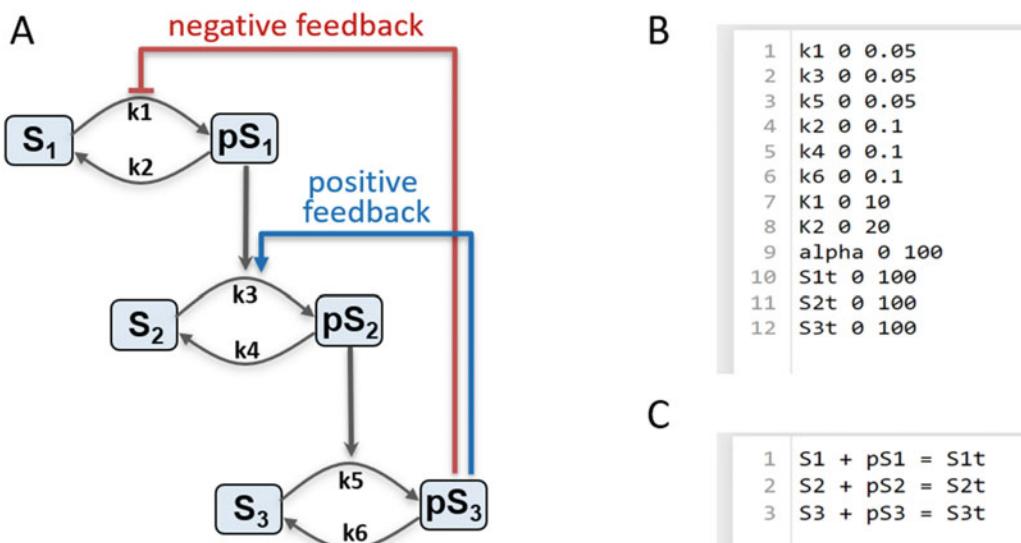


Fig. 10 The mixed-feedback signalling cascade. (a) A schematic diagram of the mixed-feedback signalling cascade consisting of three phosphorylation layers, regulated by a negative and a positive feedback loop. (b) The parameter file *mixed_p2.txt*. (c) The constraint file *mixed_con.txt*

The mixed-feedback model *mixed.xml* contains both a positive feedback as well as negative feedback loop in the three-tier phosphorylation cascade, as depicted in Fig. 10a. These intricate connections are likely to result in unintuitive behaviors of the system, which may potentially underpin key cellular processes and determine cell fates. In this example, we will sample up to 12 parameters in *mixed.xml*, which is defined in the parameter file *mixed_p2.txt*, shown in Fig. 10b.

Given the large sampling space (12 parameters, different ranges) we are working with, it is important to generate sufficient samples to thoroughly explore the parameter space. Therefore, in the code block, we will set `number_of_samples` to 10,000. In addition, you will notice that we have a constraint file *mixed_con.txt* as part of the input, which is shown in Fig. 10c. In this protein phosphorylation system, we assume that the total concentrations of the proteins in each tier is conserved over time.

With key input parameters explained, let us run the code block chains. We recommend the users to use multi-core processing if possible, since the generation of 10,000 valid samples is not trivial even for the powerful libRoadRunner engine.

```
Model models\Mixed\mixed.xml loaded correctly Output file:
output_example3_2_0.txt
Parameter search ranges file: models\Mixed\mixed_p2.txt
Number of parameters to search: 12
Random seed used: 0
Number of samples to test: 10000
Constraints file: models\Mixed\mixed_con.txt Number of constraints found in constraint file: 3
Moity conservation laws found (should be equal as line above):
3
```

The output is saved as *output_example2_2_0.txt*. Visualizing 10,000 samples all at once is difficult for any plotting software since it can easily overload their rendering engine. This is where filtering can become useful.

Under “Filter out specific output codes” in the notebook, we should see a few pre-defined code blocks. We know that the output code for potential oscillation and bistability behaviors is 2 and 6, respectively. In the figure below, we can see that the `input_file_name_filter` variable is set to `output_file_name`. This is because we had previously defined `output_file_name` as “*output_example2_2_0.txt*” in our input code blocks. If you want to load different file-names, we can simply change `input_file_name_filter` to a desired file path string. Run the code block underneath “Filter out oscillation”

as well as “Filter out bistability.”

Filter out oscillation

```
In [26]: # filter out specific outputs
# e.g. we only want code 2 (oscillation)

input_file_name_filter = output_file_name # use the output file name for DYVIPAC
filter_number = '2'
have_header = True
output_file_name_filter = f'{input_file_name_filter[:-4]}_filter_{filter_number}.txt'

read_file = open(input_file_name_filter).readlines()

with open(output_file_name_filter, 'w') as output_file:
    if have_header:
        output_file.write(read_file[0])
    for line in read_file:
        if line[0] == filter_number:
            output_file.write(line)
```

Filter out bistability

```
In [3]: # filter out specific outputs
# e.g. we only want code 6 (bistability)

input_file_name_filter = output_file_name # use the output file name for DYVIPAC
filter_number = '6'
have_header = True
output_file_name_filter = f'{input_file_name_filter[:-4]}_filter_{filter_number}.txt'

read_file = open(input_file_name_filter).readlines()

with open(output_file_name_filter, 'w') as output_file:
    if have_header:
        output_file.write(read_file[0])
    for line in read_file:
        if line[0] == filter_number:
            output_file.write(line)
```

The following output files shown below should be produced and can be found under the DYVIPAC repository.

- [output_example2_2_0.txt](#)
- [output_example2_2_0_filter_2.txt](#)
- [output_example2_2_0_filter_6.txt](#)

To visualize the newly filtered data, we can use the in-built visualization system for preliminary visualization. Note that it is very computationally intensive to render 10,000 parallel coordinate lines, and therefore, we will visualize only the filtered output files. To do this, we set the `plotting_file_name` to “`output_example2_2_0_filter_2.txt`” and `initial_params_file` to “`models\Mixed\mixed_p2.txt`.” Since we are using different parameter ranges, we set `normalise` to True.

Visualisation Data Input

```
In [32]: plotting_file_name = 'output_example2_2_0_filter_2.txt'
initial_params_file = 'models\Mixed\mixed_p2.txt'
normalise = True
figure_size = (12, 6)
image_quality_dpi = 80
```

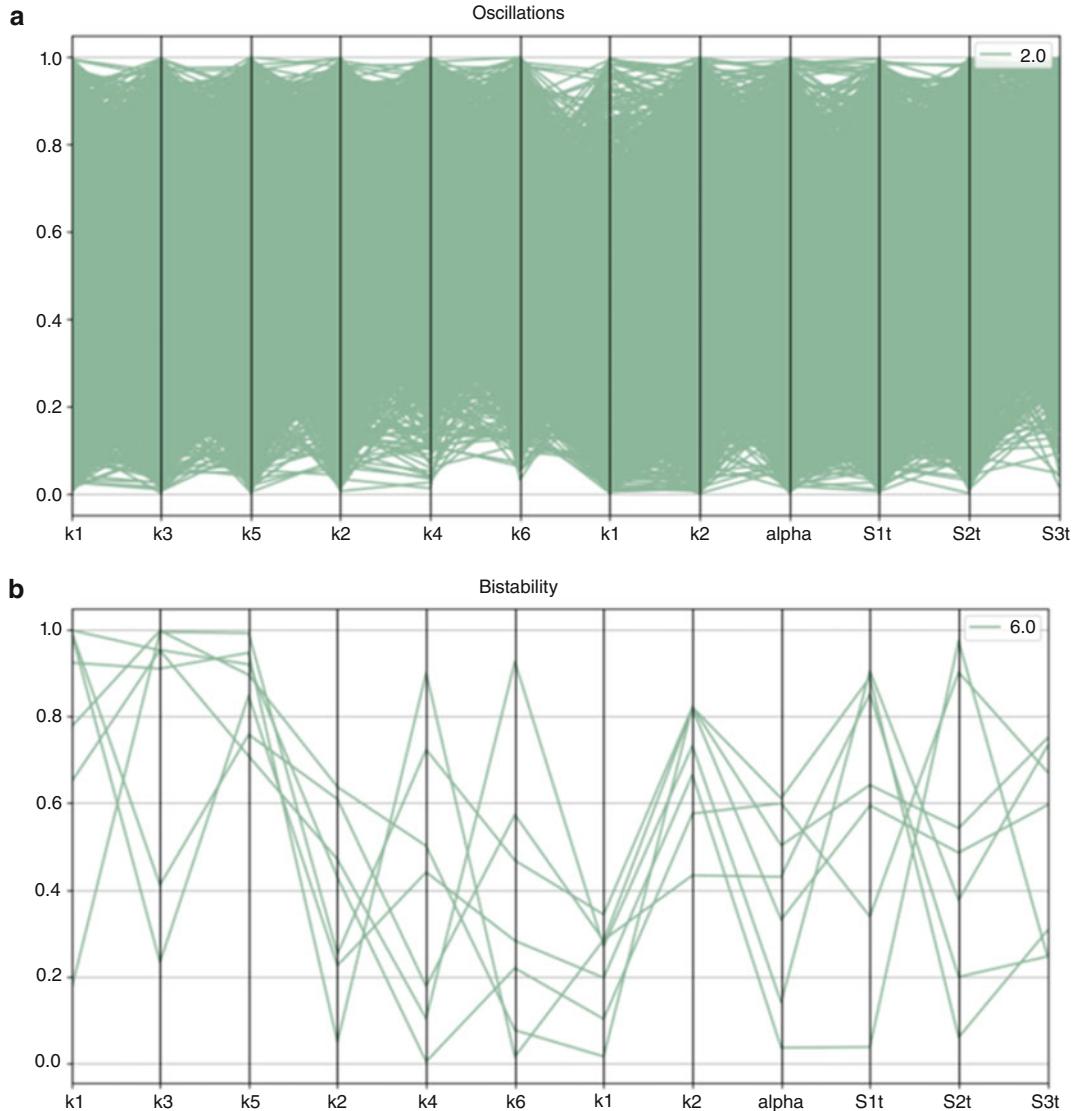


Fig. 11 Parallel coordinate plot displaying the results using pandas. (a) PC plot displaying only the oscillation parameter sets returned from sampling 12 parameters simultaneously. (b) PC plot displaying only the bistability parameter sets returned from sampling 12 parameters simultaneously

With all the visualization configurations complete, run the “Visualisation Data Input” code block, followed by the “Pandas Parallel Coordinate Plot” code block. Figure 11a shows the oscillation parallel coordinate plot generated by pandas. We can see that oscillation is a common pattern given the searched parameter space for the *mixed* model. This is as expected of systems with both positive and negative feedback mechanisms.

To visualize potential bistability behaviors, we will run the same procedure as above, but changing the `plotting_file_name` to “output_example2_2_0_filter_6.txt” instead. You should obtain a plot

as in Fig. 11b upon re-running the code blocks. We can see that bistability are possible but rather rare occurrences in the *mixed* model, given the sampled ranges. Interestingly, in the plot, we see that the parameter “K1” appears to be relatively low, while “K2” is relatively high in the normalized ranges. This observation is highlighted by the power of parallel coordinate visualization provided by the pyDYVIPAC workflow. You can also import the output files (and the filtered output files) to XDAT to generate better looking figures. See Note 5 for further discussion on the build-in vs. XDAT visualization regarding filtered results.

6 Summary

Here we have demonstrated pyDYVIPAC as a multidimensional analysis and visualization framework for ODE models of biochemical networks using parallel coordinate plots. In an integrated environment powered by Python 3 and Jupyter Notebook, pyDIVIPAC takes in three types of input files: the SBML-based .xml model definition file, parameter sample range file, and the input constraints file. Then, pyDIVIPAC uses the Monte Carlo method to extensively sample the specified parameter hyperspace while performing local stability analysis to map the observed network dynamic behaviors to the sampled parameter values. Finally, with high-dimensional visualization of the output, pyDYVIPAC provides a holistic understanding of how network dynamics can arise through unique patterns formed by a multitude of model parameters. This holistic parameter-to-dynamics relationship can in turn inform the behavior of real complex biological systems represented by the model, or alternatively, optimize the design of synthetic biological circuits.

7 Notes

1. *Setting sampling ranges.* The parameter file is configured in the fashion of [parameter name] [minimum] [maximum]. In pyDYVIPAC, there are really no restrictions on the min and max bounds. Ideally, the sampling range for each chosen parameter should be informed by biological evidence, i.e., spanning biologically plausible values. However, in the absence of such knowledge, the ranges can be relaxed from biological constraints to take any reasonable ranges for the purposes of exploration.

It is often a good idea to set a small, well-defined range of the parameters which the users would like to explore first. This is because it would take more sampling, and so computing time, to fully explore a large parameter space than a small

one. Setting small ranges first reminds the users to not expend unnecessary computational power on less useful parameter spaces. The user should also be mindful when setting certain parameters to 0; it could potentially break the model and produce invalid results (e.g., a parameter is set to 0, but it is the denominator of a fraction in a reaction). In this case, one should set the min bound to be a very small, non-zero number. In other situations, where the aim is to identify as many dynamics as possible for a given network, setting wide sampling ranges from the start would be useful, before narrowing them down once specific dynamics of interest are identified. Overall, there is no fixed rules to setting sampling ranges, and the modellers should be flexible and exploring different options depending on their aims.

2. *Selection of parameters to explore.* In pyDYVIPAC, the number of parameters (dimensions) that could be simultaneously explored is limited only by the size of the model. However, a more practical limitation is computing time since exploring multiple parameters at the same time incur the same computational cost as requiring more sampling. One can start with a smaller number of sampling parameters and gradually increase this number to see if the patterns identified at lower dimensions still hold true or break in higher dimensions.
3. *Number of samples.* In principle, a larger number of sets is always better as this ensures more exhaustive coverage of the hyper parameter space. As with the number of dimensions, the number of sampling sets is mainly constrained by computing time. It is always recommended to start with a small number of samples first. This is because for some models, in particular large ones, it is much more computationally expensive to simulate and can take a long time even on parallelization. The computational load versus number of samples is usually linear; therefore, it is always better to use a small number of samples first to tease out the expected computational time for larger sample sizes.
4. *Note on model reduction.* If a model is not minimal, then the Jacobian matrix will be singular, and the stability analysis cannot be performed. Thus, if the model contains conservation laws, these have to be specified in the constraint file, so pyDYVIPAC can utilize these to reduce the model first to be in minimal form. Moreover, the use of conservation laws requires one to be careful when initializing the species.

For example, if a conservation law, e.g., $S1 + S2 + S3 = Stot$ exists, then one may model only $S1$ and $S2$ explicitly and substitute $S3$ with $Stot - S1 - S2$. This means that a correct

initialization of the species will be S1 in the range [0, Stot] and S2 in the range [0, Stot-S1]. So in order to tell pyDYVIPAC to use this constraint, the constrain file should contain the following:

```
s1 + s2 <= stot
```

Alternatively, one can use the full model (no substitution) and pyDYVIPAC will reduce it automatically. In this case, the conservation laws should be specified in the constrain file:

```
s1 + s2 + s3 = stot
```

5. *Visualization.* Although XDAT is a great parallel coordinates visualization program, the in-built visualization module comes with a unique advantage in that it is able to normalize the data using the input parameter file's minimum and maximum range of a given parameter automatically. This sometimes leads to more appropriate visualizations, especially in filtered outputs.

For instance, suppose you sampled a parameter k in a range between 0 and 100 and then filtered out all the parameter sets that have potential oscillatory behaviors (output code 2). The filtered parameter values has a maximum of 5 and minimum of 1 for k . Then, displaying the filtered output parameter sets using the output file's normalized value for x would only show a range between 1 and 5. Because the original sampled range for k is between 0 and 100, a normalized visualization of k in the range of 1 to 5 is not only less useful but also potentially misleading.

The above problem occurs for XDAT, since it does not have the sampling information stored in the input parameters file. Therefore, XDAT is more appropriate for outputs that have not been filtered beforehand. Conversely, in-built visualizations are more appropriate for filtered outputs.

6. pyDYVIPAC was tested successfully on Windows 10–11 and MacOS. Since its primary dependency is libRoadRunner, which can run on Linux, pyDYVIPAC should also not suffer any dependency problems running on Linux. We plan to further develop pyDYVIPAC in containerized environments to allow for better compatibility between different operating systems.

References

1. Kitano H (2004) Biological robustness. *Nat Rev Genet* 5:826–837
2. Nguyen LK, Kholodenko BN (2016) Feedback regulation in cell signalling: lessons for cancer therapeutics. *Semin Cell Dev Biol* 50:85–94
3. Ferrell JE Jr, Tsai TY-C, Yang Q (2011) Modeling the cell cycle: why do certain circuits oscillate? *Cell* 144:874–885
4. Bell-Pedersen D, Cassone VM, Earnest DJ et al (2005) Circadian rhythms from multiple oscillators: lessons from diverse organisms. *Nat Rev Genet* 6:544–556
5. Xiong W, Ferrell JE Jr (2003) A positive-feedback-based bistable “memory module” that governs a cell fate decision. *Nature* 426: 460–465
6. Rata S, Suarez Peredo Rodriguez MF, Joseph S et al (2018) Two interlinked bistable switches govern mitotic control in mammalian cells. *Curr Biol* 28:3824–3832.e6
7. Nguyen LK, Dobrzański M, Fey D, Kholodenko BN (2014) Polyubiquitin chain assembly and organization determine the dynamics of protein activation and degradation. *Front Physiol* 5:4
8. Byrne KM, Monsefi N, Dawson JC et al (2016) Bistability in the Rac1, PAK, and RhoA signalling network drives Actin Cytoskeleton dynamics and cell motility switches. *Cell Syst* 2:38–48
9. Nguyen LK, Kholodenko BN, von Kriegsheim A (2018) Rac1 and RhoA: networks, loops and bistability. *Small GTPases* 9:316–321
10. von Kriegsheim A, Nguyen LK (2018) Uncovering Bistability in the Rac1/RhoA Signaling network through integrating computational modeling and experimentation. In: Rivero F (ed) *Rho GTPases: methods and protocols*. Springer, New York, pp 21–36
11. Varusai TM, Nguyen LK (2018) Dynamic modelling of the mTOR signalling network reveals complex emergent behaviours conferred by DEPTOR. *Sci Rep* 8:643
12. Nguyen LK, Muñoz-García J, Maccario H et al (2011) Switches, excitable responses and oscillations in the Ring1B/Bmi1 ubiquitination system. *PLoS Comput Biol* 7:e1002317
13. Ermentrout B (2007) Computational systems. *Neurobiology* 2:1399
14. Dhooge A, Govaerts W, Kuznetsov YA (2003) MATCONT: a MATLAB package for numerical bifurcation analysis of ODEs. *ACM Trans Math Softw* 29:141–164
15. Tyson JJ, Novak B (2020) A dynamical paradigm for molecular cell biology. *Trends Cell Biol* 30:504–515
16. Nguyen LK, Zhao Q, Varusai TM, Kholodenko BN (2014) Ubiquitin chain specific auto-ubiquitination triggers sustained oscillation, bistable switches and excitable firing. *IET Syst Biol* 8:282–292
17. Nguyen LK, Degasperis A, Cotter P, Kholodenko BN (2015) DYVIPAC: an integrated analysis and visualisation framework to probe multi-dimensional biological networks. *Sci Rep* 5:12569
18. Nguyen LK (2012) Regulation of oscillation dynamics in biochemical systems with dual negative feedback loops. *J R Soc Interface* 9: 1998–2010
19. Nguyen LK, Kulasiri D (2009) On the functional diversity of dynamical behaviour in genetic and metabolic feedback systems. *BMC Syst Biol* 3:51



Chapter 3

A Practical Guide for the Efficient Formulation and Calibration of Large, Energy- and Rule-Based Models of Cellular Signal Transduction

Fabian Fröhlich

Abstract

Aberrant signal transduction leads to complex diseases such as cancer. To rationally design treatment strategies with small molecule inhibitors, computational models have to be employed. Energy- and rule-based models allow the construction of mechanistic ordinary differential equation models based on structural insights. The detailed, energy-based description often generates large models, which are difficult to calibrate on experimental data. In this chapter, we provide a detailed, interactive protocol for the programmatic formulation and calibration of such large, energy- and rule-based models of cellular signal transduction based on an example model describing the action of RAF inhibitors on MAPK signaling. An interactive version of this chapter is available as Jupyter Notebook at github.com/FFroehlich/energy_modeling_chapter.

Key words Mechanistic modeling, Model calibration, MAPK signaling, Thermodynamic models, Allosteric interactions

1 Introduction

Cells use signal transduction to respond to extracellular cues, controlling a variety of different cellular processes such as proliferation, development, and environmental adaption [1]. Signal transduction is initiated by the binding of extracellular ligands to transmembrane receptors [2] and then propagated through phosphorylation cascades [3, 4], ultimately controlling gene expression through modulation of transcription factor activity. Mutations and changes in protein expression levels can dysregulate signal transduction leading to a variety of different diseases such as cancer [5]. Small molecule inhibitors can be used to target specific proteins that participate in aberrant signal transduction [6]. However, feedback mechanisms and cross-talk between signaling cascades, which are ubiquitous to cellular signal transduction, lead to complex, non-linear responses to drug inhibition [7], making it difficult to

anticipate the phenotypic response or resistance to these inhibitors. Accordingly, computational models that describe signal transduction, and the effect of drug perturbations on it, are required for the rational selection and dosage of targeted inhibitors [8]. Mechanistic computational models of cellular signal transduction translate biological knowledge into executable mathematical formulations [9]. To this end, ordinary differential equation (ODE) models are a popular modeling approach, as they can describe temporal dynamics and feedback mechanisms [10]. Moreover, state variables and parameters correspond to biophysical entities, facilitating a direct interpretation of model simulations and predictions.

Formulation of such models is often a long, labor-intensive process, especially for large models. Instead of directly writing model equations, a tedious and error-prone process, it is advisable to use some kind of abstraction when writing models. For example, the Systems Biology Markup Language (SBML) [11] encodes models using concepts such as reactions and events, which can then be translated into differential equation models. Rule-based formats such as Kappa [12] or BioNetGen [13] employ even higher levels of abstractions and describe model equations in terms of rules, which are generalizations of reactions [14]. While such abstractions can limit what models can be expressed, they also permit a more compact model description. A recent extension to rule-based models are energy rule-based models [15, 16], which provide an additional layer of abstraction by enabling the specification of reaction rates in terms of energies. Conceptually, energies permit the theoretically sound formulation of reaction rates based on transition theory and provide a rigorous way of specifying context-dependent reaction rates based on structural insights [17, 18]. Practically, energy-based models enforce thermodynamic constraints such as the Wegscheider–Lewis cycle conditions [19], which ensure the conservation of energy in oligomerization processes. This is particularly relevant for the specification of context-dependent inhibitor affinities. For example, many RAF inhibitors have different affinities for RAF monomers and dimers, which causes for the clinically observed phenomenon of “paradoxical activation” in which signaling in RAF mutant cancer cells is inhibited, but signaling in RAS mutant cancer cells is amplified [20–22].

Calibration of large models can also be challenging. During model calibration, parameter values are tuned such that model simulations agree with experimental data, which improves the predictive performance of models. Agreement between model simulations and experiments can be quantified using likelihood-based objective functions, in which case model calibration can be implemented through minimization of the respective objective function. For larger models, gradient-based optimization methods perform well [23]. However, for non-linear models, analytical solutions to model equations are rarely available, which means that numerical

integration is necessary for evaluation of the objective function and its derivatives. Accordingly, gradient-based methods not only require integration of model equations, but also sensitivity equations, which quantify the dependence of simulations on model parameters. For large models, adjoint sensitivity analysis is an effective way of computing these sensitivities, but also comes with its drawbacks such as not permitting the computation of cheap approximations to the second-order derivative [24].

Model formulation and calibration for large models can be a daunting task [25], even when using specialized software tools. Input formats such as SBML or PEtab [26] permit the standardized problem formulation, which allows the automation of the majority of steps. However, manual interventions are still occasionally necessary to achieve optimal performance. These manual interventions often require substantial practical experience as well as theoretical background knowledge, which creates a high barrier to entry and means that successful development and application of large models is rarely possible without guidance from experts. To rectify this situation, we here provide a detailed step-by-step protocol for the calibration of a large-scale model introduced in [27] and [28]. In contrast to previous protocols for ODE model calibrations [29], this protocol includes a description of energy-based model formulation, focuses on large-scale models of intracellular signaling, and includes an interactive Jupyter Notebook that allows reproduction of all results. As in other protocols, theoretical background is provided when appropriate. For additional analysis steps including identifiability, uncertainty analysis, or model selection, users should consult previously published reviews and protocols. This protocol uses the Python toolboxes PySB [30], PEtab [26], AMICI [31], pyPESTO, and fides [32] and is accompanied by a Jupyter Notebook that permits the interactive reproduction of results.

2 Mathematical Problem Formulation

For ODE models, the temporal evolution of abundances of n_x different molecular species x_i is determined by the vector field \mathbf{f} and the initial conditions \mathbf{x}_0 :

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \gamma), \quad \mathbf{x}(t_0) = \mathbf{x}_0(\gamma). \quad (1)$$

Both \mathbf{f} and \mathbf{x}_0 may depend on dynamic parameters γ , which may include kinetic parameters such as catalytic or binding rates or condition parameters such as ligand or drug concentrations.

Calibration is performed by comparing model simulations, i.e., solutions \mathbf{x} that satisfy (1), to experimental data $\bar{\mathbf{y}}$. As direct measurement of \mathbf{x} is usually experimentally not possible, observables

$$\mathbf{y}(t, \boldsymbol{\gamma}, \boldsymbol{\xi}) = \mathbf{g}(\mathbf{x}(t, \boldsymbol{\gamma}), \boldsymbol{\xi}) \quad (2)$$

are introduced, which may depend on abundances \mathbf{x} as well as observable parameters $\boldsymbol{\xi}$ such as scaling parameters or offsets. The agreement between model observables \mathbf{y} and experimental measurements $\bar{\mathbf{y}}$ can be quantified according to the assumed noise model. A common assumption is that the measurement noise for n_y observables y_i is additive and independent, normally distributed:

$$\bar{y}_i = y_i(t, \boldsymbol{\gamma}, \boldsymbol{\xi}) + \varepsilon_i \quad \varepsilon_i \stackrel{iid}{\sim} \mathcal{N}(0, \sigma_i^2(\boldsymbol{\psi})), \quad (3)$$

where $\boldsymbol{\psi}$ are noise parameters that encode the standard deviations $\boldsymbol{\sigma}$. Using this noise model, the negative log-likelihood

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{2} \sum_{i=k}^{n_k} \sum_{(i,j) \in T_k} \log(2\pi\sigma_i^2(\Psi_{ijk}(\boldsymbol{\theta}))) \\ &\quad + \left(\frac{\bar{y}_{ijk} - y_i(t_j, \Gamma_k(\boldsymbol{\theta}), \Xi_{ijk}(\boldsymbol{\theta}))}{\sigma_i(\Psi_{ijk}(\boldsymbol{\theta}))} \right)^2 \end{aligned} \quad (4)$$

statistically rigorously quantifies the agreement between experimental data and model simulations, where $\boldsymbol{\theta}$ are the free, unknown parameters, T_k is the set of time points t_j and observable y_i index combinations that were measured in the k -th out of n_k experimental conditions, Ψ_{ijk} is a function that maps free parameters $\boldsymbol{\theta}$ to observable, time point, and condition-specific noise parameters $\boldsymbol{\psi}$, Ξ_{ijk} is a function that maps free parameters $\boldsymbol{\theta}$ to observable, time point, and condition-specific observable parameters $\boldsymbol{\xi}$, and Γ_k is a function that maps free parameters $\boldsymbol{\theta}$ to condition-specific dynamic parameters $\boldsymbol{\gamma}$. As the logarithm is a strictly monotonously increasing function, the minimization of $J(\boldsymbol{\theta})$ is equivalent to the maximization of the likelihood. Therefore, the corresponding minimization problem

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta} \in \Theta} J(\boldsymbol{\theta}) \quad (5)$$

will infer the maximum likelihood estimate $\boldsymbol{\theta}^*$ for free parameters $\boldsymbol{\theta}$, where the search domain $\Theta \subset \mathbb{R}^{n_\theta}$ restricts the $\boldsymbol{\theta}$ to values that are biologically plausible and are less subject to numerical integration failures. For most problems, Θ is chosen as tensor product of scalar search domains $(\mathcal{L}_l, \mathcal{U}_l)$ with lower and upper boundaries $\mathcal{L}_l < \mathcal{U}_l$ and $\mathcal{L}_l, \mathcal{U}_l \in \mathbb{R} \cup \{-\infty, \infty\}$ for every parameter θ_l .

3 Formulating a Thermodynamic Model of RAF Inhibition in PySB

In the following, we will specify the model, which in turn defines the right-hand side \mathbf{f} and the initial conditions \mathbf{x}_0 of the model equations. We will specify the model using the rule-based modeling framework PySB, which instead of direct specification of \mathbf{f} , programmatically encodes model structure in a modular, object-

oriented way, ensuring reusability, composability, and extendability of models [30]. To start model specification, we instantiate a PySB Model with name thermo Raf.

```
[1]: from pysb import Model
model = Model('thermo Raf');
```

3.1 Protein Species

As first step of model construction, we have to define all molecules that we want to account for. In PySB, this is done by instantiating the Monomer class. Each monomer is initialized by a unique identifier, a list of sites as well as, optionally, a list of possible states for each site. Sites typically correspond to binding domains or residues that can be mutated or modified post-translationally, where the site states encode the different all possible mutations or post-translational modifications for each site. All sites, independent of whether possible states were specified or not, can form a bond with other sites, permitting the description of macromolecular complexes.

Here, we will define monomers BRAF and CRAF that both feature the interaction sites RBD (interaction domain with RAS), raf (dimerization domain), and rafi (inhibitor binding site). For BRAF, we also add a mutation site for the 600th amino acid, which can take values V (valine, wildtype) or E (glutamic acid, oncogenic mutation).

```
[2]: from pysb import Monomer
Monomer('BRAF', ['AA600', 'RBD', 'raf', 'rafi'], {'AA600': ['E', 'V']});
Monomer('CRAF', ['RBD', 'raf', 'rafi']);
```

Next, we will specify initial concentrations. For this model, the concentration of all molecular species will be in μM . However, we will use protein counts per cell derived from absolute proteomics to inform initial conditions. Accordingly, parameters that define initial concentrations have to be transformed from molecule per cell to μM , which is achieved by dividing them by cell volume (here assumed to be $1\text{pL}=10^{-12}\text{L}$) and the Avogadro constant (6.02210^{23} molecules) and multiplying with 10^6 to account for the unit prefix μ .

In the following code, we first introduce two PySB Expressions that specify the Avogadro constant and cell volume. As with most model components, PySB automatically creates workspace variables with the respective identifier as variable names, simplifying the programmatic reference to components. Next, we introduce two dynamic parameters (γ , see (1)) as PySB Parameters BRAF_0 and CRAF_0 that define initial abundances as molecules per cell and convert these abundances into concentrations in μM using the expressions initBRAF and initCRAF. These expressions are then used to define the initial abundances for the two monomer species CRAF and BRAF using the PySB Initial class, which specifies x_0 for

the respective molecular species x_i . In PySB, a molecular species is defined by a pattern, which is created by invoking the respective monomer with the state of each site as keyword argument. For initial conditions, the respective patterns have to be explicit, i.e., the states of all sites have to be specified. For the initial conditions, we specify that all sites are unbound (denoted by `None`), with the exception of the AA600 site. For the AA600 site, we have to pick one of the previously specified states, which we do by specifying the oncogenic variant denoted by `E`. For all molecular species without an explicit instantiation of `Initial`, the respective entry in \mathbf{x}_0 is set to 0.

```
[3]: from pysb import Parameter, Expression, Initial
import sympy as sp
# define Avogadro constant and volume as hardcoded expressions
Expression('N_Avogadro', sp.Float(6.02214076e+23))
Expression('volume', sp.Float(1e-12))
# define initial abundance parameters
Parameter('BRAF_0')
Parameter('CRAF_0')
# convert initial abundances to concentrations
Expression('initBRAF', 1000000.0 * BRAF_0 / (N_Avogadro * volume))
Expression('initCRAF', 1000000.0 * CRAF_0 / (N_Avogadro * volume))
# define initial molecular species
Initial(CRAF(RBD='None', raf='None', rafi='None'), initCRAF)
Initial(BRAF(RBD='None', raf='None', rafi='None'), initBRAF);
```

3.2 Protein Interactions

For rule-based models, all dynamic interactions are specified as rules. PySB [Rules](#) are generalizations of biochemical reactions, and their action is defined by a reactant pattern and a product pattern. When applied to a molecular species, the action of the rule, i.e., the respective biochemical reaction, is implemented by applying the difference between the reactant and product patterns to a match of the reactant pattern in a molecular species. In contrast to initials, the reactant pattern and product pattern do not have to be explicit, and a single rule can define multiple different biochemical reactions, depending on how often and in how many molecular species the reactant pattern occurs.

Non-thermodynamic rules additionally require forward reaction rates k_f and, if applicable, reverse reaction rates k_r . In contrast, thermodynamic rules are specified in terms of activation energy E_a and the phenomenological constant ϕ , the convex combination parameter that encodes how much changes in free energy affect forward and reverse rates. For thermodynamic rules, the reaction rate also depends on the free energy difference between reactants and products of the reaction. The forward k_f and reverse k_r reaction rates are computed using the Arrhenius theory of reaction rates

$$k_f = \exp(-(E_a + \phi\Delta G)), \quad k_r = \exp(-(E_a + (\phi - 1)\Delta G))$$

with

$$\Delta G = - \sum_{r \in R} \Delta \Delta G_r + \sum_{p \in P} \Delta \Delta G_p$$

where E_a is the activation energy of the reaction, i.e., the free energy difference between the reactant state and the transition state, ΔG is the free energy difference between the reactant and the product states, and $\Delta \Delta G_r$ and $\Delta \Delta G_p$ are the free energy modifiers that apply to the reactants r in R and the products p in P , respectively. All energies are assumed to be normalized by the inverse of the product between temperature and Boltzmann constant $\frac{1}{RT}$. The $\Delta \Delta G_r$ and $\Delta \Delta G_p$ energies can be specified by using the PySB EnergyPattern class. When calculating reaction rates, energy BioNetGen (eBNG) not only accounts for EnergyPatterns that apply to the reactant and product patterns specified in the rule, but also those EnergyPatterns that specifically apply to reactant and product species. Thus, reactions generated by thermodynamic rules do not always have the same reaction rate. In contrast, all reactions generated by a single non-thermodynamic rule will have the same reaction rate, unless a local function is used for rate specification.

In the following, we will specify the dimerization of RAF molecules. For this purpose, we introduce three parameters: the activation energy for the binding reaction E_a (Ea suffix), the Gibb's free energy of the formed bond ΔG (dG suffix), and the thermodynamic balance parameter ϕ (phi suffix).

```
[4]: Parameter ( 'bind_RAF_RAF_Ea' )
Parameter ( 'bind_RAF_RAF_dG' )
Parameter ( 'bind_RAF_RAF_phi' ) ;
```

To enforce that the activation energy $E_a = -\log(k_f)$ encodes the forward reaction rate and $\Delta G = -\log(K) = -\log\left(\frac{k_f}{k_r}\right)$ encodes the affinity constant K , we will pass a specially crafted custom activation energy $E_{a0} = -\phi \Delta G - E_a$, for which we create a custom expression. To then construct homo- and heterodimerization rules for all RAF paralogs, we use `itertools` to loop over all combinations of BRAF and CRAF, implementing equal affinities for all homo- and heterodimers. For every rule, we pass the custom energy Ea_0 and then specify an EnergyPattern for the single product pattern (pp) with $\Delta \Delta G_p = \Delta G$.

```
[5]: from pysb import Rule, EnergyPattern
import itertools as itt
Expression ( 'Ea0_bind_RAF_RAF' ,
            - bind_RAF_RAF_phi * bind_RAF_RAF_dG - bind_RAF_RAF_Ea ) ;
for RAF1, RAF2 in itt.combinations_with_replacement ( [ BRAF, CRAF ] , 2 ) :
    pp = RAF1 ( raf = 1 ) % RAF2 ( raf = 1 )
    Rule ( f' { RAF1 . name } _and_ { RAF2 . name } _bind_and_dissociate ' ,
          RAF1 ( raf = None ) + RAF2 ( raf = None ) | pp ,
          bind_RAF_RAF_phi , Ea0_bind_RAF_RAF , energy = True )
    EnergyPattern ( f' ep_bind_ { RAF1 . name } _ { RAF2 . name } ' , pp , bind_RAF_RAF_dG ) ;
```

3.3 RAF Inhibitor

Next, we introduce a RAF inhibitor RAFi. We assume that the inhibitor is added to the cell medium at some point and quickly diffuses in and out of cells. As extracellular space is much bigger than the volume of a cell, we can assume an infinite reservoir of molecules in the extracellular compartment and assume that the

intracellular inhibitors concentration will be unaffected by intracellular reactions. Accordingly, we specify the respective initial as fixed, which means that the respective molecular species can participate in reactions, but its concentration will remain constant.

```
[6]: Monomer ( 'RAFi' , [ 'raf' ] )
Initial ( RAFi ( raf = None ) , Parameter ( 'RAFi_0' , 0.0 ) , fixed = True );
```

Next, we define binding reactions for RAFi with both BRAF and CRAF, again implementing the same affinities and activation energies for both rules.

```
[7]: Parameter ( 'bind_RAFi_RAFA_Ea' )
Parameter ( 'bind_RAFi_RAFA_dG' )
Parameter ( 'bind_RAFi_RAFA_phi' )
Expression ( 'Ea0_bind_RAFi_RAFA' ,
            - bind_RAFi_RAFA_phi * bind_RAFi_RAFA_dG - bind_RAFi_RAFA_Ea )
for RAF in [ BRAF , CRAF ] :
    Rule ( f 'RAFI_and_(RAF_name)_bind_and_dissociate' ,
           RAFi ( raf = None ) + RAF ( raf = None ) | RAFi ( raf = 1 ) % RAF ( rafi = 1 ) ,
           bind_RAFi_RAFA_phi , Ea0_bind_RAFi_RAFA , energy = True )
EnergyPattern ( f 'ep_bind_(RAF_name)_RAFI' ,
               RAFi ( raf = 1 ) % RAF ( rafi = 1 ) , bind_RAFi_RAFA_dG );
```

3.4 Paradoxical Activation

RAF inhibitors inhibit signaling for BRAF mutant cancer cells but promote signaling for RAS mutant cancer cells. At the structural level, this can be rationalized by assuming that RAF inhibitors have higher affinity toward drug-unbound RAF dimers and lower affinity toward drug-bound RAF dimers. The symmetry conveyed by energy conservation of molecular binding reactions implies that RAF inhibitors promote dimerization at low to medium concentrations and incompletely inhibit signaling even at high concentrations, as they have lower affinity to the second protomer in a RAF dimer. As MAPK signaling for RAS mutant cells is mediated by RAF dimers, respective signaling is amplified, leading to increased growth. In the thermodynamic models, this can be implemented by specifying additional EnergyPatterns that control the Gibbs free energy of $\text{RAF}_2 - \text{RAF}_i$ trimers. Note that we do not specify how these trimers are formed, so the change in energy will equally apply to the rates of all reactions that either consume or produce these trimers. In this example, an decrease in energy would equally increase RAF dimerization when exactly one protomer is inhibitor-bound and inhibitor binding to the first protomer in a dimer, thereby implementing the previously described symmetry.

```
[8]: Parameter ( 'ep_RAFA_RAFA_ddG' )
for RAF1 , RAF2 in itt.product ( [ BRAF , CRAF ] , repeat = 2 ) :
    EnergyPattern (
        f 'ep_(RAF1.name)_({RAF2.name})_mod_RAFA_single' ,
        RAF1 ( raf = 1 , rafi = None ) % RAF2 ( raf = 1 , rafi = 2 ) % RAFi ( raf = 2 ) ,
        ep_RAFA_RAFA_ddG );
```

To implement the lower affinity to the second protomer, we add an additional EnergyPattern that changes the Gibbs free energy of $\text{RAF}_2 - \text{RAF}_i$ tetramers.

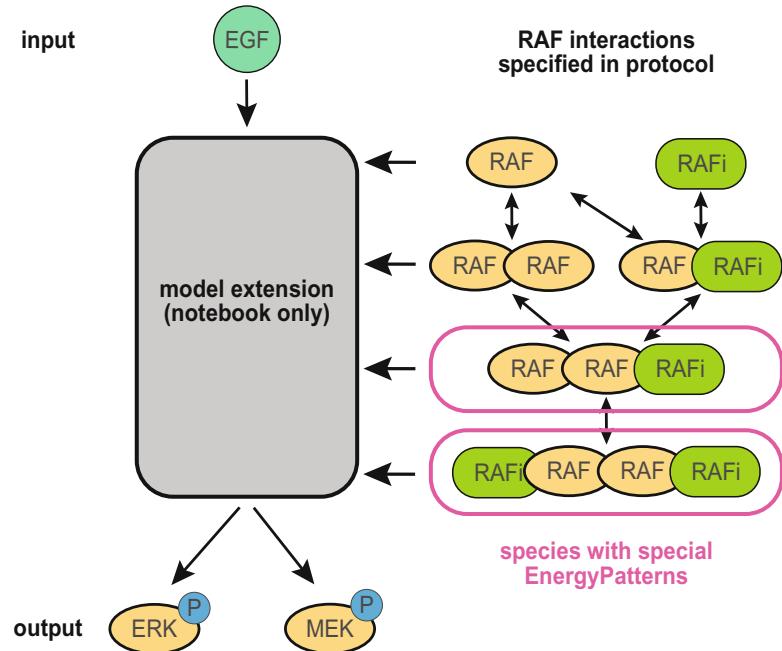


Fig. 1 Pre-calibration fit. Different rows correspond to different RAF inhibitors (vemurafenib and dabrafenib). Different columns correspond to different observables (pMEK and pERK). Colors indicate different experimental conditions (with and without EGF stimulation). Data is shown as dashed lines, and simulations as solid lines

```
[9]: Parameter ( 'ep_RAF_RAF_RAFi_RAFi_ddG' )
for RAF1 , RAF2 in itt . combinations_with_replacement ( [ BRAF , CRAF ] , r = 2 ) :
    EnergyPattern (
        f ' ep_ { RAF1 . name } - { RAF2 . name } _mod_RAFi_double ' ,
        RAF1 ( raf = 2 ) % RAF1 ( raf = 1 , rafi = 2 ) % RAF2 ( raf = 1 , rafi = 3 ) % RAFi ( raf = 3 ) ,
        ep_RAF_RAF_RAFi_RAFi_ddG ) ;
```

Here we note that these EnergyPatterns only specify that changes in affinity are permissible in the model. The nature of these changes has to be encoded in their values, where negative values lead to stabilization and positive values to destabilization of respective complexes.

This wraps up the description of RAF/RAF inhibitor interactions (Fig. 1), and we will now load the remainder of the model from a file. Briefly, the full model describes MEK and ERK phosphorylation downstream as well as EGF stimulatable EGFR signaling upstream of RAF signaling. Moreover, it incorporates negative feedback from ERK on both MAPK and EGFR signaling. A comprehensive description of this part of the model is available in [27] and [28]. However, parts of the model were substantially simplified account for the reduced set of experimental data considered in this protocol. Specifically, description of EGFR signaling is removed and replaced by a phenomenological description, MEK inhibitors were removed from the model, and two-step phosphorylation and

dephosphorylation reactions were replaced by linear one-step reactions.

```
[10]: from pysb import Observable, ANY
def extend_model():
    Content of this function is only included in the interactive notebook.
extend_model()
```

4 Importing Data in PEtab Format

With the model at hand, the next step for model calibration is specification of the objective function (4). We will specify the objective function using PEtab [26], which simplifies the definition of multiple experimental conditions. PEtab specification of a calibration problem consists of the tables describing model observables, experimental measurements, experimental conditions, and model parameters. Additionally, tables describing the visualization of simulations and data may also be included.

4.1 Observables

We will start the PEtab definition by specifying the model observables \mathbf{y} (see (2)). Observables define the the model quantities that were measured experimentally, here pMEK and pERK. First, we add respective observables to the model using PySB [Observables](#):

```
[11]: Observable('pMEK_obs', MEK(phospho='p'))
Observable('pERK_obs', ERK(phospho='p'))
```

These observables quantify all MEK (MAP2K1 + MAP2K2) and ERK (MAPK1 + MAPK3) molecules that are phosphorylated on the site phospho, which accounts for phosphorylation on S218/S222 or S222/S226 on MEK and T185/Y187 or T202/Y204 on ERK. These PySB [Observables](#) define linear sums of model species $\sum_i w_i x_i$, where w_i is an integer that quantifies how often the specified pattern matches the species x_i .

Next, we construct the PEtab [observables table](#). The model quantifies pMEK and pERK in concentrations, but measurements are noise corrupted, measured in fluorescence intensity, and also include background fluorescence. As the scaling between concentrations and intensity and the amount of background signal is unknown, we include scaling and offset parameters in the PEtab observable definition `petab.OBSERVABLE_FORMULA`, which specifies the observable function g_i (see (2)). To account for noise corruption, we specify a single noise parameter ψ_i as PEtab noise formula `petab.NOISE_FORMULA`, which corresponds to a Gaussian distribution (default when no `petab.NOISE_DISTRIBUTION` is specified), with the respective parameter as standard deviation σ_i (see (3)). The prefix `noiseParameter1` indicates that the value of the parameter will be provided in the respective column of the [measurement table](#), which permits the specification of time point and condition-specific noise levels via Ψ_{ijk} (see (4)). Here, the

specified observable formula g_i does not include any explicit PEtab noise parameters. However, the scale and offset parameters for each observable could be replaced by `observableParameter[0-9]+` placeholders, which would enable the designation of time point and condition-specific values in the [measurement table](#).

```
[12]: import petab
import pandas as pd
observables_table = pd.DataFrame([
    {
        petab.OBSERVABLE_ID : obs_id,
        petab.OBSERVABLE_FORMULA :
            f'{obs_id}_scale*{obs_id}_obs + {obs_id}_offset',
        petab.NOISE_FORMULA : f'noiseParameter1_{obs_id}',
    } for obs_id in ['pMEK', 'pERK']
]).set_index(petab.OBSERVABLE_ID)
```

4.2 Measurements and Conditions

As experimental measurements, we will load one of the datasets from [\[27\]](#).

```
[13]: import synapseclient as sc
pd.set_option('display.max_rows', 5)
pd.set_option('display.max_columns', 6)
syn = sc.Synapse()
syn.login(silent=True)
data_df = pd.read_csv(syn.get('syn22804081').path)
```

The data contains the dose response to multiple RAF and MEK inhibitors as well as respective combinations. However, to minimize computation time requirements, we will only consider data involving the two RAFis vemurafenib and dabrafenib. The model itself only contains generic RAFi parameters and species, which we can now map to specific inhibitors using experimental conditions. This permits the simultaneous estimation of inhibitor-specific kinetic rates in conjunction with all other model parameters. Such a multi-experiment setup can improve parameter identifiability.

In the following, we will simultaneously generate the [condition table](#), encoding the condition-specific mapping Γ_k (see (4)) of dynamic parameters γ (see (1)), and [measurement table](#), encoding measurements \bar{y}_{ijk} and, potentially, respective mappings for observable parameters ξ (see (2)) via Ξ_{ijk} (see (4)) and noise parameters ψ (see (3)) via Ψ_{ijk} (see (4)). The [condition table](#) describes the necessary information about experimental conditions, i.e., what drug was used and what concentrations were applied. The [measurement table](#) describes the measurement itself as well as additional information, such as under which experimental conditions the data was measured.

The data we loaded above is in a matrix format, but PEtab requires the data in a long format. Thus, additional processing is necessary. Briefly, the code below extracts all datapoints and sets what model observable `petab.OBSERVABLE_ID` they belong to, the value of the measurement `petab.MEASUREMENT`, the time of measurement `petab.TIME`, the noise parameter `petab.NOISE_PARAMETERS`, and information about the experimental condition.

For the EGF stimulation condition, it is important to consider the experimental setup, as cells are subjected to 24h of

pretreatment with drugs. We here assume that this is sufficient time for the system to reach a steady state. To describe such an experimental setup in PEtab, it is necessary to reference the respective experimental condition in the `condition table` by specifying the respective `petab.CONDITION_ID` in the `petab.PREEQUILIBRATION_CONDITION_ID` column. The condition for the actual experiment can then be specified in the `petab.SIMULATION_CONDITION_ID`.

To enable the visualization of data and fitting results, we here also specify the `petab.DATASET_ID`. This does not influence the actual fitting but serves as grouping identifier during visualization.

```
[14]: RAFis = [ 'Vemurafenib' , 'Dabrafenib' , 'PLX8394' , 'LY3009120' , 'AZ_628' ]
MEKis = [ 'Cobimetinib' , 'Trametinib' , 'Selumetinib' , 'Binimetinib' , 'PD0325901' ]
conditions = dict ( )
def format_petab ( row ) :
    suffixes = [ '' ] + [ f'_{idx}' for idx in range ( 1 , 10 ) ]
    row [ MEKis ] . any ( ) :
        return datapoints
    # loop over columns of the data matrix
    for suffix in suffixes :
        # extract datapoint
        datapoint = {
            petab . OBSERVABLE_ID : 'pMEK' if row . pMEK else 'pERK' ,
            petab . MEASUREMENT : row [ f' Mean { suffix }' ] ,
            petab . TIME : row [ 'Time_EGF' ] ,
            petab . NOISE_PARAMETERS : row [ f' Std { suffix }' ] ,
        }
        # extract condition information
        if ( row [ RAFis ] != 0 ) . sum ( ) != 1 or \
            ( ( row [ 'Vemurafenib' ] == 0 ) and ( row [ 'Dabrafenib' ] == 0 ) ) :
            continue
        # find first nonzero
        rafi = RAFis [ ( row [ RAFis ] != 0 ) . argmax ( ) ]
        # extract drug concentration
        drug_conc = row [ f' Concentration (uM) { suffix }' ] if row [ rafi ] == - 1 \
            else row [ rafi ]
        # condition id must be sanitized, must match '[-a-zA-Z]+[u_.]*$'
        drug_str = f' { rafi }_{drug_conc}' . replace ( ' ' , ' ' ) . replace ( '-' , ' ' )
        condition_str = drug_str + (
            f' _EGF_ { ( row [ "EGF" ] ) } ' if row [ 'EGF' ] > 0 else ''
        )
        condition = {
            petab . CONDITION_ID : condition_str , 'RAFI_0' : drug_conc ,
            'EGF_0' : row [ "EGF" ] , 'bind_RAFI_RAF_Ea' : f' bind_{ rafi }_RAF_Ea' ,
            'bind_RAFI_RAF_DG' : f' bind_{ rafi }_RAF_DG' ,
            'ep_RAF_RAF_RAFI_ddG' : f' ep_RAF_RAF_{ rafi }_ddG' ,
            'ep_RAF_RAF_RAFI_ddG' : f' ep_RAF_RAF_{ rafi }_ddG' ,
        }
        # set baseline for datapoint
        datapoint [ petab . PREEQUILIBRATION_CONDITION_ID ] = drug_str
        # set id for condition and datapoint
        datapoint [ petab . SIMULATION_CONDITION_ID ] = condition_str
        datapoint [ petab . DATASET_ID ] = ('EGF__', if row [ 'EGF' ] > 0 \
            else 'ctrl__') + rafi
        condition [ petab . CONDITION_ID ] = condition_str
        datapoints . append ( datapoint )
        if condition_str not in conditions :
            conditions [ condition_str ] = condition
    return datapoints
measurement_table = pd . DataFrame ( [
    d
    for ir , row in data_df . iterrows ( )
    for d in format_petab ( row )
])
condition_table = pd . DataFrame ( conditions . values ( ) ) . set_index ( petab . CONDITION_ID ) ;
```

In the code above, we extracted experimentally measured standard deviations, which can be noisy as they are often computed from a small number of biological or technical replicates. As the magnitude of standard deviations determines the importance of datapoints during calibration, outliers can lead to overemphasis or ignorance of individual datapoints, resulting in poor fits to the overall data. To avoid such issues, we use the same averaged standard deviations for all datapoints of each observable. For averaging, we compute the root mean square, as variances are additive, but standard deviations are not.

```
[16]: import numpy as np
for group, frame in measurement_table.groupby('petab.OBSERVABLE_ID'):
    measurement_table.loc[measurement_table['petab.OBSERVABLE_ID'] == group,
                          'petab.NOISE_PARAMETERS'] = \
        np.sqrt(frame['petab.NOISE_PARAMETERS'].apply(np.square).mean())
measurement_table;
```

4.3 Parameters

The last table necessary for model training is the [parameter table](#). This table describes all free parameters θ (`petab.PARAMETER_ID`), whether they have to be estimated (`petab.ESTIMATE`) as well as their scales (`petab.PARAMETER_SCALE`), nominal values (`petab.NOMINAL_VALUE`), boundary values \mathcal{L}_l (`petab.LOWER_BOUND`) and \mathcal{U}_l (`petab.UPPER_BOUND`), and information about priors.

Both parameter boundaries and priors can be used to restrict the search space to biologically plausible parameter values, or prevent numerical issues in regions of parameter space where numerical integration of model equations is challenging. Parameter boundaries set hard thresholds that cannot be exceeded, while parameter priors are soft constraints that also regularize the calibration. While priors are necessary for Bayesian analysis such as parameter sampling, they are not generally required for optimization-based approaches.

For most models, free parameters θ will primarily occur in the model specification as dynamic parameters γ . But, all of previously described tables may introduce new free parameters: noise parameters ψ in noise formulas σ or respective condition-specific mappings Ψ_{ijk} , observable parameters ξ in observable formulas ϱ or respective condition-specific mappings Ξ_{ijk} as well as the condition-specific dynamic parameter mappings Γ_k . Moreover, the respective tables, i.e., the respective mappings Ψ_{ijk} , Ξ_{ijk} and Γ_k , may set numerical values for the noise, observable, and dynamic parameters, rendering them non-free. As the parameter table must only describe free parameters θ , some processing is usually necessary to identify free and non-free parameters.

In the following, we will construct a list of `condition_pars`, which are either mapped to free, condition-specific parameters, such as the generic RAFi energy parameters, or set to non-free parameters with fixed numeric values, such as the `RAFi_0` initial concentration. To construct the list of free parameters, we identify all dynamic parameters in the model that are not contained in these `condition_pars` and add all newly introduced condition-specific parameters in the [condition table](#) as well as the four offset and scaling parameters we introduced in the [observables table](#).

```
[16]: import numpy as np
condition_pars = [
    par.name
    for par in model.parameters
    if par.name in condition_table.columns
]
free_parameters = [
    par.name
    for par in model.parameters
    if par.name not in condition_pars
] + [
    name
    for par in condition_pars
    for name in np.unique(condition_table[par])
    if isinstance(name, str)
] + [
    'pMEK_offset', 'pMEK_scale', 'pERK_offset', 'pERK_scale'
]
```

All of the parameters in the model follow a consistent naming scheme, which makes it easier to programmatically specify parameter boundaries, scales, and whether parameters are to be estimated. Multiple studies have demonstrated that parameter estimation is more efficient when estimating parameters on a logarithmic scale [23, 33]. Yet, the thermodynamic formulation we employed for model construction relies on an exponential dependency between energies and kinetic rates, which complements these practical insights with a theoretical foundation why a logarithmic scale is more natural. However, this also suggests that energetic parameters, in contrast to kinetic rates, should be estimated on a linear scale.

The upper and lower boundaries should ideally be set to biologically plausible values. However, most models employ simplified descriptions of the underlying biochemical processes, which means that model parameters may no longer have a one-to-one correspondence to true biochemical constants and plausible parameter ranges may be difficult to derive. Moreover, plausible parameter ranges may not be known for all parameters. Therefore, in many applications, parameter boundaries are initially based on educated guesses and then refined based on estimation results such that optima are contained in the specified ranges and integration failures are not too frequent. Therefore, the boundaries presented here are based on the values used in the original publication [27] and were then refined to work with the simplified model.

Similar to parameter boundaries, nominal values can be difficult to derive. However, the numerical values are only important for model calibration when respective parameters are not estimated. In this example, we do not estimate the initial concentrations and ϕ parameters and only set the nominal parameter values of those parameters to non-trivial values. The initial concentrations are set to approximate molecular counts as measured in the original publication, and ϕ parameters are all set to 1, meaning that free energy differences only affect the reverse rate.

```
[17]: lbs = {
    'Ea' : - 8 , 'dG' : - 8 , 'ddG' : - 8 , 'phi' : 0 , 'offset' : 5e-2 ,
    'scale' : 1e0 , 'eq' : 1e1 , 'O' : 1e3 , 'kcat' : 1e-2 , 'gexpslope' : 1e0 ,
    'kdeg' : 1e-3 , 'kM' : 1e-3 , 'kcatr' : 1e-5 , 'koff' : 1e-5 ,
}
ubs = {
    'Ea' : 8 , 'dG' : 8 , 'ddG' : 8 , 'phi' : 1 , 'offset' : 2e-1 ,
    'scale' : 1e6 , 'eq' : 1e5 , 'O' : 1e5 , 'kcat' : 1e5 , 'gexpslope' : 1e6 ,
    'kdeg' : 1e2 , 'kM' : 1e1 , 'kcatr' : 1e2 , 'koff' : 1e0 ,
}
initials = {
    'BRAF_O' : 1e3 , 'CRAF_O' : 1e4 , 'MEK_O' : 1e5 , 'ERK_O' : 1e5 , 'RAS_O' : 5e4 ,
}
parameter_table = pd.DataFrame([
    {
        'petab.PARAMETER_ID': par,
        'petab.PARAMETER_SCALE': petab.LOG10 if par.endswith(('_Ea', '_dG', '_ddG', '_phi')) else petab.LIN,
        'petab.LOWER_BOUND': lbs[par.split('_')[1]][-1],
        'petab.UPPER_BOUND': ubs[par.split('_')[1]][-1],
        'petab.NOMINAL_VALUE': initials[par] if par.endswith('_O') else 1.0,
        'petab.ESTIMATE': False if par.endswith('_phi', '_O') else True,
    } for par in free_parameters
]).set_index('petab.PARAMETER_ID');
```

4.4 SBML Export

PEtab expects a model in SBML format (an extension allowing other modeling formats is being developed at the time of writing), but we constructed the model in the PySB/BNGL format. Therefore, we need to export the model in SBML format.

```
[18]: import libsbml
from pysb.export import export
sbml_reader = libsbml.SBMLReader()
sbml_doc = sbml_reader.readSBMLFromString(export(model, 'sbml'))
sbml_model = sbml_doc.getModel()
```

In the SBML model, PySB [Observables](#) are exported as SBML [Parameters](#) and [AssignmentRule](#), with programmatically generated identifiers that do not match the PySB [Observable](#) names. As we referenced these observable names when specifying the PEtab observables, we have to change the SBML identifiers to match the PySB names. These changes have to be applied to the SBML [Parameter](#) as well as the SBML [AssignmentRule](#) that sets their value.

```
[19]: # rename pysb exported observables
sbml_model.getParameter('_obs1').setId('pMEK_obs')
sbml_model.getAssignmentRuleByVariable('_obs1').setVariable('pMEK_obs')
sbml_model.getParameter('_obs2').setId('pERK_obs')
sbml_model.getAssignmentRuleByVariable('_obs2').setVariable('pERK_obs');
```

As last step, we create a PEtab [Problem](#) using all of the previously constructed tables as well as the exported SBML model. As problem construction was a lengthy, error-prone process, we perform a static analysis of the PEtab problem using the PEtab library provided linter [petab.lint_problem](#). Here we suppress any SBML warnings, as they are only due to the implementation of the SBML export in PySB. These warnings cause the linter to terminate with “Not OK,” but as there are no further issues, we can proceed with calibration.

```
[20]: import petab
import logging
import sys
from petab.models import SbmlModel

petab_problem = petab.Problem(
    model = SbmlModel( sbml_model = sbml_model , sbml_reader = sbml_reader , sbml_document = sbml_doc ) ,
    measurement_df = measurement_table , condition_df = condition_table ,
    observable_df = observables_table , parameter_df = parameter_table
)
petab.sbm1.logger.setLevel( logging.ERROR )
petab.lint.logger.setLevel( logging.INFO )
petab.lint.logger.addHandler( logging.StreamHandler( sys.stdout ) )
petab.lint_problem( petab_problem ) ;

Checking model...
Checking measurement table...
Checking condition table...
Checking observable table...
Checking parameter table...
Not OK
```

5 Calibrating the Model in pyPESTO

Model calibration based on PEtab [26] format is supported by a growing number of tools, including COPASI [34] and data2dynamics [35]. Here we used the Python-based calibration tool pyPESTO for calibration. pyPESTO uses AMICI [31] for model import and simulation, constructs functions to evaluate the objective function $J(\theta)$ and its gradient $\nabla_{\theta}J(\theta)$, and provides an interface to a plethora of different optimizers.

To facilitate efficient evaluation of the objective function and its derivatives, we will first compile the model in AMICI. Model compilation should take about a minute on modern desktop machines but ensures that numerical solutions \mathbf{x} and their sensitivities $\nabla_{\theta}\mathbf{x}$, which are required for the evaluation of $\nabla_{\theta}J(\theta)$, are computed using the SUNDIALS solver suite [36]. AMICI compilation is triggered by calling the `compile_model` method of a pyPESTO `PetabImporter` instance. The compiled model will be automatically used by other methods of the same `PetabImporter` instance.

```
[21]: import pypesto
import pypesto.petab
import amici
importer = pypesto.petab.PetabImporter( petab_problem ,
                                         model_name = model.name ,
                                         validate_petab = False )
importer.compile_model( verbose = logging.ERROR )
```

Simulations in AMICI are highly customizable, and many steps, including everything that was specified in the PEtab problem, are automated by pyPESTO. Yet, not all options are set automatically, and some user input may be required. For example, the considered example includes pre-equilibrations (specified by `petab.PREEQUILIBRATION_CONDITION`), which are automatically handled by pyPESTO and AMICI. However, AMICI supports multiple different methods to compute steady states as well as respective sensitivities. By default, a combination of Newton's method and simulation is employed to approximate the steady state [37]. To find a steady state by Newton's method, the Newton–Raphson algorithm is applied to the root-finding problem

$$\{\mathbf{x} : \mathbf{f}(t, \mathbf{x}, \boldsymbol{\gamma}) = 0\}.$$

In contrast, the simulation-based approach numerically integrates (1) until \mathbf{f} is sufficiently small with respect to some norm. For both approaches, AMICI uses a convergence criteria based on a combination of absolute and relative tolerances, similar to error control during numerical integration:

$$\sum_i^{n_x} \frac{f_i(t, \mathbf{x}, \boldsymbol{\gamma})}{a + rx_i} < 1,$$

where a and r are absolute and relative tolerances that can be set via the `Solver` methods `setAbsoluteToleranceSteadyState` and `setRelativeToleranceSteadyState`, respectively. In both cases, sensitivities $\nabla_{\boldsymbol{\eta}} \mathbf{x}_{ss}$ of the steady state \mathbf{x}_{ss} can be computed using the Implicit Function Theorem (IFT) [38], which is the most efficient option when applicable:

$$\nabla_{\boldsymbol{\eta}} \mathbf{x}_{ss} = \nabla_x \mathbf{f}(t, \mathbf{x}_{ss}, \boldsymbol{\eta})^{-1} \nabla_{\boldsymbol{\eta}} \mathbf{f}(t, \mathbf{x}_{ss}, \boldsymbol{\eta}).$$

However, sensitivity computation via the IFT requires that the Jacobian $\nabla_x \mathbf{f}(t, \mathbf{x}_{ss}, \boldsymbol{\gamma})$ is not singular, which is, for example, not the case for models with conservation laws. More details about pre-equilibration in AMICI are available in the [online documentation](#).

As the model we constructed does not account for protein synthesis and degradation, total protein abundances are conserved quantities in the model, and the IFT is not applicable. Thus, we here employ a purely simulation-based steady-state search by setting the allowed newton steps to 0 and compute the steady-state sensitivities using forward sensitivity analysis by setting the steady-state sensitivity mode accordingly. To apply these options during optimization, we create AMICI `Solver` and `Model` instances from the `PetabImporter` and set respective options.

```
[22]: import amici
solver = importer.create_solver()
model = importer.create_model()
solver.setNewtonMaxSteps(0)
model.setSteadyStateSensitivityMode(amici.SteadyStateSensitivityMode.integrationOnly)
```

AMICI also supports different sensitivity analysis methods for dynamic simulations (specified by `petab.SIMULATION_CONDITION_ID`) [24]. For models with over 100 parameters, it usually makes sense to use adjoint sensitivity analysis [23]. However, for models that also require pre-equilibration, the computation time of the simulation and gradient evaluation may be dominated by pre-equilibration, and the choice of sensitivity method for dynamic simulations may have negligible impact on computation time. Accordingly, other considerations, such as the availability of approximate second-order information, should be taken into account when selecting the sensitivity method. In contrast to

forward sensitivity analysis, adjoint sensitivity analysis does not provide an approximation to the Hessian of the objective function [24], rendering some optimizers inapplicable and potentially reducing the convergence rate of those that remain applicable. Therefore, we here chose forward sensitivity analysis for dynamic simulations.

```
[23]: solver.setSensitivityMethod(amici.SensitivityMethod.forward)
```

Another set of crucial parameters to tune are integration step limits and tolerances. Lower tolerances generally improve accuracy of the objective function and its derivatives, which influences local convergence rate and success [32, 39]. However, lower tolerances also increases the computation time [40], so fewer optimization iterations could easily be offset by longer compute times per iteration. Moreover, both integration tolerances and step limits influence numerical integrability [40], which can lead to complications during optimization. Low step limits can promote numerical integration failure and, thereby, prevent optimization from exploring difficult to integrate parameter regions. However, if parameters from these regions yield good fits to the data, this can also deteriorate or bias calibration. Accordingly, setting adequate values for step limits and tolerances is integral for model calibration. Nevertheless, there are little to no theoretical or practical guidelines on how to choose adequate values. Ranges of 10^{-8} to 10^{-12} are usually adequate [40], but most of the time values are chosen and updated empirically. Here we chose tolerances of 10^{-12} and a step limit of 10^5 integration steps, which is rather conservative, but ensures adequate local convergence for the considered problem.

```
[24]: solver.setAbsoluteTolerance(1e-12)
solver.setRelativeTolerance(1e-12)
solver.setMaxSteps(int(1e5))
```

pyPESTO also supports extrapolation methods that iteratively update initial guesses for steady states by linearly extrapolating previously found steady states based on steady-state sensitivities. As this may lead to issues when combined with forward, simulation-based sensitivity analysis, we deactivate this option here. To set the option, we create a new [AmiciObjective](#) instance and pass the model solver and model instance we previously created to ensure that all options that we previously set are applied during evaluation of the objective.

```
[25]: obj = importer.create_objective(model=model, solver=solver)
obj.guess_steadystate = False
```

Lastly, we instantiate a pyPESTO [Problem](#) and attach the [AmiciObjective](#) we created.

```
[26]: pypesto_problem = importer.create_problem(objective=obj)
```

As gradient-based methods are local methods, they have guaranteed convergence to local minima (at least theoretically), and multiple repeated optimization runs initialized at different points in Θ have to be started to explore the objective function landscape [41]. Selecting an appropriate number of optimization runs can be challenging but should generally be adapted to the downstream analysis. For example, when model selection or uncertainty analysis using profile likelihoods will be performed, it is crucial to ensure that optimization reaches a local optimum and that the local optimum is, or at least yields a fit equivalent to, the global optimum. This can be achieved by running many optimization runs and allocating large computational budgets for each run. For large models, the total computational budget usually is insufficient to achieve these goals, and the number of runs and the computational budget have to be limited, with an unknown tradeoff between the two. To deal with this tradeoff, we will here use a two-staged approach where we first run many starts with a low computational budget of 10 min. This will just be enough for a handful of iterations, and we will then select the most promising runs and continue optimization with a larger computational budget. As optimizer, we employ fides [32], a high-performance trust-region optimizer permitting the direct specification of the computational budgets, which can be interfaced in pyPESTO using the [FidesOptimizer](#) class.

```
[27]: from pypesto.optimize import FidesOptimizer
import fides
import logging
optimizer = FidesOptimizer(options={fides.Options.MAXTIME: 600},
                           verbose=logging.ERROR)
```

Now we have everything at hand to run the first calibration stage. Here, we run 100 optimization runs with random starting points (uniformly sampled by default) using the pyPESTO [minimize](#) function. The calibration is parallelized using pyPESTOs [MultiThreadEngine](#) with 4 threads.

```
[28]: from pypesto.optimize import minimize
result = minimize(
    pypesto_problem, optimizer, n_starts=100,
    engine=pypesto.engine.MultiThreadEngine(4),
    progress_bar=False,
```

To assess whether the quality of the calibration, we will now use the PEtab visualization feature. The visualization instructions are specified in a [visualization table](#), where, in the case of line plots, every row corresponds to a single pair of connected lines for data and simulation. Here we make use of the `petab.DATASET_ID`

values that we specified in the [measurement table](#). In the following, we will generate separate plots for each observable and each inhibitor, where every plot shows the dose response in the EGF stimulated and unstimulated condition. The visualization table is then attached to the previously created `petab_problem`.

```
[29]: visualization_table = pd.DataFrame([
    {
        petab.PLOT_ID : f'{rafi}_{obs}' ,
        petab.PLOT_TYPE_SIMULATION : petab.LINE_PLOT ,
        petab.PLOT_TYPE_DATA : petab.MEAN_AND_SD ,
        petab.DATASET_ID : condition ,
        petab.X_VALUES : 'RAFI_0' ,
        petab.Y_VALUES : obs ,
        petab.X_SCALE : petab.LOG10 ,
        petab.X_LABEL : rafi ,
        petab.Y_LABEL : obs ,
        petab.LEGEND_ENTRY : condition . split ( '___' ) [ 0 ]
    }
    for rafi in RAFIs
    for obs in observables_table . index
    for condition in measurement_table [ petab.DATASET_ID ] . unique ( )
    if condition . split ( '___' ) [ 1 ] == rafi
])
petab_problem . visualization_df = visualization_table
```

While experimental data for visualization is already available in the [measurement table](#), simulations of the calibrated model still have to be computed. To generate model simulations, we extract the parameter values of the best pre-calibration. As optimization results are sorted in increasing order of objective function values, this can be achieved by taking the result at the first index. PEtab compatible simulation results can be extracted using the pyPESTO objective and the AMICI function [amici.petab_objective.rdatas_to_simulation_df](#). As the results contain the full parameter vector, including parameter values that were not estimated, but the objective only expects estimated parameters, the parameters need to be subsetted according to the free indices.

```
[30]: x = pypesto_problem . get_reduced_vector ( result . optimize_result . list [ 0 ] [ 'x' ] ,
                                             pypesto_problem . x_free_indices )
simulation = pypesto_problem . objective ( x , return_dict = True )
simulation_df = amici . petab_objective . rdatas_to_simulation_df (
    simulation [ 'rdatas' ] ,
    model = pypesto_problem . objective . amici_model ,
    measurement_df = measurement_table ,
)
```

We can now visualize both data and simulation using [petab.visualize.plot_problem](#).

```
[31]: import petab . visualize
import matplotlib . pyplot as plt
plt . rc ( ' font ' , size = 25 )
plt . rc ( ' lines ' , linewidth = 3 )
petab . visualize . plot_problem (
    petab_problem = petab_problem ,
    simulations_df = simulation_df ,
)
```

The generated figure (Fig. 2) suggests that model simulations are qualitatively similar to training data, but there are still large quantitative differences. This suggests that the pre-calibrated parameter values are an adequate first guess but require further refinement.

Before we proceed with the second calibration round, we can check the accuracy of the objective function gradient at the

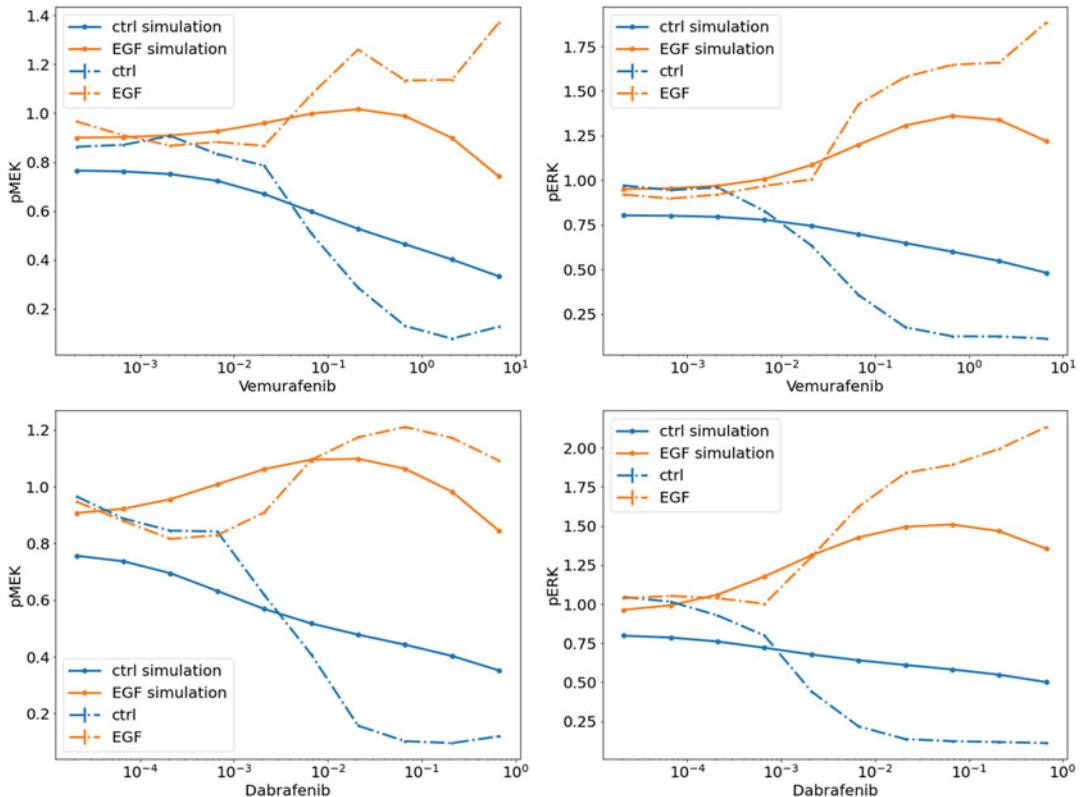


Fig. 2 Pre-calibration fit. Different rows correspond to different RAF inhibitors (vemurafenib and dabrafenib). Different columns correspond to different observables (pMEK and pERK). Colors indicate different experimental conditions (with and without EGF stimulation). Data is shown as dashed lines, and simulations as solid lines

pre-calibrated parameter values. For this purpose, we use the `check_grad_multi_eps` method, which approximates the objective function gradient using finite differences with different step sizes.

```
[32]: pd.set_option('display.max_rows', 5)
pd.set_option('display.max_columns', 8)
pypesto_problem.objective.check_grad_multi_eps(x, multi_eps=[1e-1, 1e-3, 1e-5], verbosity=0)
```

```
[33]:      grad    fd_f    fd_b    fd_c \
bind_RAF_RAF_Ea -0.000676  0.066034 -0.082687 -0.008326
bind_RAF_RAF_dG 151.299971 161.357110 141.392663 151.374887
pERK_offset     -14.721473 -7.347211 -22.133442 -14.740327
pERK_scale      -1593.045905 -1569.867010 -1616.507993 -1593.187501
                    fd_err  abs_err  rel_err  eps
bind_RAF_RAF_Ea  0.148720  0.007650  0.083453  0.100
bind_RAF_RAF_dG 19.964446  0.074916  0.000495  0.100
pERK_offset      14.786231  0.018853  0.001279  0.001
pERK_scale       46.640983  0.141596  0.000089  0.001
[28 rows x 8 columns]
```

The method computes forward (`fd_c`), backward (`fd_b`), and central (`fd_c`) finite differences and compares them to the sensitivity-based gradient (`grad`), using the step size `eps` that yields the lowest finite difference approximation error among the provided step sizes. The results show a low error in the finite difference approximation (`fd_err`), indicating accurate evaluation

of the objective function. Moreover, the small absolute (`abs_err`) and relative difference (`rel_err`) between sensitivity-based gradient and finite difference approximation indicates accurate evaluation of the gradient. The large magnitude of the gradient suggests that we did not yet reach a local optimum and that further optimization will likely improve agreement between data and simulation. In cases where either `fd_err`, `abs_err`, or `rel_err` takes large values, it is recommended to decrease integration tolerances until adequate numbers are achieved, where entries in `grad` can serve as reference.

We will now perform this refinement by instantiating a new optimizer with a larger computational budget of two hours.

```
[33]: optimizer_fine = FidesOptimizer ( options = { fides . Options . MAXTIME : 7200 } ,
                                     verbose = logging . ERROR )
```

To initialize the second calibration round at the pre-calibrated parameter values, we extract the respective values from the results object and set them as guesses in the pyPESTO problem.

```
[34]: n_refine = 10
pypesto_problem . x_guesses_full = np . vstack ( result . optimize_result . x [ : n_refine ] )
```

For startpoint generation, pyPESTO will first start optimization runs at the provided parameter guesses and only sample new startpoint if more optimization runs than guesses are requested. As for the first calibration run, we run the calibration, using the pyPESTO [minimize](#) function.

```
[35]: result_refined = minimize (
    Pypesto_problem , optimizer_fine , n_starts = n_refine ,
    engine = pypesto . engine . MultiThreadEngine ( 4 ) ,
    progress_bar = False ,
)
```

As for the pre-calibration, we now visualize the results using PETab and the previously constructed visualization table.

```
[36]: x = pypesto_problem . get_reduced_vector (
    result_refined . optimize_result . list [ 0 ] [ 'x' ] ,
    pypesto_problem . x_free_indices
)
simulation = pypesto_problem . objective ( x , return_dict = True )
simulation_df = anici . petab_objective . rdatas_to_simulation_df (
    simulation [ 'rdatas' ] ,
    model = importer . create_model () ,
    measurement_df = measurement_table ,
)
plt . rc ( ' font ' , size = 20 )
plt . rc ( ' lines ' , linewidth = 3 )
petab . visualize . plot_problem (
    petab_problem = petab_problem ,
    simulations_df = simulation_df ,
)
```

These generated figures (Fig. 3) now indicate adequate agreement between experimental data and model simulations. As a last step of the analysis, we now compare the objective function values across both calibration rounds using a waterfall plot.

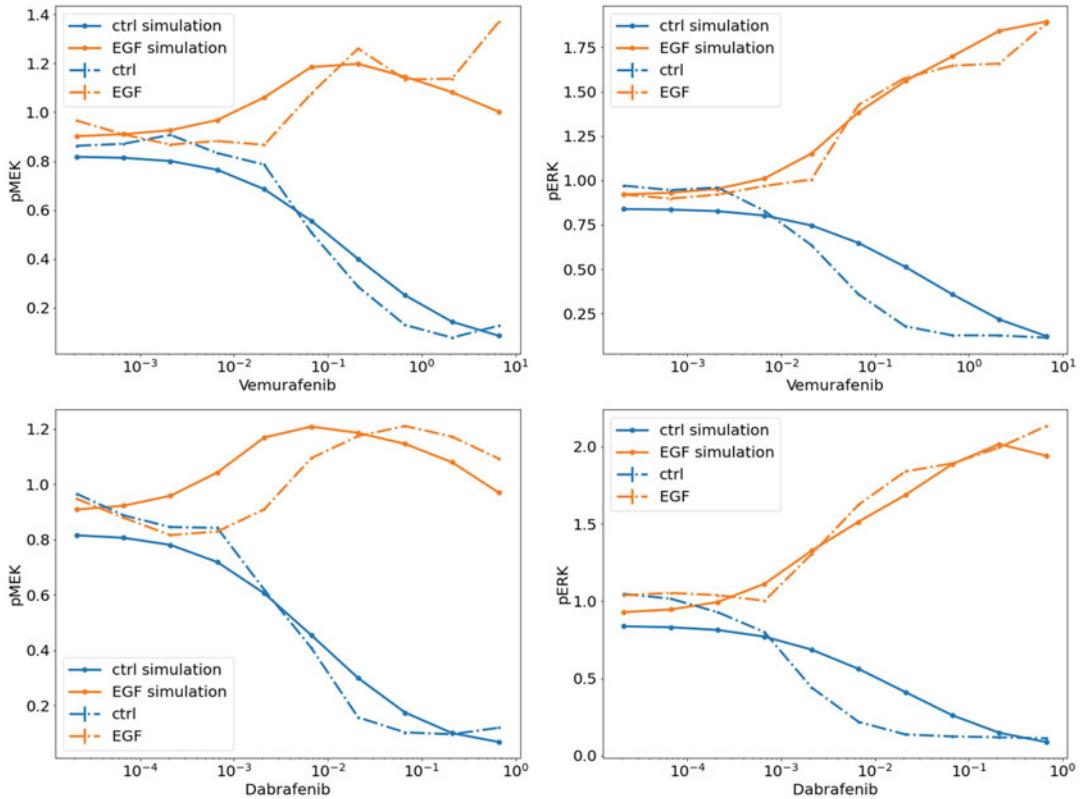


Fig. 3 Calibration fit. Different rows correspond to different RAF inhibitors (vemurafenib and dabrafenib). Different columns correspond to different observables (pMEK and pERK). Colors indicate different experimental conditions (with and without EGF stimulation). Data is shown as dashed lines, and simulations as solid lines

```
[37]: from pypesto.visualize import waterfall
waterfall( results = [ result , result_refined ] ,
legends = [ 'pre-calibration' , 'calibration' ] ) ;
```

The generated waterfall plot (Fig. 4) shows sorted objective function values of different optimizer runs for pre-calibration (blue) and calibration (yellow). The objective function values are normalized to the best objective function value achieved across both runs. We observe that for the majority of fully calibrated runs, we achieve much lower objective function values compared to the pre-calibrated runs, indicating that the higher computational budget facilitated better fits. For a small subset of starts, we observe repeated convergence to similar objective function values that form a plateau in the waterfall plot, suggesting that the respective runs may have converged to the same local minimum. Ideally, the majority of objective function values would lie in a plateau, but this would likely require many more optimizer runs with larger computational budgets and would thus have to be performed on a computer cluster.

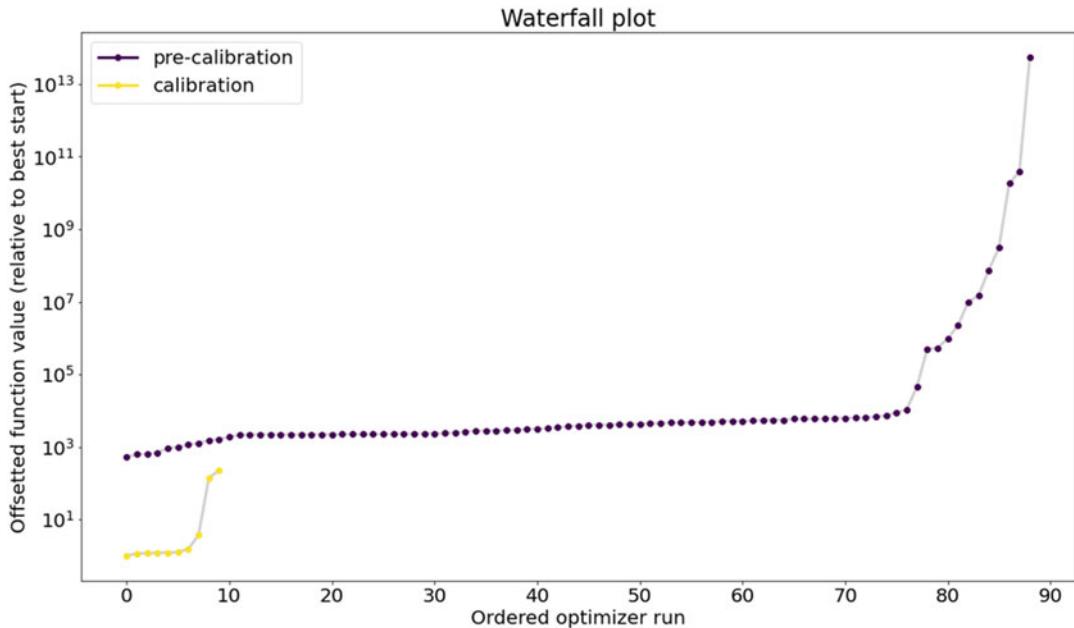


Fig. 4 Waterfall plot. Objective function values from (pre-) calibration runs are sorted according to their sorted numerical value. Y-axis is offsetted such that lowest objective function value is shown at 1

6 Discussion

Even with state-of-the-art methods, the process of model formulation, problem specification, and calibration we described in this protocol remains a labor-intensive process, where good understanding of the underlying mathematical concepts is necessary to achieve optimal, or even adequate performance for large kinetic models. We anticipate that the extensive description and code we provide in this protocol will serve as an introduction to model calibration and may be used as template for future research projects.

In terms of future developments, more automation in terms of tolerance selection, sensitivity methods, and computational budget allocation will be necessary to improve accessibility of methods and reduce the amount of required user interaction. Some frameworks such as Data2Dynamics [35] already provide adaptive tolerance updating schemes in cases of numerical integration failure. This could still be complemented by adaptive updating of tolerances based on objective function evaluation accuracy, but underlying mathematical theory and good benchmark problems are missing. We are convinced that the introduction of large benchmarks in [23, 33], guidelines for their evaluation [42] as well as the standardization of their formulation [26] will prove to be crucial to improve automation of model calibration in the future.

References

1. Lavoie H, Gagnon J, Therrien M (2020) ERK signalling: a master regulator of cell behaviour, life and fate. *Nat Rev Mol Cell Biol* 21:607–632
2. Ullrich A, Schlessinger J (1990) Signal transduction by receptors with tyrosine kinase activity. *Cell* 61:203–212
3. McKay MM, Morrison DK (2007) Integrating signals from RTKs to ERK/MAPK. *Oncogene* 26:3113
4. Hunter T (2000) Signaling—2000 and beyond. *Cell* 100:113–127
5. Sanchez-Vega F, Mina M, Armenia J, Chatila WK, Luna A, La KC, Dimitriadoy S, Liu DL, Kantheti HS, Saghafinia S, Chakravarty D, Daian F, Gao Q, Bailey MH, Liang W-W, Foltz SM, Shmulevich I, Ding L, Heins Z, Ochoa A, Gross B, Gao J, Zhang H, Kundra R, Kandoth C, Bahceci I, Dervishi L, Dogrusoz U, Zhou W, Shen H, Laird PW, Way GP, Greene CS, Liang H, Xiao Y, Wang C, Iavarone A, Berger AH, Bivona TG, Lazar AJ, Hammer GD, Giordano T, Kwong LN, McArthur G, Huang C, Tward AD, Frederick MJ, McCormick F, Meyerson M, Caesar-Johnson SJ, Demchok JA, Felau I, Kasapi M, Ferguson ML, Hutter CM, Sofia HJ, Tarnuzzer R, Wang Z, Yang L, Zenklusen JC, Zhang JJ, Chudamani S, Liu J, Lolla L, Naresh R, Pihl T, Sun Q, Wan Y, Wu Y, Cho J, DeFreitas T, Frazer S, Gehlenborg N, Getz G, Heiman DI, Kim J, Lawrence MS, Lin P, Meier S, Noble MS, Saksena G, Voet D, Zhang H, Bernard B, Chambwe N, Dhankani V, Knijnenburg T, Kramer R, Leinonen K, Liu Y, Miller M, Reynolds S, Shmulevich I, Thorsson V, Zhang W, Akbani R, Broom BM, Hegde AM, Ju Z, Kanichi RS, Korkut A, Li J, Liang H, Ling, Liu W, Lu Y, Mills GB, Ng K-S, Rao A, Ryan M, Wang J, Weinstein JN, Zhang J, Abeshouse A, Armenia J, Chakravarty D, Chatila WK, de Brujin I, Gao J, Gross BE, Heins ZJ, Kundra R, La K, Ladanyi M, Luna A, Nissan MG, Ochoa A, Phillips SM, Reznik E, Sanchez-Vega F, Sander C, Schultz N, Sheridan R, Onur Sumer S, Sun Y, Taylor BS, Wang J, Zhang H, Anur P, Peto M, Spellman P, Benz C, Stuart JM, Wong , Yau C, Neil Hayes D, Parker JS, Wilkerson MD, Ally A, Balasundaram M, Bowlby R, Brooks D, Carlsen R, Chuah E, Dhalla N, Holt R, Jones SJM, Kasaian K, Lee D, Ma Y, Marra MA, Mayo M, Moore RA, Mungall AJ, Mungall K, Gordon Robertson A, Sadeghi S, Schein JE, Sipahimalani P, Tam A, Thiessen N, Tse K, Wong T, Berger AC, Beroukhim R, Cherniack AD, Cibulskis C, Gabriel SB, Gao GF, Ha G, Meyerson M, Schumacher SE, Shih J, Kucherlapati MH, Kucherlapati RS, Baylin S, Cope L, Danilova L, Bootwalla MS, Lai PH, Maglione DT, Van Den Berg DJ, Weisenberger DJ, Todd Auman J, Balu S, Bodenheimer T, Fan C, Hoadley KA, Hoyle AP, Jefferys SR, Jones CD, Meng S, Mieczkowski PA, Mose LE, Perou AH, Perou CM, Roach J, Shi Y, Simons JV, Skelly T, Soloway MG, Tan D, Veluvolu U, Fan H, Hinoue T, Laird PW, Shen H, Zhou W, Bellair M, Chang K, Covington K, Creighton CJ, Dinh H, Doddapaneni HV, Donehower LA, Drummond J, Gibbs RA, Glenn R, Hale W, Han Y, Hu J, Korchina V, Lee S, Lewis L, Li W, Liu X, Morgan M, Morton D, Muzny D, Santibanez J, Sheth M, Shinbrot E, Wang L, Wang M, Wheeler DA, Xi L, Zhao F, Hess J, Appelbaum EL, Bailey M, Cordes MG, Ding L, Fronick CC, Fulton LA, Fulton RS, Kandoth C, Mardis ER, McLellan MD, Miller CA, Schmidt HK, Wilson RK, Crain D, Curley E, Gardner J, Lau K, Mallory D, Morris S, Paulauskis J, Penny R, Shelton C, Shelton T, Sherman M, Thompson E, Yena P, Bowen J, Gastier-Foster JM, Gerken M, Leraas KM, Lichtenberg TM, Ramirez NC, Wise L, Zmuda E, Corcoran N, Costello T, Hovens C, Carvalho AL, de Carvalho AC, Fregnani JH, Longatto-Filho A, Reis RM, Scapulatempo-Neto C, Silveira HCS, Vidal DO, Burnette A, Eschbacher J, Hermes B, Noss A, Singh R, Anderson ML, Castro PD, Ittmann M, Huntsman D, Kohl B, Le X, Thorp R, Andry C, Duffy ER, Lyadov V, Paklina O, Setdikova G, Shabunin A, Tavobilov M, McPherson C, Warnick R, Berkowitz R, Cramer D, Feltmate C, Horowitz N, Kibel A, Muto M, Raut CP, Malykh A, Barnholtz-Sloan JS, Barrett W, Devine K, Fulop J, Ostrom QT, Shimmel K, Wolinsky Y, Sloan AE, De Rose A, Giulante F, Goodman M, Karlan BY, Hagedorn CH, Eckman J, Harr J, Myers J, Tucker K, Zach LA, Deyarmin B, Hu H, Kvecher L, Larson C, Mural RJ, Somiari S, Vicha A, Zelinka T, Bennett J, Iacocca M, Rabeno B, Swanson P, Latour M, Lacombe L, TÅltu B, Bergeron A, McGraw M, Staugaitis SM, Chabot J, Hibshoosh H, Sepulveda A, Su T, Wang T, Potapova O, Voronina O, Desjardins L, Mariani O, Roman-Roman S, Sastre X, Stern M-H, Cheng F, Signoretti S, Berchuck A, Bigner D, Lipp E, Marks J, McCall S, McLendon R, Secord A, Sharp A, Behera M, Brat DJ, Chen A, Delman K, Force S, Khuri F, Magliocca K, Maithel S,

Olson JJ, Owonikoko T, Pickens A, Ramalingam S, Shin DM, Sica G, Van Meir EG, Zhang H, Eijkenboom W, Gillis A, Korpershoek E, Looijenga L, Oosterhuis W, Stoop H, van Kessel KE, Zwarthoff EC, Calatozzolo C, Cuppini L, Cuzzubbo S, DiMeco F, Finocchiaro G, Mattei L, Perin A, Pollo B, Chen C, Houck J, Lohavanichbutr P, Hartmann A, Stoehr C, Stoehr R, Taubert H, Wach S, Wullrich B, Kyeler W, Murawa D, Wiznerowicz M, Chung K, Jeffrey Edenfield W, Martin J, Baudin E, Bubley G, Bueno R, De Rienzo A, Richards WG, Kalkanis S, Mikkelsen T, Noushmehr H, Scarpace L, Girard N, Aymerich M, Campo E, Ginál’ E, Guillermo AL, Van Bang N, Hanh PT, Phu BD, Tang Y, Colman H, Evasion K, Dottino PR, Martignetti JA, Gabra H, Juhl H, Akeredolu T, Stepa S, Hoon D, Ahn K, Kang KJ, Beuschlein F, Breggia A, Birrer M, Bell D, Borad M, Bryce AH, Castle E, Chandan V, Cheville J, Copland JA, Farnell M, Flotte T, Giama N, Ho T, Kendrick M, Kocher J-P, Kopp K, Moser C, Nagorney D, OázBrien D, OázNeill BP, Patel T, Petersen G, Que F, Rivera M, Roberts L, Smallridge R, Smyrk T, Stanton M, Houston Thompson R, Torbenson M, Yang JD, Zhang L, Brimo F, Ajani JA, Angulo Gonzalez AM, Behrens C, Bondaruk J, Broaddus R, Czerniak B, Esmaeli B, Fujimoto J, Gershenson J, Guo C, Lazar AJ, Logothetis C, Meric-Bernstam F, Moran C, Ramondetta L, Rice D, Sood A, Tamboli P, Thompson T, Troncoso P, Tsao A, Wistuba I, Carter C, Haydu L, Hersey P, Jakrot V, Kakavand H, Kefford R, Lee K, Long G, Mann G, Quinn M, Saw R, Scolyer R, Shannon K, Spillane A, Stretch J, Synott M, Thompson J, Wilmott J, Al-Ahmadie H, Chan TA, Ghossein R, Gopalan A, Levine DA, Reuter V, Singer S, Singh B, Tien NV, Broudy T, Mirsaidi C, Nair P, Drwiega P, Miller J, Smith J, Zaren H, Park J-W, Hung NP, Kebebew E, Marston Linehan W, Metwalli AR, Pacak K, Pinto PA, Schiffman M, Schmidt LS, Vocke CD, Wentzensen N, Worrell R, Yang H, Moncrieff M, Goparaju C, Melamed J, Pass H, Botnariuc N, Caraman I, Cernat M, Chemencedji I, Clipca A, Doruc S, Gorincioi G, Mura S, Pirtac M, Stanciu I, Tcaciu D, Albert M, Alexopoulou I, Arnaout A, Bartlett J, Engel J, Gilbert S, Parfitt J, Sekhon H, Thomas G, Rassl DM, Rintoul RC, Bifulco C, Tamakawa R, Urba W, Hayward N, Timmers H, Antenucci A, Facciolo F, Grazi G, Marino M, Merola R, de Krijger R, Gimenez-Roqueplo A-P, Pichál A, Chevalier S, McKercher G, Birsoy K,

Barnett G, Brewer C, Farver C, Naska T, Pennell NA, Raymond D, Schilero C, Smolenski K, Williams F, Morrison C, Borgia JA, Liptay MJ, Pool M, Seder CW, Junker K, Omberg L, Dinkin M, Manikhas G, Alvaro D, MC Bragazzi, Cardinale V, Carpino G, Gaudio E, Chesla D, Cottingham S, Dubina M, Moiseenko F, Dhanasekaran R, Becker K-F, Janssen K-P, Slotta-Huspenina J, Abdel-Rahman MH, Aziz D, Bell S, Cebulla CM, Davis A, Duell R, Bradley Elder J, Hilty J, Kumar B, Lang J, Lehman NL, Mandt R, Nguyen P, Pilarski R, Rai K, Schoenfeld L, Senecal K, Wakely P, Hansen P, Lechan R, Powers J, Tischler A, Grizzle WE, Sexton KC, Kastl A, Henderson J, Porten S, Waldmann J, Fassnacht M, Asa SL, Schadendorf D, Couce M, Graefen M, Huland H, Sauter G, Schlomm T, Simon R, Tennstedt P, Olabode O, Nelson M, Bathe O, Carroll PR, Chan JM, Disaia P, Glenn P, Kelley RK, Landen CN, Phillips J, Prados M, Simko J, Smith-McCune K, VandenBerg S, Roggin K, Fehrenbach A, Kendler A, Sifri, Steele SR, Jimeno A, Carey F, Forgie I, Mannelli M, Carney M, Hernandez B, Campos B, Herold-Mende C, Jungk C, Unterberg A, von Deimling A, Bossler A, Galbraith J, Jacobus L, Knudson M, Knutson T, Ma D, Milhem M, Sigmund R, Godwin AK, Madan R, Rosenthal HG, Adebamowo C, Adebamowo SN, Boussioutas A, Beer D, Giordano T, Mes-Masson A-M, Saad F, Bocklage T, Landrum L, Mannel R, Moore K, Moxley K, Postier R, Walker J, Zuna R, Feldman M, Valdivieso F, Dhir R, Luketich J, Mora Pinero EM, Quintero-Aguilo M, Carlotti Jr. CG, Dos Santos JS, Kemp R, Sankarankutty A, Tirapelli D, Catto J, Agnew K, Swisher E, Creaney J, Robinson B, Shelley CS, Godwin EM, Kendall S, Shipman C, Bradford C, Carey T, Haddad A, Moyer J, Peterson L, Prince M, Rozek L, Wolf G, Bowman R, Fong KM, Yang I, Korst R, Kimryn Rathmell W, Fantacone-Campbell JL, Hooke JA, Kovatich AJ, Shriver CD, DiPersio J, Drake B, Govindan R, Heath S, Ley T, Van Tine B, Westervelt P, Rubin MA, Lee JI, Arends ND, Mariamidze A, Van Allen EM, Cherniack AD, Ciriello G, Sander C, Schultz N (2018) Oncogenic signaling pathways in the cancer genome atlas. *Cell* 173:321–337.e10

6. Sawyers C (2004) Targeted cancer therapy. *Nature* 432:294–297
7. Lito P, Pratillas CA, Joseph EW, Tadi M, Halilovic E, Zubrowski M, Huang A, Wong WL, Callahan MK, Merghoub T, Wolchok JD, de Stanchina E, Chandrarlapaty S,

- Poulikakos PI, Fagin JA, Rosen N (2012) Relief of profound feedback inhibition of mitogenic signaling by RAF inhibitors attenuates their activity in brafv600e melanomas. *Cancer Cell* 22:668–682
8. Fitzgerald JB, Schoeberl B, Nielsen UB, Sorger PK (2006) Systems biology and combination therapy in the quest for clinical efficacy. *Nat Chem Biol* 2:458–466
9. Gyori BM, Bachman JA, Subramanian K, Muhlich JL, Galescu L, Sorger PK (2017) From word models to executable models of signaling networks using automated assembly. *Mol Syst Biol* 13:954
10. Aldridge BB, Burke JM, Lauffenburger DA, Sorger PK (2006) Physicochemical modeling of cell signalling pathways. *Nat Cell Biol* 8: 1195–1203
11. Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, Cuellar AA, Dronov S, Gilles ED, Ginkel M, Gor V, Goryanin II, Hedley WJ, Hodgman TC, Hofmeyr J-H, Hunter PJ, Juty NS, Kasberger JL, Kremling A, Kummer U, Le Novâtre N, Loew LM, Lucio D, Mendes P, Minch E, Mjolsness ED, Nakayama Y, Nelson MR, Nielsen PF, Sakurada T, Schaff JC, Shapiro BE, Shimizu TS, Spence HD, Stelling J, Takahashi K, Tomita M, Wagner J, Wang J (2003) The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 19:524–531
12. Boutillier P, Maasha M, Li X, Medina-Abarca HF, Krivine J, Feret J, Cristescu I, Forbes AG, Fontana W (2018) The Kappa platform for rule-based modeling. *Bioinformatics* 34:i583–i592
13. Blinov ML, Faeder JR, Goldstein B, Hlavacek WS (2004) BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20:3289–3291
14. Hlavacek WS, Faeder JR, Blinov ML, Posner RG, Hucka M, Fontana W (2006) Rules for modeling signal-transduction systems. *Science's STKE* 2006:re6
15. Ollivier JF, Shahrezaei V, Swain PS (2010) Scalable rule-based modelling of allosteric proteins and biochemical networks. *PLOS Comput Biol* 6:e1000975
16. Sekar JAP, Hogg JS, Faeder JR (2016) Energy-based modeling in BioNetGen. In: 2016 IEEE international conference on bioinformatics and biomedicine (BIBM), pp 1460–1467
17. Kholodenko BN (2015) Drug resistance resulting from kinase dimerization is rationalized by thermodynamic factors describing allosteric inhibitor effects. *Cell Rep* 12:1939–1949
18. Rukhlenko OS, Khorsand F, Krstic A, Rozanc J, Alexopoulos LG, Rauch N, Erickson KE, Hlavacek WS, Posner RG, Gómez-Coca S, Rosta E, Fitzgibbon C, Matallanas D, Rauch J, Kolch W, Kholodenko BN (2018) Dissecting RAF inhibitor resistance by structure-based modeling reveals ways to overcome oncogenic RAS signaling. *Cell Syst* 7:161–179.e14
19. Wegscheider R (1911) über simultane gleichgewichte und die beziehungen zwischen thermodynamic und reactionskinetik homogener systeme. *Monatshefte für Chemie und verwandte Teile anderer Wissenschaften* 32:849–906
20. Poulikakos PI, Zhang C, Bollag G, Shokat KM, Rosen N (2020) RAF inhibitors transactivate RAF dimers and ERK signalling in cells with wild-type BRAF. *Nature* 464:427–430
21. Hatzivassiliou G, Song K, Yen I, Brandhuber BJ, Anderson DJ, Alvarado R, Ludlam MJ, Stokoe D, Gloor SL, Vigers G, Morales T, Aliagas I, Liu B, Sideris S, Hoeflich KP, Jaiswal BS, Seshagiri S, Koeppen H, Belvin M, Lori S (2010) RAF inhibitors prime wild-type RAF to activate the MAPK pathway and enhance growth. *Nature* 464:431–435
22. Heidorn SJ, Milagre C, Whittaker S, Nourry A, Niculescu-Duvas I, Dhomen N, Hussain J, Reis-Filho JS, Springer CJ, Pritchard C, Marais R (2010) Kinase-dead BRAF and oncogenic RAS cooperate to drive tumor progression through CRAF. *Cell* 140:209–221
23. Villaverde AF, Fröhlich F, Weindl D, Hasenauer J, Banga JR (2019) Benchmarking optimization methods for parameter estimation in large kinetic models. *Bioinformatics* 35:830–838
24. Fröhlich F, Kaltenbacher B, Theis FJ, Hasenauer J (2017) Scalable parameter estimation for genome-scale biochemical reaction networks. *PLoS Comput Biol* 13:1–18
25. Babtie AC, Stumpf MPH (2017) How to deal with parameters for whole-cell modelling. *J R Soc Interface* 14:20170237
26. Schmiester L, Schälte Y, Bergmann FT, Camba T, Dudkin E, Egert J, Fröhlich F, Fuhrmann L, Hauber AL, Kemmer S, Lakrisenko P, Loos C, Merkt S, Müller W, Pathirana D, Raimández E, Refisch L, Rosenblatt M, Stapor PL, Städter P, Wang D, Wieland F-G, Banga JR, Timmer J, Villaverde AF, Sahle S, Kreutz C, Hasenauer J, Weindl D (2021) Petabåtinteroperable specification of

- parameter estimation problems in systems biology. PLOS Comput Biol 17:e1008646
27. Gerosa L, Chidley C, Fröhlich F, Sanchez G, Lim SK, Muhlich J, Chen J-Y, Vallabhaneni S, Baker GJ, Schapiro D, Atanasova MI, Chylek LA, Shi T, Yi L, Nicora CD, Claas A, Ng TSC, Kohler RH, Lauffenburger DA, Weissleder R, Miller MA, Qian W-J, Steven Wiley H, Sorger PK (2020) Receptor-driven ERK pulses reconfigure MAPK signaling and enable persistence of drug-adapted BRAF-mutant melanoma cells. Cell Syst 11:478–494.e9
 28. Fröhlich F, Gerosa L, Muhlich J, Sorger PK (2022) Mechanistic model of MAPK signaling reveals how allosteric and rewiring contribute to drug resistance. bioRxiv 2022.02.17.480899
 29. Villaverde AF, Pathirana D, Fröhlich F, Hasenauer J, Banga JR (2022) A protocol for dynamic model calibration. Briefings Bioinform 23:bbab387
 30. Lopez CF, Muhlich JL, Bachman JA, Sorger PK (2013) Programming biological models in Python using PySB. Mol Syst Biol 9:646
 31. Fröhlich F, Weindl D, Schälte Y, Pathirana D, Paszkowski L, Lines GT, Stapor P, Hasenauer J (2021) AMICI: high-performance sensitivity analysis for large ordinary differential equation models. Bioinformatics 37(20):3676–3677
 32. Fröhlich F, Sorger PK (2022) Fides: reliable trust-region optimization for parameter estimation of ordinary differential equation models. PLOS Comput Biol 18:e1010322
 33. Hass H, Loos C, Raimundez-Aálvarez E, Timmer J, Hasenauer J, Kreutz C (2019) Benchmark problems for dynamic modeling of intracellular processes. Bioinformatics 35: 3073–3082
 34. Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, Singhal M, Xu L, Mendes P, Kummer U (2006) COPASI—a complex pathway simulator. Bioinf 22:3067–3074
 35. Räue A, Steiert B, Schelker M, Kreutz C, Maiwald T, Hass H, Vanlier J, Tönsing C, Adlung L, Engesser R, Mader W, Heinemann T, Hasenauer J, Schilling M, Höfer T, Klipp E, Theis FJ, Klingmüller U, Schöberl B, Timmer J (2015) Data2Dynamics: a modeling environment tailored to parameter estimation in dynamical systems. Bioinformatics 31:3558–3560
 36. Hindmarsh AC, Brown PN, Grant KE, Lee SL, Serban R, Shumaker DE, Woodward CS (2005) SUNDIALS: suite of Nonlinear and Differential/ALgebraic equation solvers. ACM Trans Math Softw 31:363–396
 37. Lines GT, Paszkowski L, Schmiester L, Weindl D, Stapor P, Hasenauer J (2019) Efficient computation of steady states in large-scale ODE models of biochemical reaction networks. IFAC-PapersOnLine 52:32–37
 38. Fiedler A, Raeth S, Theis FJ, Hausser A, Hasenauer J (2016) Tailored parameter optimization methods for ordinary differential equation models with steady-state constraints. BMC Syst Biol 10(1):1–19
 39. Tönsing C, Timmer J, Kreutz C (2019) Optimal paths between parameter estimates in non-linear ODE systems using the nudged elastic band method. Front Phys 7:149
 40. Städter P, Schälte Y, Schmiester L, Hasenauer J, Stapor PL (2020) Benchmarking of numerical integration methods for ODE models of biological systems. bioRxiv 2020.09.03.268276
 41. Fröhlich F, Loos C, Hasenauer J (2019) Scalable inference of ordinary differential equation models of biochemical processes. In: Gene regulatory networks: methods and protocols, pp 385–422
 42. Kreutz C (2019) Guidelines for benchmarking of optimization-based approaches for fitting mathematical models. Genome Biol 20:281



Chapter 4

Systems Biology: Identifiability Analysis and Parameter Identification via Systems-Biology-Informed Neural Networks

Mitchell Daneker, Zhen Zhang, George Em Karniadakis, and Lu Lu

Abstract

The dynamics of systems biological processes are usually modeled by a system of ordinary differential equations (ODEs) with many unknown parameters that need to be inferred from noisy and sparse measurements. Here, we introduce systems-biology-informed neural networks for parameter estimation by incorporating the system of ODEs into the neural networks. To complete the workflow of system identification, we also describe structural and practical identifiability analysis to analyze the identifiability of parameters. We use the ultradian endocrine model for glucose-insulin interaction as the example to demonstrate all these methods and their implementation.

Key words Systems biology, Parameter estimation, Structural identifiability, Practical identifiability, Physics-informed neural networks

1 Introduction

Systems biology aims to understand biological systems at a system level, including their structures and their dynamics [1]. Often, a biological system is modeled by a system of ordinary differential equations (ODEs), which describes the dynamics of the various concentrations of chemical and molecular species as a function of time. These biological models usually introduce some parameters that are unknown and required to be estimated accurately and efficiently. Hence, one central challenge in systems biology is the estimation of unknown model parameters (e.g., rate constants), after which we can perform the prediction of model dynamics. Parameter estimation requires observations of the state variables of the system, but due to technical limitations, only part of the state variables are observable in experiments, which makes parameter estimation even more difficult.

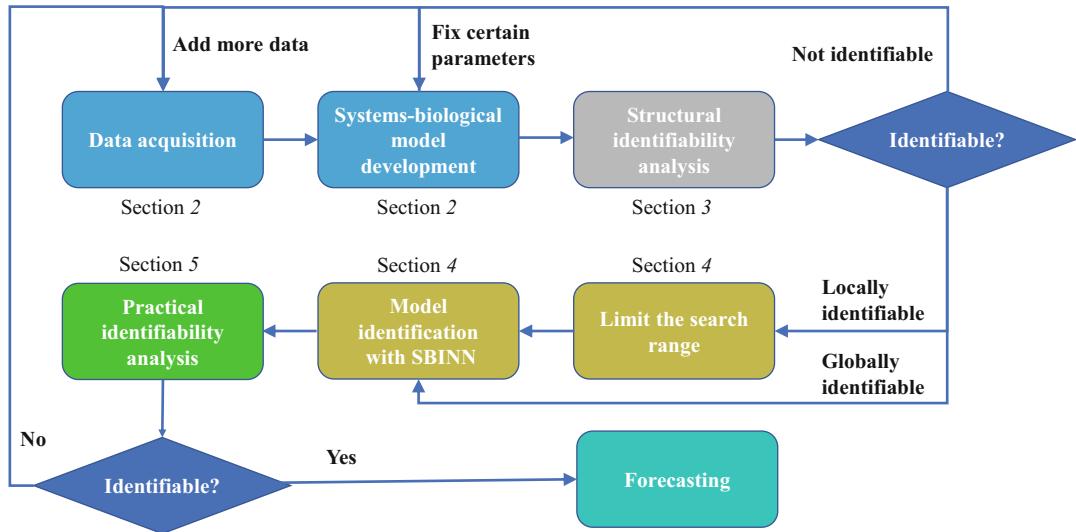


Fig. 1 The workflow for the development and identification of a systems biological model

In this chapter, we introduce the workflow for the development and identification of systems biological models (Fig. 1). The whole workflow includes the following several steps:

- Step 1: Data acquisition and systems-biological model development (Subheading 2). As the first step, we need to collect experimental data for the underlying system and develop ODEs to model the system dynamics. This is not the focus of this chapter, and we directly use the ultradian endocrine model for glucose-insulin interaction [2].
- Step 2: Structural identifiability analysis (Subheading 3). With a proposed model, we determine which parameters of the model are structurally identifiable. If the parameters are not structurally identifiable, Step 1 is revisited such as adding more data or fixing certain parameters. If the parameters are locally identifiable, we need to limit their search range.
- Step 3: Parameter estimation via systems-biology informed neural network (SBINN) (Subheading 4). We next use an SBINN to infer the unknown model parameters from the data.
- Step 4: Practical identifiability analysis (Subheading 5). With the inferred parameters, we check the quality of the estimates via practical identifiable analysis. If the parameters are practically identifiable, we can use the identified model for forecasting; otherwise, we need to revisit Step 1.

The code used in the chapter is publicly available from the GitHub repository <https://github.com/lu-group/sbinn>.

2 Ultradian Endocrine Model for Glucose-Insulin Interaction

To demonstrate the methods, we consider the system of the glucose-insulin interactions and use a relatively simple ultradian model [2] with 6 state variables and 21 parameters. The state variables are plasma insulin concentration I_p , interstitial insulin concentration I_i , glucose concentration G , and a three-stage filter (b_1, b_2, b_3) that mimics the response of the plasma insulin to glucose levels.

Equations (1) and (2) provide the system of equations for the model where major parameters include (i) E , a rate constant for exchange of insulin between the plasma and remote compartments; (ii) I_G , the exogenous (externally driven) glucose delivery rate; (iii) t_p , the time constant for plasma insulin degradation; (iv) t_i , the time constant for the remote insulin degradation; (v) t_d , the delay time between plasma insulin and glucose production; (vi) V_p , the volume of insulin distribution in the plasma; (vii) V_i , the volume of the remote insulin compartment; and (viii) V_g , the volume of the glucose space [2, 3]. Furthermore, in Eq. (2), $f_1(G)$ provides the rate of insulin production; $f_2(G)$ defines insulin-independent glucose utilization; $f_3(I_i)$ is the insulin-dependent glucose utilization; and $f_4(b_3)$ represents delayed insulin-dependent glucose utilization:

$$\frac{dI_p}{dt} = f_1(G) - E\left(\frac{I_p}{V_p} - \frac{I_i}{V_i}\right) - \frac{I_p}{t_p}, \quad \frac{dI_i}{dt} = E\left(\frac{I_p}{V_p} - \frac{I_i}{V_i}\right) - \frac{I_i}{t_i}, \quad (1a)$$

$$\frac{dG}{dt} = f_4(b_3) + I_G(t) - f_2(G) - f_3(I_i)G, \quad \frac{db_1}{dt} = \frac{1}{t_d}(I_p - b_1), \quad (1b)$$

$$\frac{db_2}{dt} = \frac{1}{t_d}(b_1 - b_2), \quad \frac{db_3}{dt} = \frac{1}{t_d}(b_2 - b_3), \quad (1c)$$

where f_1-f_4 and the nutritional driver of the model $I_G(t)$ are given by

$$f_1(G) = \frac{R_m}{1 + \exp\left(\frac{-G}{V_g C_1} + \alpha_1\right)}, \quad f_2(G) = U_b \left(1 - \exp\left(\frac{-G}{C_2 V_g}\right)\right), \quad (2a)$$

$$f_3(I_i) = \frac{1}{C_3 V_g} \left(U_0 + \frac{U_m}{1 + (\kappa I_i)^{-\beta}}\right), \quad f_4(b_3) = \frac{R_g}{1 + \exp\left(\alpha \left(\frac{b_3}{C_5 V_p} - 1\right)\right)}, \quad (2b)$$

$$\kappa = \frac{1}{C_4} \left(\frac{1}{V_i} + \frac{1}{E t_i}\right), \quad I_G(t) = \sum_{j=1}^N m_j k \exp(k(t_j - t)), \quad (2c)$$

where the nutritional driver $I_G(t)$ is a systematic forcing term that acts as nutritional intake of glucose and is defined over N discrete nutrition events [4] with k as the decay constant and event j occurs at time t_j with carbohydrate quantity m_j . The nominal values of the parameters are provided in 1.

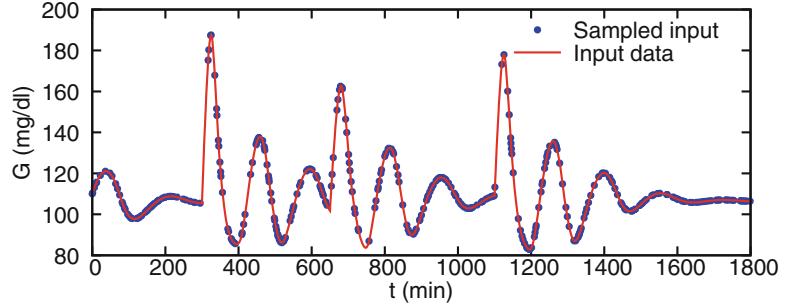


Fig. 2 Ultradian glucose-insulin model observation data for parameter inference. Three hundred and sixty measurements on glucose level (G) only are randomly sampled in the time window of 0–1800 min (~1 day). (Figure is adapted with permission from Ref. [6])

Synthetic data is generated by numerically solving the system from time $t = 0$ to $t = 1800$ min with the initial conditions $\mathbf{x}(0) = [12.0 \text{ } (\mu\text{U}/\text{ml}), 4.0 \text{ } (\mu\text{U}/\text{ml}), 110.0 \text{ } (\text{mg}/\text{dl}), 0.0, 0.0, 0.0]$ and three nutrition events $(t_j, m_j) = [(300, 60), (650, 40), (1100, 50)] \text{ (min,g)}$ pairs. This completes the first two steps of the flowchart in Fig. 1. We assume the only observable is the glucose level measurements G , which are sampled randomly, as shown in Fig. 2.

3 Structural Identifiability Analysis

In this section, we investigate whether a set of unknown parameters in the Ultradian endocrine model for glucose-insulin interaction is structurally identifiable from the glucose concentration data G . Simply fitting a model to the data is not sufficient to show how reliable the estimated parameters are. Insufficient data can produce very different sets of parameters without affecting the fit of data if a model is structurally non-identifiable. To resolve the non-identifiability issue, there are two options: one is to acquire data for more species, another is to fix certain parameters as their nominal values.

Suppose we are given a dynamical system of the following abstract form:

$$\dot{X}' = f(X, \Theta, u), \quad y = g(X, \Theta, u),$$

where $X = (X_1, \dots, X_n)$ represents the state variables, $y = (y_1, \dots, y_m)$ represents the observables, $\Theta = (\theta_1, \dots, \theta_k)$ contains the parameters to identify, and u represents the input variable to the system. A parameter set Θ is called structurally *globally* identifiable if

$$g(X, \Theta, u) = g(X, \Phi, u) \Rightarrow \Theta = \Phi \quad (3)$$

for every $\Phi = (\phi_1, \dots, \phi_k)$ in the same space as Θ . *Local* identifiability only requires Eq. (3) to hold in a neighborhood of Θ . As a consequence, if a model parameter turns out to be locally identifiable, it is suggested that one should limit the search range for this parameter before fitting the model. For globally identifiable parameters, this step is not required.

In this section, we only test for the local identifiability of the system since the existing software packages may suffer from out-of-memory issues when testing the global identifiability of a system with a large number of state variables and a small number of observables. For convenience, we will refer to a system as being identifiable when it is structurally locally identifiable. We use the Julia library *StructuralIdentifiability* [5] to test for structural identifiability of the model. The existing algorithms implemented in the library require both f and g to be rational functions, which are fractions of polynomials.

3.1 Preprocessing

There are exponential functions and power functions in Eq. (2) of the model, and thus a preprocessing step is required to get rid of the transcendental components of the system. One method is to introduce a set of extra state variables g_i which are equal to these transcendental components and apply the chain rule to find their derivatives. In our example, one can set

$$\begin{aligned} g_1(t) &= 1 + \exp\left(\frac{-G(t)}{V_g C_1} + \alpha_1\right), & g_2(t) &= 1 - \exp\left(\frac{-G(t)}{C_2 V_g}\right), \\ g_3(t) &= 1 + (\kappa I_i(t))^{-\beta}, & g_4(t) &= 1 + \exp\left(\alpha\left(\frac{h_3(t)}{C_5 V_p} - 1\right)\right). \end{aligned}$$

It follows from the chain rule that

$$\begin{aligned} \frac{dg_1}{dt} &= -\frac{g_1 - 1}{V_g C_1} \frac{dG}{dt}, & \frac{dg_2}{dt} &= -\frac{g_2 - 1}{V_g C_2}, \\ \frac{dg_3}{dt} &= -\beta k \frac{g_3 - 1}{k I_i} \frac{dI_i}{dt}, & \frac{dg_4}{dt} &= \frac{\alpha}{C_5 V_p} (g_4 - 1) \frac{dh_3}{dt}. \end{aligned}$$

The ODE system in Eqs. (1) and (2) can be rewritten in the following rational form:

$$\frac{dI_p}{dt} = \frac{R_m}{g_1} - E\left(\frac{I_p}{V_p} - \frac{I_i}{V_i}\right) + \frac{I_p}{t_p}, \quad \frac{dI_i}{dt} = E\left(\frac{I_p}{V_p} - \frac{I_i}{V_i}\right) - \frac{I_i}{t_i}, \quad (4a)$$

$$\frac{dG}{dt} = \frac{R_g}{g_1} + I_G - U_b g_2 - \frac{G}{C_3 V_g} \left(U_0 + \frac{U_m}{g_3} \right), \quad \frac{dg_1}{dt} = -\frac{g_1 - 1}{V_g C_1} \frac{dG}{dt}, \quad (4b)$$

$$\frac{dg_2}{dt} = -\frac{g_2 - 1}{V_g C_2}, \quad \frac{dg_3}{dt} = -\beta k \frac{g_3 - 1}{k I_i} \frac{dI_i}{dt}, \quad (4c)$$

$$\frac{dg_4}{dt} = \frac{\alpha}{C_5 V_p} (g_4 - 1) \frac{db_3}{dt}, \quad \frac{db_1}{dt} = \frac{1}{t_d} (I_p - b_1), \quad (4d)$$

$$\frac{db_2}{dt} = \frac{1}{t_d} (b_1 - b_2), \quad \frac{db_3}{dt} = \frac{1}{td} (b_2 - b_3), \quad (4e)$$

where I_G is treated as the input to the system and G is the output/observable of the system. Note that the initial conditions of all the ODE systems are assumed to be unknown for the *StructuralIdentifiability* library to work. This means there are four extra degrees of freedom lying in the initial conditions of g_1 , g_2 , g_3 , and g_4 in Eq. (4) compared to Eqs. (1) and (2). Consequently, any identifiable parameter in the new system will be identifiable in the original system, but not the other way around. Our goal now reduces to find a set of identifiable parameters in Eq. (4).

3.2 Structural Identifiability Results

By inspection of the three scaling invariances of the ODE in Eq. (4), namely,

$$\left\{ \begin{array}{l} R_m = cR_m \\ I_p(0) = cI_p(0) \\ I_i(0) = cI_i(0) \end{array} \right., \quad \left\{ \begin{array}{l} \alpha = ca \\ C_5 = cC_5 \end{array} \right., \quad \left\{ \begin{array}{l} U_0 = cU_0 \\ U_m = cU_m \\ C_3 = cC_3 \end{array} \right. \quad (5)$$

we found intrinsic structural non-identifiability of these parameters of the model. To get rid of the scaling invariances, one needs to fix one parameter in each system of equations of Eq. (5). For illustration purposes, we fix R_m , C_3 , and C_5 as their nominal values in Table 1 as an example and check whether the rest of the parameters are locally identifiable, see the code in Fig. 3.

Here, 15 undetermined parameters are remaining in the modified system, and it is impossible to fit all of them simultaneously, as shown in the first row of Table 2. This is reasonable because we assume that there is only one observable G and it is hard to infer all parameters with limited amount of data. As demonstrated in Fig. 1, to resolve the identifiability issue, one possible option is to acquire more data. It can be observed from the second row of Table 2 that taking I_i and I_G as additional observables makes t_p and t_i locally identifiable. Still, a large proportion of parameters remain structurally non-identifiable.

The second option (fixing certain parameters) is also considered. Here, we consider three different cases, where (V_p) , (V_p, V_i) , and (V_p, V_i, V_g) are fixed, respectively. We still assume that we only have the glucose concentration G available. In the fourth and fifth rows of Table 2, we see that more parameters become identifiable when we fix V_p and V_i , but the model is still not identifiable. It is only identifiable when all three parameters are set as fixed values.

In summary, the ODE system of Eq. (4) is structurally locally identifiable when R_m , C_3 , C_5 , V_p , V_i , and V_g are fixed. As a final step, we relate the identifiability of the modified system to original ultradian glucose-insulin model described by Eqs. (1) and (2).

Table 1
Parameters for the ultradian glucose-insulin model [3]

Parameter	Nominal value	Unit	Search range	Inferred value
V_p	3	lit	—	—
V_i	11	lit	—	—
V_g	10	lit	—	—
E	0.2	lit min ⁻¹	(0.100, 0.300)	0.201
t_p	6	min	(4.00, 8.00)	5.99
t_i	100	min	(60.0, 140)	101.20
t_d	12	min	(25/3, 50/3)	11.98
k	0.0083	min ⁻¹	(0.00166, 0.0150)	0.00833
R_m	209	mU min ⁻¹	(41.8, 376)	208.62
α_1	6.6		(1.32, 11.9)	6.59
C_1	300	mg lit ⁻¹	(60.0, 540)	301.26
C_2	144	mg lit ⁻¹	(28.8, 259)	37.65
C_3	100	mg lit ⁻¹	—	—
C_4	80	mU lit ⁻¹	(16.0, 144)	78.76
C_5	26	mU lit ⁻¹	(5.20, 46.8)	25.94
U_b	72	mg min ⁻¹	(14.4, 130)	71.33
U_0	4	mg min ⁻¹	(0.800, 7.20)	0.0406 C_3
U_m	90	mg min ⁻¹	(18.0, 162)	0.890 C_3
R_g	180	mg min ⁻¹	(36.0, 324)	179.86
α	7.5		(1.50, 13.5)	7.54
β	1.772		(0.354, 3.190)	1.783

The search range of the first seven parameters is adopted from [2], and the search range of the other parameters is $(0.2p^*, 1.8p^*)$, where p^* is the nominal value of that parameter

Note that the scaling invariance between R_m and $I_p(0)$, $I_i(0)$ breaks when the latter two are provided in the training as the initial conditions. Also the scaling invariance between α and C_5 does not hold in the original system, since the value for α can be uniquely determined by the initial condition for g_4 . The scaling invariance for U_0 , U_m , and C_3 still holds in Eqs. (1) and (2), but one can expect U_0/C_3 and U_m/C_3 to be a constant. Therefore, one needs to only to fix V_p , V_i , and V_g in the parameter estimation process.

```

using StructuralIdentifiability
ode = @ODEmodel(
    Ip'(t) = 209/g1(t)-E*(Ip(t)/Vp-Ii(t)/Vi)-Ip(t)/tp,
    Ii'(t) = E*(Ip(t)/Vp-Ii(t)/Vi)-Ii(t)/ti,
    G'(t) = Rg/g4(t)+IG(t)-Ub*g2(t)-(U0+Um/g3(t))/Vg/100*G(t),
    h1'(t) = (Ip(t)-h1(t))/td,
    h2'(t) = (h1(t)-h2(t))/td,
    h3'(t) = (h2(t)-h3(t))/td,
    g1'(t) = -(g1(t)-1)/Vg/C1*(Rg/g4(t)+IG(t)-Ub*g2(t)-
        (U0+Um/g3(t))/Vg/100*G(t)),
    g2'(t) = -(g2(t)-1)/Vg/C2*(Rg/g4(t)+IG(t)-Ub*g2(t)-
        (U0+Um/g3(t))/Vg/100*G(t)),
    g3'(t) = -beta*(g3(t)-1)/Ii(t)*(E*(Ip(t)/Vp-Ii(t)/Vi)-Ii(t)/ti),
    g4'(t) = alpha/26/Vp*(g4(t)-1)*(h2(t)-h3(t))/td,
    # Data Available
    y(t) = G(t) # Data of G
    # y2(t) = Ip(t), # Use this line, if we also have data of Ip
    # y3(t) = Ii(t), # Use this line, if we also have data of Ii
)
test_local_identifiability(ode)

```

Fig. 3 Specify the ODE model. We specify the parametric ODE model in Eq. (4) using the @ODEmodel macro. $x'(t)$ is the derivative of state variable $x(t)$, which is assumed to be unknown if not specified otherwise. $y(t)$ defines the output variable which is assumed to be given. The last line tests the local identifiability of the model

Table 2

Local structural identifiability result of the ultradian endocrine model with different observables and parameters

Parameter	V_p	V_i	V_g	E	t_p	t_i	t_d	C_1	C_2	U_b	U_0	U_m	R_g	α	β
Given G	\times	\times	\times	\times	\times	\times	\checkmark	\times	\times	\checkmark	\times	\times	\checkmark	\times	\checkmark
Given G, I_p, I_i	\times	\times	\times	\times	\checkmark	\checkmark	\checkmark	\times	\times	\checkmark	\times	\times	\checkmark	\times	\checkmark
Given G	–	\times	\times	\times	\times	\times	\checkmark	\times	\times	\checkmark	\times	\times	\checkmark	\checkmark	\checkmark
Given G	–	–	\times	\checkmark	\checkmark	\checkmark	\checkmark	\times	\times	\checkmark	\times	\times	\checkmark	\checkmark	\checkmark
Given G	–	–	–	\checkmark											

R_m , C_3 , and C_5 are prefixed. More parameters become structurally identifiable when more data (I_p and I_i) are given. With only G given, the model is structurally locally identifiable when V_p , V_i , and V_g are fixed

4 Parameter Estimation via SBINN

4.1 Deep Neural Networks

Deep neural networks (DNNs) function by recursively transforming inputs linearly and nonlinearly, i.e., compositional functions. Many types of DNNs have been developed such as convolutional neural networks and recurrent neural networks, and here we only consider fully connected neural networks (FNNs). An FNN is composed of many layers (Fig. 4). We denote an L -layer neural

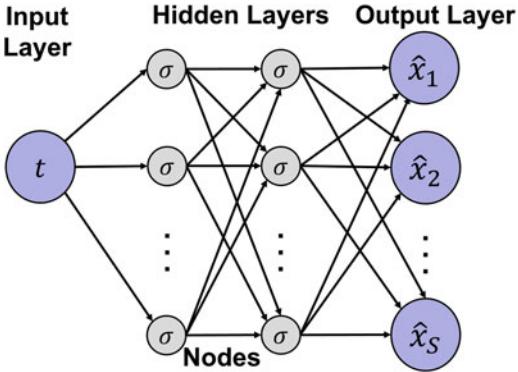


Fig. 4 Architecture of a fully connected neural network. A neural network consists of an input layer (the input t), several hidden layers (composed of weights W^ℓ , bias b^ℓ , and activation function σ), and an output layer

network (i.e., $(L - 1)$ hidden layers) by $\mathcal{N}^L(\mathbf{x}) : \mathbb{R}^{d_{\text{in}}} \rightarrow \mathbb{R}^{d_{\text{out}}}$, where d_{in} and d_{out} are the dimensions of the input and output, respectively. Each layer has a number of neurons, which can be thought as data processors, which take the output of the previous layer as the input, transform it, and then provide the output to the next layer. We use \mathcal{N}_ℓ to denote the number of neurons in the ℓ -th layer. At the input layer, we have $N_0 = d_{\text{in}}$, and at the output layer, we have $N_L = d_{\text{out}}$.

To define a FNN rigorously, in the ℓ -th layer, we define a weight matrix W^ℓ , a bias \mathbf{b}^ℓ , and an activation function σ . Examples of σ include logistic sigmoid ($1/(1 + e^{-x})$), the hyperbolic tangent (tanh), and the rectified linear unit (ReLU, $\max\{x, 0\}$). Then a FNN is defined as

input layer:	$\mathcal{N}^0(\mathbf{x}) = \mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$
hidden layers:	$\mathcal{N}^\ell(\mathbf{x}) = \sigma(W^\ell \mathcal{N}^{\ell-1}(\mathbf{x}) + b^\ell) \in \mathbb{R}^{N_\ell}, \text{ for } 1 \leq \ell \leq L - 1$
output layer:	$\mathcal{N}_L(\mathbf{x}) = W_L \mathcal{N}_{L-1}(\mathbf{x}) + b_L \in \mathbb{R}^{d_{\text{out}}}$

All the weights and biases are the neural network parameters $\boldsymbol{\theta}$.

4.2 Systems-Biology-Informed Neural Networks (SBINN)

SBINN was proposed in [6] and uses systems-biological models (e.g., Eqs. (1) and (2)) to inform a deep neural network. The network input is time t , and the output is a vector of state variables $\hat{\mathbf{x}}(t; \boldsymbol{\theta}) = (\hat{x}_1(t; \boldsymbol{\theta}), \hat{x}_2(t; \boldsymbol{\theta}), \dots, \hat{x}_S(t; \boldsymbol{\theta}))$, which acts as a proxy to the ODE solution.

We use Python to implement the code, see Appendix A for an introduction to Python. We can directly implement SBINN using general deep learning frameworks such as TensorFlow [7] and PyTorch [8], but the implementation becomes much easier if we use the open-source library DeepXDE [9]. DeepXDE is a library for scientific machine learning and can use either TensorFlow or

```
import deepxde as dde
from deepxde.backend import tf
```

Fig. 5 Importing DeepXDE and the TensorFlow backend

```
E_ = dde.Variable(0.0)
tp_ = dde.Variable(0.0)
...
beta_ = dde.Variable(0.0)
var_list_ = [E_, tp_, ... , beta_]
```

Fig. 6 Creating parameters to estimate. We initialize all parameters to zero and create a list of these parameters

PyTorch as its computational engine (called backend). We begin with importing DeepXDE and the backend being used (Fig. 5). Here we choose TensorFlow as the backend, and the code for PyTorch backend is almost the same.

We then implement SBINN. As the first step, we define all parameters to estimate (all the parameters in Table 1 except V_p , V_i , and V_g , which are easily measurable) with an initial guess of zero using `dde.Variable`, and create a list of all the variables to be used later (Fig. 6).

Next we use these parameters to implement the ODEs for the system. Because we only use the observation of G , based on our structural identifiability analysis, we need to limit the search range for the parameters. In this case, the range of seven parameters is adopted from [2], and the range for other parameters is set as $(0.2p^*, 1.8p^*)$, where p^* is the nominal value of that parameter (Table 1). We implement the search range and the ODE system of Eqs. (1) and (2) in Fig. 7.

The next step is to import the data measurements of glucose concentration G via `dde.PointSetBC` (Fig. 8). Other data measurements such as the initial conditions can be done similarly.

We have implemented the ODE system and data measurements. Next, we build our neural network model. In order to speed up network training, rather than just the FNN described in Subheading 4.1, we can add additional layers described as follows (Fig. 9).

- Input-scaling layer. In the case of a large time domain, t varies by multiple orders of magnitude, which negatively effects our NN training. We apply a linear scaling function to t using the maximum value of the time domain T to create $\tilde{t} = t/T$ which will be $\sim O(1)$.
- Feature layer. Often, ODE solutions have a pattern such as periodicity or exponential decay. Rather than letting a NN determine these features on its own, we add these patterns in a feature layer. Though feature choice is problem specific, the

```

def ODE(t, y):
    # y includes all state variables
    Ip = y[:, 0:1] # Ip is the first state variable
    ...# We import the rest in the same fashion
    h3 = y[:, 0:6]
    # Define search range for all variables
    E = (tf.tanh(E_)+1)*0.1+0.1 #(0.1,0.3)
    ...
    # Compute other variables
    f1 = Rm*tf.math.sigmoid(G/(Vg*C1)-a1)
    ...
    # Compute derivatives
    dIP_dt = dde.grad.jacobian(y,t,i=0,j=0)
    ...
    dh3_dt = dde.grad.jacobian(y,t,i=5,j=0)
    return [
        # Define the ODE in standard form
        dIP_dt-(f1-tmp-Ip/tp),
        ...
        dh3_dt-(h2-h3)/td,
    ]

```

Fig. 7 Implementation of the ODE system in Eqs. (1) and (2). The Python function ODE returns the residuals of all ODEs, i.e., the difference between the left-hand side and the right-hand size of the ODE

```

data_t = # The time instances of the measurements
data_G = # The value of G in the measurements
observe_y2 = dde.PointSetBC(data_t,data_G,component=2)

```

Fig. 8 Implementation of the data observation of G

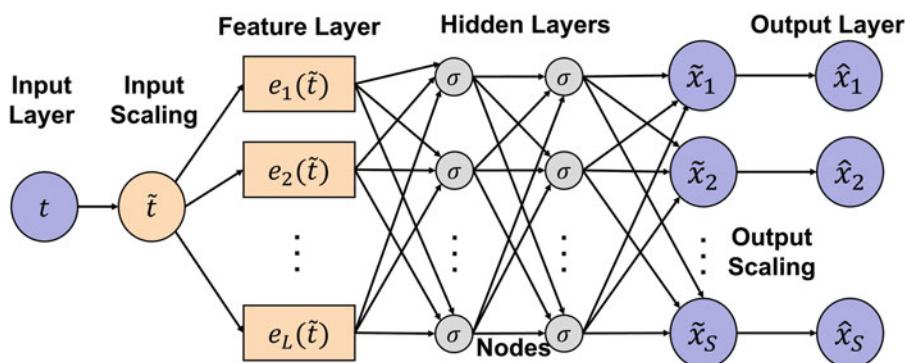


Fig. 9 Neural network architecture for SBINN. The input-scaling layer and output-scaling layer scale the network input and outputs to order one. The feature layer provides features directly to the first fully connected layer

```

def feature_transform(t):
    t = 0.01*t
    return tf.concat((t,tf.sin(t),tf.sin(2*t),tf.sin(3*t),
                      tf.sin(4*t),tf.sin(5*t))),axis=1)
net.apply_feature_transform(feature_transform)

```

Fig. 10 Input scaling and feature transform. We use the periodicity of sin as our feature

```

def output_transform(t, y):
    # The scaling values are [1,1,1e2,1,1,1]
    return tf.constant([1,1,1e2,1,1,1])*y
net.apply_output_transform(output_transform)

```

Fig. 11 Output transform to scale the outputs of the network

setup is similar for any problem. We use the L function $e_1(\cdot)$, $e_2(\cdot)$, ..., $e_L(\cdot)$ to construct the L features $e_1(\tilde{t})$, $e_2(\tilde{t})$, ..., $e_L(\tilde{t})$ as seen in Fig. 10. If no pattern is easily identifiable, it is best to leave out the feature layer than to include something incorrect; this is just a technique to aid in training, not a requirement for the SBINN to work.

- Output-scaling layer. The outputs \hat{x}_1 , \hat{x}_2 , ..., \hat{x}_S may have a disparity of magnitudes. As such, we can scale the network outputs by $\hat{x}_1 = k_1 \hat{x}_1$, $\hat{x}_2 = k_2 \hat{x}_2$, ..., $\hat{x}_S = k_S \hat{x}_S$ like in Fig. 11, where k_1 , k_2 , ..., k_S are the magnitudes of the ODE solution x_1 , x_2 , ..., x_S , respectively.

To train the neural network, we need to constrain it to the system of ODEs and the observations we created. This is done by defining a loss function, which computes the difference between the output of the neural network and the desired behavior: following the data at the time t_1 , t_2 , ..., $t_{N_{data}}$ and the ODEs at time points τ_1 , τ_2 , ..., $\tau_{N_{ode}}$. The ODE time points could be chosen at random or uniformly spaced. We define the total loss as a function of θ and p :

$$\mathcal{L}(\theta, p) = \mathcal{L}^{\text{data}}(\theta) + \mathcal{L}^{\text{ode}}(\theta, p) + \mathcal{L}^{\text{aux}}(\theta).$$

$\mathcal{L}^{\text{data}}$ is defined for M sets of observations y :

$$\begin{aligned} \mathcal{L}^{\text{data}}(\theta) &= \sum_{m=1}^M w_m^{\text{data}} \mathcal{L}_m^{\text{data}} \\ &= \sum_{m=1}^M w_m^{\text{data}} \left[\frac{1}{N^{\text{data}}} \sum_{n=1}^{N^{\text{data}}} (y_m(t_n) - \hat{x}_{s_m}(t_n; \theta))^2 \right]. \end{aligned}$$

\mathcal{L}^{ode} is for our system of ODEs:

$$\begin{aligned} \mathcal{L}^{\text{ode}}(\theta, p) &= \sum_{s=1}^S w_s^{\text{ode}} \mathcal{L}_s^{\text{ode}} \\ &= \sum_{s=1}^S w_s^{\text{ode}} \left[\frac{1}{N^{\text{ode}}} \sum_{n=1}^{N^{\text{ode}}} \left(\frac{d\hat{x}_s}{dt} \Big|_{\tau_n} - f_s(\hat{x}_s(\tau_n; \theta), \tau_n; p) \right)^2 \right]. \end{aligned}$$

The last term in the total loss function is \mathcal{L}^{aux} , which is used for additional information on system identification. For example, here we assume that we have the measurements of the state variables at two distinct times T_0 and T_1 . While this is essentially a part of the data loss, we include it as its own loss function due to it being given for all state variables at the two time instants. Here, we use the initial condition for T_0 and the final time instant for T_1 , and other choices of T_0 and T_1 can also be used depending on the available data information.

$$\begin{aligned}\mathcal{L}^{\text{aux}}(\boldsymbol{\theta}) &= \sum_{s=1}^S w_s^{\text{aux}} \mathcal{L}_s^{\text{aux}} \\ &= \sum_{s=1}^S w_s^{\text{aux}} \frac{(x_s(T_0) - \hat{x}_s(T_0; \boldsymbol{\theta}))^2 - (x_s(T_1) - \hat{x}_s(T_1; \boldsymbol{\theta}))^2}{2}.\end{aligned}$$

The weights w were selected such that all parts of the loss function would be of the same order of magnitude.

With the loss functions set up, we can train the network and infer the parameters of the ODEs \boldsymbol{p} by minimizing the loss function via a gradient-based optimizer, e.g., Adam [10]:

$$\boldsymbol{\Theta}^*, \boldsymbol{p}^* = \arg \min_{\boldsymbol{\theta}, \boldsymbol{p}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{p}).$$

We first train 10,000 epochs by setting all weights to zero except for data and then train against all parts of the loss function (Fig. 12). We also track the variables during training by using the callback *VariableValue*. We also plot the loss function as a function of epochs.

4.3 Results of SBINN

The inferred parameters are given in Table 1. We observe good agreement between the inferred values and the target values. We next consider the case of a nutrition event at $t_j = 2000$ min with carbohydrate intake of $m_j = 100$ g. We then test how well our model can predict this extra nutrition event using the inferred parameters, the results of which are in Fig. 13. With high accuracy, we determined the glucose levels after the nutrition event.

5 Practical Identifiability Analysis

The Fisher information matrix (FIM) can be used to develop confidence intervals of parameters as well as determine their practical identifiability, assuming parameters are structurally identifiable, as outlined in Fig. 1. The main difference between structural identifiability and practical identifiability lies in the fact that structural identifiability analysis is normally conducted before the fitting of the model and is used to study the uniqueness of parameters assuming noiseless observables. On the other hand, practical identifiability analysis is performed a posteriori and is often used to analyze whether the inferred parameter value will be sensitive to

```

model = dde.Model(data, net)
#Compiles and trains the model for the first time
model.compile("adam", lr=1e-3, loss_weights=[0,0,0,0,0,0,0,1e-2])
model.train(epochs=10000, display_every=1000)
#Sets up variable tracking
variablefilename = "variables.csv"
variable = dde.callbacks.VariableValue(var_list_, period=1000,
    filename=variablefilename)
#Compiles and trains the model and saves the loss history
model.compile("adam", lr=1e-3, loss_weights=[1,1,1e-2,1,1,1,1,1e-2],
    external_trainable_variables=var_list_)
losshistory, train_state = model.train(epochs=600000, display_every
    =1000, callbacks=[variable])

```

Fig. 12 Training the model using the Adam optimizer with a learning rate of 0.001. We train first for 10,000 epochs to let the network understand the data, then for 600,000 for the network to train all the losses

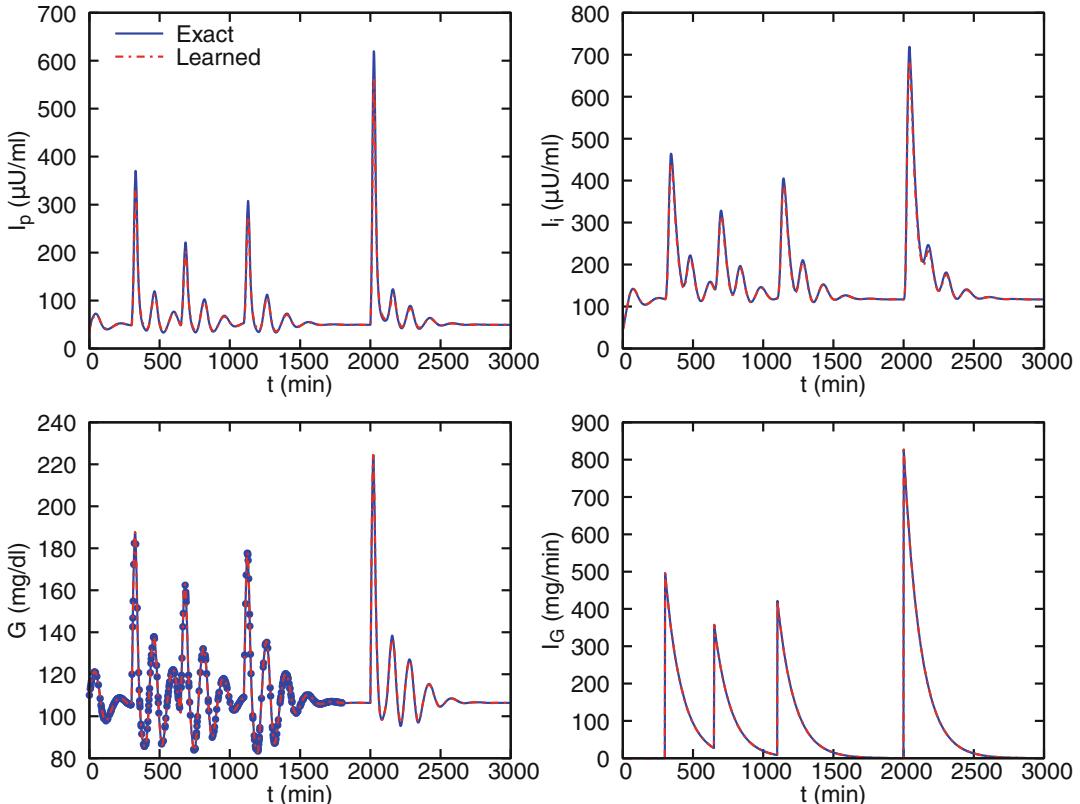


Fig. 13 Inferred dynamics and forecasting via SBINN. The network learned the system from time $t = [0, 1800]$. The estimated parameters were able to accurately forecast a meal event at $t_f = 2000$ min. (Figure is adapted with permission from [6])

```

function f_apop(dx, x, p, t)
    x1, x2, x3, x4, x5, x6 = x #Define observations
    E, tp, ..., beta = p #Define parameters
    meal_t = 300, 650, 1100 #Define constants
    meal_q = 60e3, 40e3, 50e3
    Vp = 3
    ...
    f1 = Rm/(1+exp(x3/(Vg*C1)-a1)) #Compute equations
    ...
    dt1 = t - meal_t[1]
    IG1 = 0.5 * meal_q[1] * k * exp(-k * dt1) * (sign(dt1) + 1)
    ...
    IG = IG1 + IG2 + IG3 #Compute sum differently than in Python
    dx[1] = (f1 - tmp - x1 / tp) #Compute derivatives
    ...
    dx[6] = (x5 - x6) / td
end

```

Fig. 14 Implementation of the ODE system

noise in the data. As a consequence, we need both analyses to determine whether the fitting result would be reliable.

We use Julia for practical identifiability analysis. In Julia, we import the required packages via using. If you are unfamiliar with Julia, see Appendix B. We start by defining the system in Eqs. (1) and (2). In our case, I_p is written as xl , and so on. For their derivatives, we define them as $dx[1]$, $dx[2]$, etc. We also declare all parameters that our SBINN determined as a vector p . System definition in Julia is found in Fig. 14.

To implement the practical sensitivity analysis, we first compute FIM, which is constructed by estimating the sensitivities of the system of ODEs with respect to the parameters. The code to compute FIM is in Fig. 15. We note that even though the data was generated with no noise, we need to consider a noise level of the measurements to compute a meaningful FIM, and we use a low noise level of 1% in the code. In this example, we only have one observable variable; the code for the problem with more than one observable is almost the same, and we only need to modify cov error and cols to indicate the indices of observable variables—see the example and code in [6].

There are different ways to utilize FIM, and here we show two important ones: (1) computing the correlation matrix of all parameters and (2) computing eigenvectors of FIM associated with the zero eigenvalues (i.e., null eigenvectors).

The correlation matrix R is computed as $R_{ij} = \text{FIM}_{ij}^{-1} / \text{FIM}_{ii}^{-1}$. $|R_{ij}| \approx 1$ indicates that the two parameters i and j are highly correlated,

```

p = [0.201, 5.99, ... 1.783] # Define inferred parameter values
x0 = [36., 44., 11000., 0., 0., 0.] # Define initial conditions
tspan = (0.0, 1800.0) # Define the timespan
prob_apop = ODELocalSensitivityProblem(f_apop,x0,tspan,p)
sol_apop = solve(prob_apop,alg_hints=[:stiff],saveat=0.1)
x_apop,dp_apop = extract_local_sensitivities(sol_apop)
sigma = 0.01 * std(x_apop,dims=2) # 0.01 is the noise level
cov_error = sigma[3] # Compute covariance
dp = dp_apop # dp is a matrix of sensitivities
cols = 3:3
Nt = length(dp[1][1,:])
Nstate = length(dp[1][:,1])
Nparam = length(dp[:,1])
FIM = zeros(Float64, Nparam, Nparam)
perm = vcat(1,sort(rand(2:Nt-1,Nt/5)),Nt)
for i in perm
    S = reshape(dp[1][:,i],(Nstate,1))
    for j = 2:Nparam
        S = hcat(S,reshape(dp[j][:,i],(Nstate,1)))
    end
    FIM += S[cols,:]'*inv(cov_error)*S[cols,:]
end

```

Fig. 15 Computing FIM. *sigma[3]* in the code refers to the standard deviation of the third state variable G

and thus are not individually identifiable from each other. The correlation matrix is shown in Fig. 16.

Next, we compute eigenvalues and eigenvectors of FIM. The eigenvalues is shown in Fig. 17 left. There is only one eigenvalue close to 0, and the associated eigenvector (i.e., null eigenvector) is shown in Fig. 17 right, where the value for the C_2 component is dominant and all other components are approximately zero. This indicates C_2 has little to no effect on state variable G and is therefore practically unidentifiable from the dataset.

In this example, the result from the null eigenvector analysis is consistent with our inferred values in Table 1, but the correlation matrix is not. We note that FIM-based practical identifiability analysis has many limitations and can be problematic in certain problems. There are other methods available for determining practical identifiability such as the bootstrapping approach [11] or using a probabilistic framework to quantify sensitivity of the system dynamics to variations in its parameters [12].

6 Discussion of Time-Dependent Parameters

We have considered all parameters are constants, but in real problems; the parameters could vary over time, i.e., time-dependent parameters. Here, we briefly describe the idea of implementing

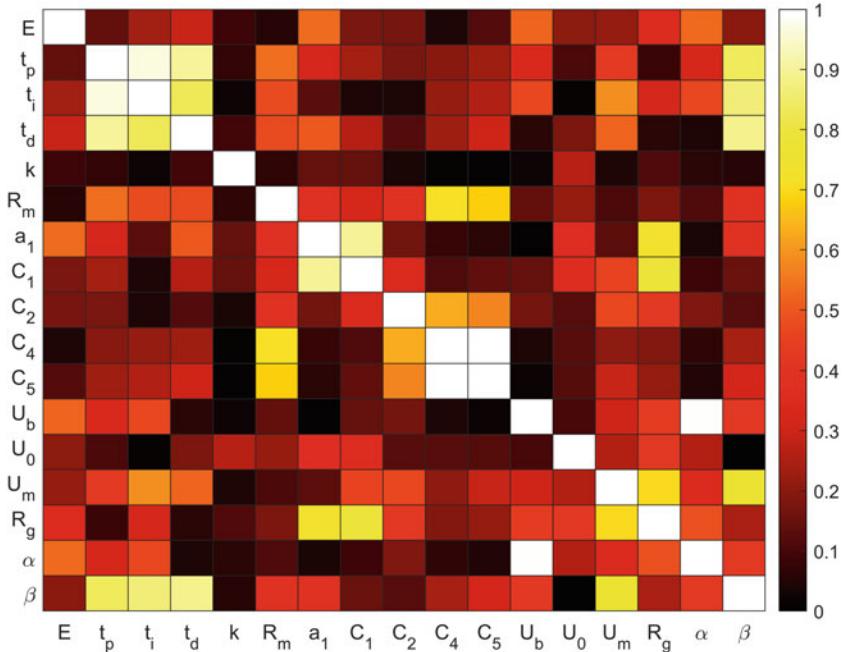


Fig. 16 Correlation matrix

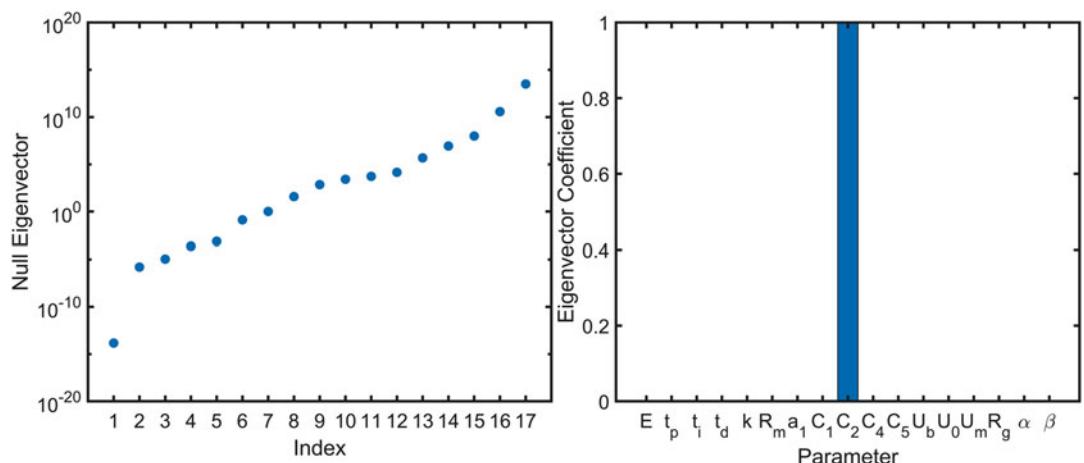


Fig. 17 Null eigenvector analysis. (Left) Eigenvalues of FIM. There is one eigenvalue close to 0. (Right) The eigenvector associated with the eigenvalue. The dominant component is C_2

time-dependent parameters in SBINN. Let us assume p_1 is a time-dependent parameter to be inferred. We add an extra neuron \hat{p}_1 in the network output to represent p_1 , as shown in Fig 18, and then \hat{p}_1 becomes a function of time. Everything else remains the same as the SBINN we introduced in Subheading 4.

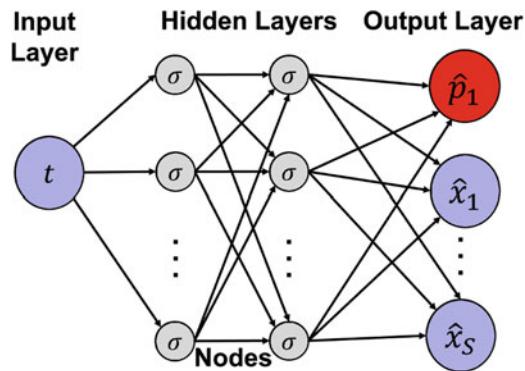


Fig. 18 SBINN for time-dependent parameters

7 Summary

We have provided a complete workflow for analyzing biological systems described by a system of ODEs, including structural identifiability analysis, parameter estimation via systems-biology-informed neural networks (SBINN), and practical identifiability analysis based on the Fisher information matrix (FIM).

Appendix

A. Python

Python is the most common language for machine learning due to the plethora of libraries available for free. Learning python is fairly easy due to its popularity, and there are a number of free, high-quality videos and other tutorials on how to use python. We note that common software for installing Python is Anaconda, from which most common libraries have already been installed. The code we provide should remove the majority of the guesswork if solving similar problems to those stated in this tutorial; otherwise, the DeepXDE documentation <https://deepxde.readthedocs.io> should be of help.

B. Julia

Julia is another language used for machine learning. It is very similar to Python as far as the syntax is concerned. We recommend using the softwares Atom and the Juno IDE, though Jupiter notebook and similar programs will suffice. There are also a variety of online sources that provide free help for learning this language. If you understand the fundamentals of Python, you should be able to read our provided Julia code and use it for your own situation with only a few minor tweaks that do not involve a heavy amount of coding or even a thorough knowledge of Julia.

References

1. Kitano H (2002) Systems biology: a brief overview. *Science* 295(5560):1662–1664
2. Sturis J, Polonsky KS, Mosekilde E, Van Cauter E (1991) Computer model for mechanisms underlying ultradian oscillations of insulin and glucose. *Am J Physiol Endocrinol Metab* 260(5):E801–E809
3. Albers DJ, Levine M, Gluckman B, Ginsberg H, Hripcak G, Mamikina L (2017) Personalized glucose forecasting for type 2 diabetes using data assimilation. *PLoS Comput Biol* 13(4):e1005232
4. Albers DJ, Elhadad N, Tabak E, Perotte A, Hripcak G (2014) Dynamical phenotyping: using temporal analysis of clinically collected physiologic data to stratify populations. *PloS One* 9(6):e96443
5. Dong R, Goodbrake C, Harrington HA, Pogudin G (2021) Differential elimination for dynamical models via projections with applications to structural identifiability. *arXiv preprint arXiv:2111.00991*
6. Yazdani A, Lu L, Raissi M, Karniadakis GE (2020) Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLoS Comput Biol* 16(11): e1007575
7. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al (2016) Tensorflow: a system for large-scale machine learning. In: 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), pp 265–283
8. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L et al (2019) Pytorch: an imperative style, high-performance deep learning library. *Adv Neural Inf Proces Syst* 32:8026–8037
9. Lu L, Meng X, Mao Z, Karniadakis GE (2019) DeepXDE: a deep learning library for solving differential equations. *arXiv preprint arXiv:1907.04502*
10. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*
11. Balsa-Canto E, Alonso AA, Banga JR (2008) Computational procedures for optimal experimental design in biological systems. *IET Syst Biol* 2(4):163–172
12. Foo J, Sindi S, Karniadakis GE (2009) Multi-element probabilistic collocation for sensitivity analysis in cellular signalling networks. *IET Syst Biol* 3(4):239–254



Chapter 5

A Practical Guide to Reproducible Modeling for Biochemical Networks

Veronica L. Porubsky and Herbert M. Sauro

Abstract

While scientific disciplines revere reproducibility, many studies – experimental and computational alike – fall short of this ideal and cannot be reproduced or even repeated when the model is shared. For computational modeling of biochemical networks, there is a dearth of formal training and resources available describing how to practically implement reproducible methods, despite a wealth of existing tools and formats which could be used to support reproducibility. This chapter points the reader to useful software tools and standardized formats that support reproducible modeling of biochemical networks and provides suggestions on how to implement reproducible methods in practice. Many of the suggestions encourage readers to use best practices from the software development community in order to automate, test, and version control their model components. A Jupyter Notebook demonstrating several of the key steps in building a reproducible biochemical network model is included to supplement the recommendations in the text.

Key words Reproducibility, Biochemical networks, Computational modeling, Modeling software, SBML, SED-ML, FAIR, COMBINE archive

1 Introduction

The use of computational modeling to understand and interrogate biochemical systems is increasing as the field expands and garners interest in the potential predictive power and efficiency offered by performing experimentation in silico. However, with more widespread use and the ability to build larger, more complex models from existing components, it is not a simple task to implement models which can be reproduced. We are facing a longstanding reproducibility crisis in both experimental and computational research [1–6], and computational models of biochemical networks have not escaped this challenge [7–13]. In a study performed by the European Bioinformatics Institute using 455 kinetic models

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-1-0716-3008-2_5.

published in peer-reviewed journals and stored in the BioModels database, the researchers found that half of the models could not be reproduced [14]. Many of these models are probably similar in their theoretical, mathematical, and computational implementations to the biochemical network models of interest to the readers of this chapter. Unfortunately, while reproducibility has been lauded as a governing principle for all scientific disciplines, it has not been well-rewarded in practice and is rarely taught in conjunction with the theory required for biological modeling. This appears to be changing – increasingly, academic journals like PLOS Computational Biology are providing incentives for researchers who take the extra steps to engage in reproducible practices and submit computational models that can be independently reproduced to spend the time required for this endeavor, awarding badges that certify the degree to which a study could be reproduced [15].

We will define reproducibility in terms similar to those used by the Association for Computing Machinery [16]. Reproducibility is accomplished when a modeler creates *de novo* some subset of the materials used in a published modeling study. This could include generating new datasets, modifying the mathematical model describing how components of the biochemical system interact, using distinct software to simulate the model, or including new analyses. As a result, it can be expected that reproducing a modeling study without reusing any of the existing components will be a more extensive endeavor than modifying one component or a small subset of the model, software, or data in the original study. Repeatability is also important to consider. It involves simply repeating the study with the original data, software, and hardware. This requires that the modeling study is distributed in full and that it is readily accessible. The simplest way to ensure repeatability is to distribute the study in a virtual machine or container, which will provide all materials – including software and hardware dependencies – together. It may be possible to repeat a study with some differences in software dependencies and hardware, but often repeatability is impeded by a failure to make the model, software, and data available when the study is published.

There are many advantages to building models that are reproducible, including making a model that can be readily understood, modified, and reused by other researchers. Additionally, we can have greater confidence in the conclusions drawn from studies performed using a reproducible model, because they will have undergone extensive verification and validation, and can be interrogated by other modelers. Together, these advantages ensure that the modeling community can collaboratively build and expand upon reproducible models and share knowledge and resources readily. If all data, software, and documentation of the model construction and simulation process are shared under open-source licenses, and standardized formats, ontologies, and minimum

information standards are used to guide model building, reproducibility will be readily achieved. With greater community support and availability of the model code and data, larger models can be constructed from the component pieces. These more complex models will have extensive applications and allow us to understand complicated biological processes and augment the knowledge discovered through experimental methods.

While the greater modeling community has developed a multitude of tools, libraries, and standardized formats to assist modelers, it can be overwhelming to execute a modeling study while considering the added constraint of reproducible practices and integration of unfamiliar software development workflows. In this chapter, we provide useful tools and practices to assist biochemical network modelers in executing reproducible workflows. While the tasks which can be included in a modeling workflow are vast and the formalisms for representing a biochemical network mathematically are diverse, we will cover the core stages which are common to most modeling studies. A subset of the major milestones in creating a reproducible modeling study which is repeated and reproduced by an independent investigator is illustrated in Fig. 1. These methods can be practiced by researchers at any stage in their modeling career and can be implemented individually in order to incrementally improve the reproducibility of the study. A subset of the recommendations provided in this chapter will include simple examples which a modeler could perform with minimal experience, using Python and libraries created to support modeling of biochemical networks in Python. Each example will build on the previous analyses using the repressor circuit model constructed by Elowitz and Liebler [17], a synthetic biochemical network using transcriptional regulation to produce oscillatory dynamics, viewed as the birth of systems biology. These will give readers a tangible, practical idea of how to execute the recommended strategies in the stages of data collection, model construction, parameter estimation, simulation, testing, packaging, and publishing. These examples are shown in the supplemental files distributed with this chapter which include a static HTML version of the Jupyter Notebook used for each example, along with access to a GitHub repository containing the Jupyter Notebook, supplemental files – with functions to execute parameter estimation and visualization, and instructions for using the notebook. These are available to download and run at https://github.com/vporubsky/MiMB_reproducible_biomodeling.

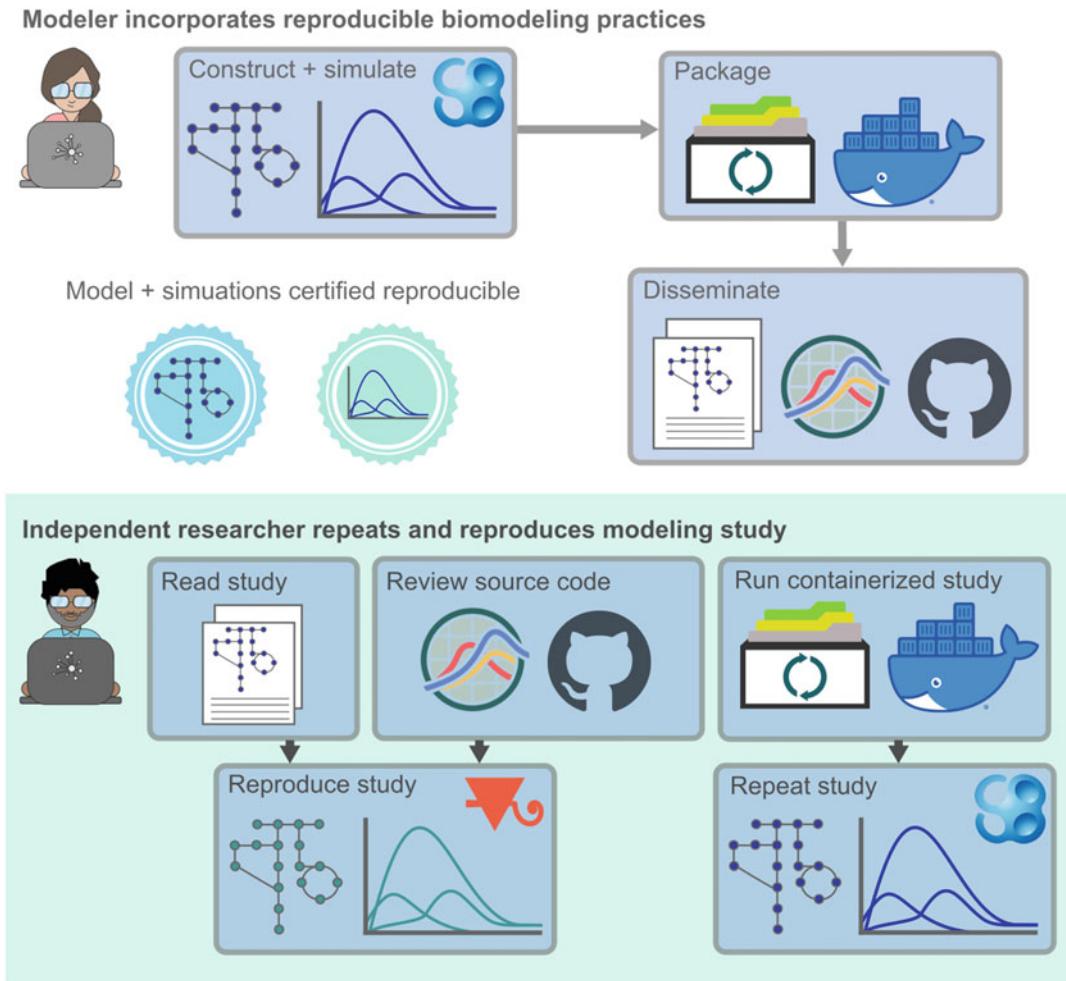


Fig. 1 Overview of a reproducible modeling study. Illustration of key steps in the process of reproducible modeling for biochemical networks. In the top panel, a modeler produces an original research study using best practices, including using standardized formats like SBML and the COMBINE archive, distributing a self-contained modeling study to include all software dependencies with Docker, and publishing their model and software in the BioModels database and GitHub. An independent researcher repeats the original study by running the COMBINE archive or Docker container. The researcher can also reproduce the study by reading the published study results, reviewing the model and source code, and creating a new model or using distinct tools to investigate the same biochemical network and questions posed by the original study

2 Data Collection

Once a modeler has chosen a biochemical network of interest to model, and identified the biochemical species and interactions within the network, one of the first steps is to collect relevant data to ensure their model adequately represents the underlying biology. This may involve running new experiments to generate data, harvesting values from existing literature, or importing data from

online data sources – collectively, these acts of curating data relevant to a modeling workflow contribute to data collection. It is often necessary to collect data in collaboration with experimentalists, as it will ensure that the data is relevant to inform the modeling study and in a form that can be used to validate the model predictions. The conditions needed for a particular investigation will likely not be readily available. Whether experiments are used to generate new data for explicit integration into a biochemical modeling study or data is gathered from existing databases, modelers should share their data collection methods and the origin of the experimental data used to create their model. Ontologies, which share the precise relationship between concepts in a given scientific domain, and minimum information standards have been well-developed for experimental methods useful in modeling biochemical networks and should be followed whenever possible [18–24]. Modelers planning to perform additional experiments for their modeling study can find many useful protocols in the *Methods in Molecular Biology* series.

2.1 Databases for Modeling Biochemical Networks

There are numerous databases containing intracellular biochemical data that may be used to model biochemical networks. These include BioCyc, BRENDA, ChEBI, KEGG, and KEGG Pathway in particular, the RCSB PDB, Reactome, Rhea, SABIO-RK, UniMod, and UniProt. We will provide a brief description of these databases, but a more complete list of useful data sources is available in Appendix A of Goldberg et al. [25].

BioCyc (<https://biocyc.org/>) is a collection of genome and metabolic pathway databases and can be used to find data sources specific to a biological pathway of interest [26]. The database includes regulatory networks, omics data, and metabolic modeling information which expands on the standard gene and metabolic network data. In addition to the pathway information, BioCyc provides functionality to visualize groups of related pathways and hosts a tool called SmartTables which can be used to define groups of genes, metabolites, and pathways of interest without requiring extensive knowledge of programming. This feature allows the user to easily browse lists of objects, which could expedite the data collection process.

BRENDA (<https://www.brenda-enzymes.org/index.php>) contains the largest collection of data relevant to enzyme interactions, including kinetic data, protein, and genome sequences [27]. These data can be queried by taxonomic or disease labels, among other methods. The database undergoes continuous curation and updates to include new enzyme data, and this curation involves text and data mining from primary literature. Pathway maps can be visualized directly on the BRENDA database website to help modelers understand the greater biological context surrounding an enzymatic reaction.

Chemical Entities of Biological Interest (ChEBI) (<https://www.ebi.ac.uk/chebi/>) represents both a database and an ontology which describes chemical entities in terms of their shared structural features and their role in biochemical systems, which makes it a useful resource in modeling biochemical networks [28]. ChEBI is a collection of primarily small chemical compounds. The database contains a vast amount of fully curated data entries and largely covers metabolites in a range of species from *Escherichia coli* to *Homo sapiens*. ChEBI identifiers are standard identifiers that are useful for annotating biochemical models.

KEGG and KEGG Pathway (<https://www.genome.jp/kegg/>), from the Kyoto Encyclopedia of Genes and Genomes, provide genomic information [29]. In the case of KEGG pathway, graphical depictions of processes like metabolism, trafficking across membranes, and signal transduction describe how higher order functional information is linked to the underlying genetic information. This can be particularly useful for identifying conserved pathways that may be used to inform a model. The database standardizes gene annotations, and KEGG identifiers may also be useful in annotating biochemical network models.

RCSB PDB (<https://www.rcsb.org/>) provides structural data about macromolecules obtained from X-ray crystallography, NMR, and cryoelectron microscopy [30]. It would be most useful to modelers interested in binding to specific domains or modeling a small subset of proteins with a high degree of molecular detail.

Reactome (<https://reactome.org/>) contains metabolic maps that are stored as data models and can be used to understand reactions in several biochemical processes [31]. Signal transduction, transport, metabolism, and cellular processes are all represented in the Reactome database. The database also provides an annotation model which allows domain experts to explicitly identify processes and the subjects involved in those processes.

Rhea (<https://www.rhea-db.org/>) is a database of curated biochemical reactions which uses the ChEBI ontology to identify the chemical species and employs stoichiometric balance [32]. The entries contained in the database include those relevant to enzyme-catalyzed and transport reactions and are applicable for reconstructing metabolic networks and other pathways of interest. The curation of the reaction entries ensures that each reaction is linked to the original publications containing the reaction and source data and also includes entries to link to enzyme and pathway databases that use the reaction or enzyme.

SABIO-RK (<http://sabio.h-its.org/>) contains kinetic rate laws extracted from literature [33]. For each entry in the database, the rate equation, parameter values, and environmental conditions during measurement of the kinetics for the reaction are all recorded. SABIO-RK has documentation to guide through programmatic accessing of the database.

UniProt (<https://www.uniprot.org/>) is a database of protein sequences which contain annotations about the function of domains in the sequence [34]. Similarly, UniMod (<https://unimod.org/>) is a database which contains protein modifications that can be identified through mass spectrometry and holds data with precise measurements to identify both natural and artificial modifications [35].

2.2 Metadata Management

Metadata in biochemical network modeling are data that describe data [36]. When collecting data through new experiments or by curating measurements available in online databases, it is important to also provide a description about the data to ensure that other researchers can understand the components of your model. Information about statistical analyses applied to data after collection and a description of the estimated uncertainty found in measurements as a result of noise is critical when evaluating confidence in the model predictions and should be included [37, 38]. Include metadata about a measurement such as the units, accuracy estimates, and any additional annotations. These will provide information to help other modelers decide if data can be used for their model. It is also important to provide the historical record of data, such as the lab that generated the data, the environmental conditions under which it was obtained, the protocol used to make it, the paper that originally published the data, and the online data source where it was collected [36].

2.3 Automated Data Collection from Online Databases

A key feature to improve reproducibility during data collection, and which will be a common theme in several of the tasks to follow, is to automate data collection and harvesting. The extent to which human error in recording and entering data manually prohibits reproducibility cannot be understated. This aside, programmatically collecting relevant data, can improve our metadata management. A useful tool in this endeavor is the BioServices Python package, which allows many of the databases described above to be queried programmatically [39]. Figure 2a shows an example workflow for collecting data and metadata from the KEGG database programmatically. An example query to perform data collection with BioServices from the KEGG database is also demonstrated in Fig. 2b. When possible, data should be directly imported into a modeling study, metadata stored, and no manual manipulation of the data performed. If data is altered – for example, if a dataset is represented using a single statistical metric or a subset of the original data, the code used to derive this value should be provided. To store experimental data for modeling building in a structured way, SBtab [40] and ObjTables [41] provide a set of rules and conventions to organize table-based data formats. Such tabular data storage can facilitate automatic programmatic integration in the model by indexing the table.

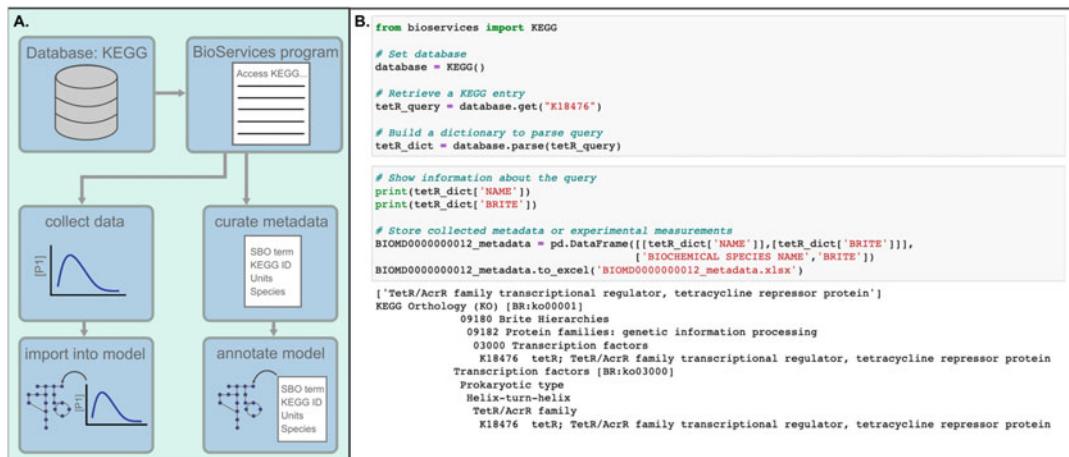


Fig. 2 Data collection. (a) A workflow is shown to demonstrate how BioServices can be used to programmatically collect data for model construction, or to collect curated metadata which describes the measurements to be used for annotating the model. (b) An example using BioServices to collect metadata about the TetR transcriptional regulator, tetracycline repressor protein, is shown. The KEGG database is accessed to get the KEGG orthology entry for the biochemical species with identifier K18476. The name of the biochemical species and the BRITE hierarchical classification are collected from the KEGG database, printed, and stored in an Excel spreadsheet

3 Model Construction

Constructing a model involves theoretically, mathematically, and computationally implementing a representation of the biological system and environmental context of interest. This includes encoding the species involved in the system and their interactions. We will focus on dynamic models which have biochemical reactions that evolve as a function of time. Such models require rate laws for the biochemical reactions, explicit initial conditions, and values for fixed parameters like rate constants that fully determine the underlying network dynamics.

3.1 Structured Model Description Formats

Each model should be represented with a structured format that ensures all model components and their relationships with each other can be understood. An effective format that modelers should consider is the systems biology markup language (SBML) [42], as this has been heavily developed and is a well-supported standard within the systems biology modeling community. Multiple standardized model description formats have been developed to support exchange and formal representation of biochemical models. The SBML format describes models as a list of transformations that occur to the biochemical species in the network. There are defined elements that represent spatial compartments in the model, the individual biochemical species, chemical transformations and

reactions, and parameter values for these reactions. SBML provides rules to add constraints to the model and to perform general mathematical computations. SBML is not easily understood by humans, and therefore must be read, written, and interpreted for visualization or analysis using software which supports the standard. The Antimony script language [43] allows modelers to easily express and understand models in a reaction scheme format using a simple text-based description. Tellurium [44] is a modeling tool which fully supports the use of Antimony and can be used to import, export, and convert SBML between the human-readable and machine-readable format. There are alternative structure formats if SBML does not offer the necessary functionality for a given modeling study. BioPAX is a modeling language that can be used to represent biological pathways and export them to SBML or CellML [45], while BioNetGen [46] and PySB [47] are used to generate rule-based models and can also export to SBML.

Models can also be described by writing mathematical relationships in Matlab, Python, or other programming languages. This does not provide infrastructure to make the relationships between model components clear, so it is necessary to document the code to describe them and to take precautions to ensure that components and values are being manipulated programmatically rather than through manual updates. Describe the model components in a data format, for example, create a computer-readable table with a column for each molecular attribute in the model so that the information about that biochemical species can be queried with code. The SEEK platform for data and model management can be used along with RightField for annotating tabular data [48, 49] which might still be used in general-purpose programming with Python or Matlab. Standard identifiers for biological and chemical species can still be used to name the model components in this framework. We also provide an example of using Tellurium to convert between Antimony and SBML model descriptions in the associated Jupyter Notebook (*see* supplemental files).

3.2 Understanding the System Through Visualization

It may be useful to generate a visual map of the system, which will help independent researchers understand the biochemical species and interactions which are described in the model. This can be accomplished with the systems biology graphical notation (SBGN) [50]. There are multiple software tools that allow modelers to read, write, and edit SBGN and even convert between modeling formats, but as an example of the visualization, we examine a network for the repressilator circuit using libsbgnpy [51], shown in Fig. 3a. This can be especially useful when constructing a model that builds upon existing models. By converting a biochemical network to SBGN representation, the network can be visualized to understand the system, its species, and interactions.

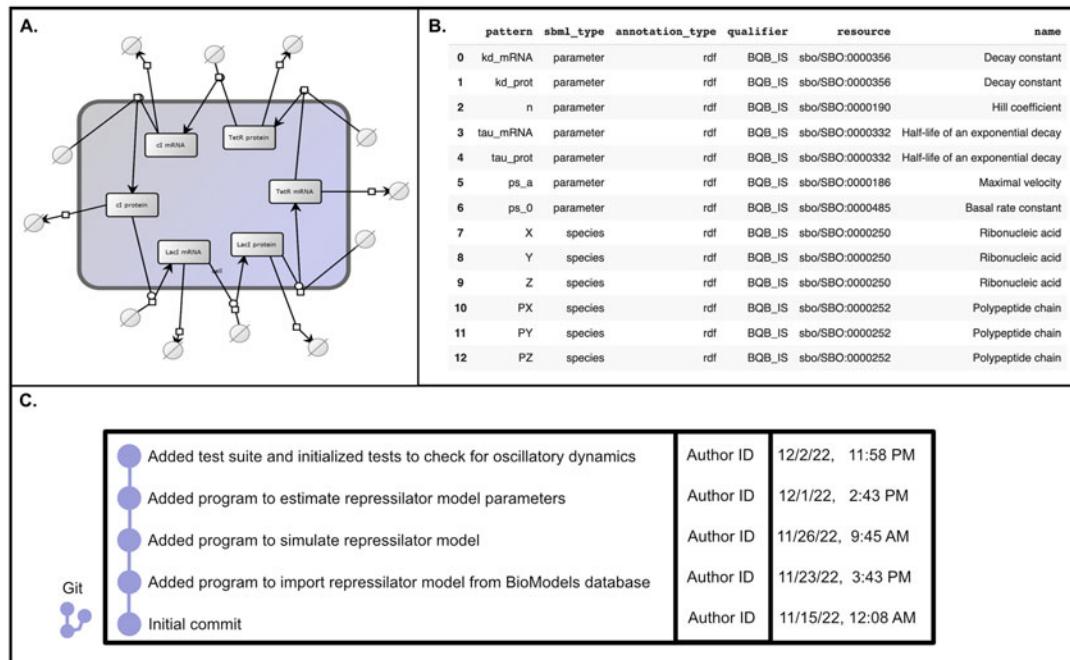


Fig. 3 Model construction. **(a)** An SBGN representation of the repressor model visualized using libsbgnpy shows the topology of the biochemical system, with interactions between each component of the gene regulatory network. This SBGN output is generated using an auto-layout, but could be adjusted manually. **(b)** Annotations to be imported into the SBML model for the repressor modeling study using the sbmlutils model annotator functionality are shown. These annotations are present in the published repressor model (BIOMD0000000012) but are shown here to demonstrate the type of metadata which should be included in annotations to ensure that model elements are unambiguous and identifiable. For example, the SBO identifiers for each biochemical species and parameter can be included in annotations for SBML models. **(c)** An example of version-control is illustrated. This could be accomplished by linking a GitHub repository to an integrated development environment, such as PyCharm. Documentation of changes made for each update are specified through a commit message, and the developer making the change and the time of the update is recorded to ensure that the model development process can be tracked and that earlier versions of the model and software can be revisited

There are conventions for such maps which will be easily interpreted by the greater modeling community and should be implemented. SBGN readily implements these, using standard arrowheads to denote mass transfer and activating interactions, while blunt-end arrowheads indicate repression interactions. Given that automatically generated layouts may not arrange species and interactions in the most intuitive manner or may not be rendered at publication-level quality; it is common to see figures which have been created manually in a graphical editor. In this case, figures of interaction maps which are manually developed to accompany a modeling study should include a detailed legend to describe the species and interactions and should remain consistent with existing conventions. Another standardized method for

visualizing biochemical networks is the SBML layout and render extension, which provides specifications for arranging and rendering the members of a biochemical network in a diagram [52]. This can be used with models encoded in SBML and similarly follows the conventions agreed upon by the community.

3.3 Naming Conventions and Annotation

Another important consideration when constructing a model involves the use of standardized naming conventions and detailed annotations to describe the components of the model. During the model development process, it may happen that a species in the model, such as the Lac repressor protein in the repressilator circuit, is assigned the name “P1.” This name does not carry any biological meaning, and an independent researcher would be unable to identify this species as the Lac repressor protein. It would be better to use a name with biological relevance, such as “Lac_repressor_protein” which makes the species readily identifiable. Still, this can be improved by adding annotations to the model to describe the species more completely. Annotation is the process of adding metadata about the biochemical species, rate law, parameter, data, or other model component to precisely identify the components of a model. Annotations can be added that would describe the identity or function of the “Lac_repressor_protein,” such as standard identifiers from a database such as ChEBI or the PDB. For example, the Lac repressor protein has UniProt ID: P03023. This could be included in the annotation stage of model construction. Adding this additional information ensures that models and their components can be repurposed by deconstructing them into identifiable components. Annotated models that include standard identifiers also ensure that the model can be queried, so that all models containing a given species might be readily discovered. Try to include information about how the model was built, the purpose and biological scope of the model, the identities of the authors, and any assumptions and data sources used in its construction as part of the annotation process. The minimal information required in the annotation of biochemical models (MIRIAM) standard should be considered to ensure all relevant information is included [53]. Use the systems biology ontology (SBO) [54] to categorize the identities of model components using standardized identifiers – this can include assumptions made, the type of rate law for a given reaction, and the roles of species in reactions.

Annotations can be encoded in a variety of ways, but in Fig. 3b and the associated Jupyter Notebook (*see* supplemental files), we demonstrate adding annotations to a the repressilator circuit from an Excel table using the Python package, sbmlutils [55], to add a series of SBO identifiers. The published SBML model for the repressilator circuit is well-annotated, so these annotations are already present in the model, but this demonstrates the process for importing annotations to a standardized format from a table of

stored metadata. This package works exclusively with models encoded in the SBML [56] format. An alternative package which can be used for model annotation is Annotations can be added to SBML using a variety of tools of other tools including libomexmeta [57], Antimony [43], a modeling language distributed with Tellurium [44], and popular modeling tools such as COPASI [58] and JWS Online [59]. For models that are not encoded in SBML, it is common to use the resource description framework (RDF), as it is ideal for representing data that is highly interrelated. It is possible to encode annotations for a model using RDF as a general-purpose solution in various programming environments.

All model components, including the relationships between the components, the biochemical species, and the reactions, should be identified with standard identifiers in the annotated model file. If possible, include information about the biological entity being modeled, including the species, tissue, cell type or strain, and genotype, and the environmental conditions, including the temperature, pressure, and external nutrients surrounding the biological entity.

3.4 Documentation and Versioning

It is useful to keep a log of changes made during the modeling workflow, as models are often incrementally updated, and to provide thorough documentation of design decisions and assumptions which are not explicitly represented in the model. Platforms which support version control with git simplify this process, as changes must often be documented as they are made to the repository, and the differential changes can be easily visualized to understand iterative updates. A commonly used version-control tool is Git. For cloud-based storage of Git repositories that support version control, use GitHub [60]. A simple example of version-control of the GitHub repository from the PyCharm integrated development environment used for this chapter is shown in Fig. 3c. While Git may be new to some modelers, other methods of documenting source code, assumptions, and changes made can be used. A simple method in Python is to provide detailed in-line comments or doc-strings which describe the code. Examples of this are shown throughout the Jupyter Notebook (*see* supplemental files) contained in the associated repository.

4 Simulation

Simulating a biochemical network model is the process of executing the mathematical formulation of the model to generate predictions that allow us to understand how the system behaves. In the examples demonstrated here, the biochemical networks are dynamical systems, and model simulation involves integrating a series of

differential equations to generate predictions of the behavior of the system being modeled as time series data collected at discrete points.

4.1 Writing a Simulation Experiment

We demonstrated how to use SBML to describe biochemical reaction models, with species, compartments, reactions, and rate laws, but this specification does not provide information about which simulation experiments to execute. Instead, a simulator is typically designed to be compatible with the SBML specification and allows the modeler to write simulation experiments. Tellurium [44] provides a programming environment with many utilities to support biochemical network modeling by combining the Antimony language with human-readable and human-writable simulation experiment descriptions through PhraSED-ML [61], and a powerful underlying simulator, libRoadRunner [62]. libRoadRunner can accommodate both deterministic and Gillespie-based stochastic time series simulations, execute steady-state analysis, and generate stability and structural analysis of the stoichiometry matrix and metabolic control analysis, depending on the needs of the modeler [62]. In the associated Jupyter Notebook (*see* supplemental files), we demonstrate how to execute a time series simulation of the repressilator circuit model with Tellurium.

For improved reproducibility, it is important to describe simulation experiments with enough detail that the conditions required for the experiment are explicitly stated. The simulation experiment description markup language (SED-ML) [63] is a standard for biochemical models which accomplishes this task. SED-ML supports time series simulation, steady-state computation, iteration, numerical algorithm specification, and can specify how the results of a simulation should be presented as tabular data or plots. A simple Python syntax for generating SED-ML scripts, a human-readable form of SED-ML, was developed, called phraSED-ML. An example of writing a simulation experiment with Tellurium and phraSED-ML is shown in Fig. 4a, b and demonstrates how a SED-ML file can be exported. Additionally, it should be noted that SBML and SED-ML files can be combined into a directory called the COMBINE archive, which will be discussed in greater detail as a resource for dissemination [64].

4.2 Explicit Representation of Simulation Inputs, Algorithms, and Initial Conditions

The level of detail included in a simulation experiment can vary considerably, but to ensure the experiment is reproducible, include initial conditions and the numerical integration algorithms used. The minimum information about a simulation experiment (MIASE) guidelines can be used to decide what software and data must be included to allow other researchers to reproduce the simulation [65]. SED-ML follows the MIASE guidelines and allows modelers to encode the initial conditions and specify common algorithms with standard identifiers. Adequate annotation of



Fig. 4 Simulation. **(a)** An example of writing a simulation experiment on the repressorator model (saved as an SBML file, BIOMD00000000012.xml) with Tellurium and PhraSED-ML. The experiment is exported as a SED-ML file for portability to other simulators. **(b)** The simulation output for the simulation experiment executed in part A is shown, with the concentration of PX plotted over time. **(c)** Storing results in a structured format with appropriate documentation, or metadata about the simulation data, makes the simulation experiments more reproducible. Here, a simulation is executed using Tellurium and libRoadRunner and the time course for each biochemical species is collected and stored in an HDF5 file. Additional attributes are added to demonstrate how metadata can be included with a dataset using HDF5 – namely, the versions of relevant software, BioModels database ID, and colloquial name for the model system – are included as attributes of the dataset

the simulation experiment can be accomplished by including terms from the kinetic simulation algorithm ontology (KiSAO), which is commonly used to annotate SED-ML documents [66]. The KiSAO can be used to uniquely identify numerical integration algorithms. To ensure that published results can be replicated, all software and data required to produce the original simulation results should be referenced in publications.

Stochastic simulations require additional efforts to enable reproducibility. The random number generator algorithms and seed values used to obtain a stochastic result should be shared to ensure that the exact result can be replicated. To share robust stochastic results, stochastic simulations should be repeated with different seeds to estimate the distribution of possible results. This set of stochastic simulations should be sufficiently large to perform statistical analyses that can characterize the properties of the distribution. Additionally, the seed values used for each of the simulation results could be retained to allow the exact distribution to be regenerated.

4.3 Executing a Simulation Experiment

Once a simulation is described in adequate detail, several simulators have been constructed which are useful for modeling biochemical networks, and which support the import and export of SED-ML. These include COPASI [58, 67], which is the most widely used tool in this domain, created with an accessible graphical user interface (GUI) as well as a command line for more experienced modelers requiring additional analysis capabilities. Java web simulation online (JWS online) provides an online platform to store models, run simulations on those models and any accompanying datasets, and is compatible with standards [59, 68]. libRoadRunner is a simulation library which provides access to multiple numerical integration algorithms and a high-performance simulator [69]. Tellurium is most commonly used to interface with libRoadRunner, providing support for SED-ML and allowing additional analysis beyond simple simulation of the network model [70].

Executing a simulation experiment for a differential equations-based model of a biochemical network involves inputting the initial conditions, parameter values, and any perturbations which will modify the model or the parameter values during the simulation, integrating the differential equations used to define the system, and generating time series which represent how the biochemical species evolve in time. This can be accomplished by writing a program to call the simulator to perform the simulation and may involve repeated calls to the simulator with different initial conditions or perturbations. Ideally, a data file containing the initial conditions and parameters to be used in the model will be created which could be accessed by the program used to run the simulation. This would remove the need for manual manipulation of parameter values. In the examples for the repressilator in the Jupyter Notebook and Fig. 4a, b, the simulation is deterministic, so the additional considerations for stochastic simulations are ignored.

4.4 Recording Simulation Results

Finally, we must consider how to most effectively save simulation results so that they can be shared. Unprocessed simulation results, which in the case of differential equations-based modeling are commonly time series data of concentrations for variable species in the network, should be stored. When generating graphs, tables, or figures for publication, it is useful share the raw data used to generate the graphs as well as the code used to generate them from the raw data. While standardized formats for storing results for computational modeling of biochemical networks have been developed, they have not been widely adopted. However, there are guidelines that can be followed to ensure results are stored appropriately.

The key for storing results and data is to generate structured and space-efficient formats which can be easily interpreted and parsed by code. Common filetypes for storing simulation results include comma-separated values (CSV) or tab-separated values

(TSV) files, Excel spreadsheets, or the hierarchical data format (HDF), which offer structured and efficient data storage that is especially useful for large datasets [71]. RightField is another storage format which provides annotation features to embed labels, especially those containing standardized identifiers, for data stored in Excel spreadsheets [48]. Annotations are useful for identifying entries in the stored results, making RightField spreadsheets a desirable option for reproducible modeling. One platform which manages model data, provides standard-compatibility, and uses RightField spreadsheets for data and model management, is the SEEK platform [72]. SEEK enables the modeler to retain links between the simulation data generated from a model and the model itself, and explicitly identifies data used to construct the model from data used to validate the model [69]. An example of storing simulation results produced with Tellurium in an HDF5 file with annotations added to make the dataset identifiable and transparent is shown in Fig. 4c.

5 Parameter Estimation

Parameter estimation is used when the data collection stage of the modeling study fails to provide a subset of the parameter values needed to fully specify the model, or when the values of some parameters are approximated to fall within some range or probabilistic distribution. Parameter estimation is typically performed by minimizing the difference between experimental measurements of the system and predictions generated using the model and a possible parameter set. An example fit of the repressilator model to synthetic data computed by estimating a subset of parameter values is shown in Fig. 5a. Optimization is performed over a constrained parameter space, continuously altering parameter values until a minimum is reached. Often, there are multiple sets of parameters that minimize the divergence between experimental data and computational predictions equally well. In these cases, the parameters are non-identifiable. It is important to share the methods used for generating the final set of parameters, and we advise you to perform uncertainty quantification and repeated sampling in order to accurately report the distribution of possible parameter values.

5.1 Parameter Estimation Algorithm and Reporting Estimated Parameter Values

Most parameter estimation algorithms use established optimization methods and may involve some manual parameter tuning by the modeler for effective estimation [70]. It is important to use reusable programs instead of manually tuning parameter values whenever possible. These reusable programs can be developed in COPASI [63], SBML-PET [71], PyBioNetFit [78], PESTO [72], AMIGO [77], or SBstoat [73], which are useful for parameter estimation using SBML models. COPASI performs estimation by

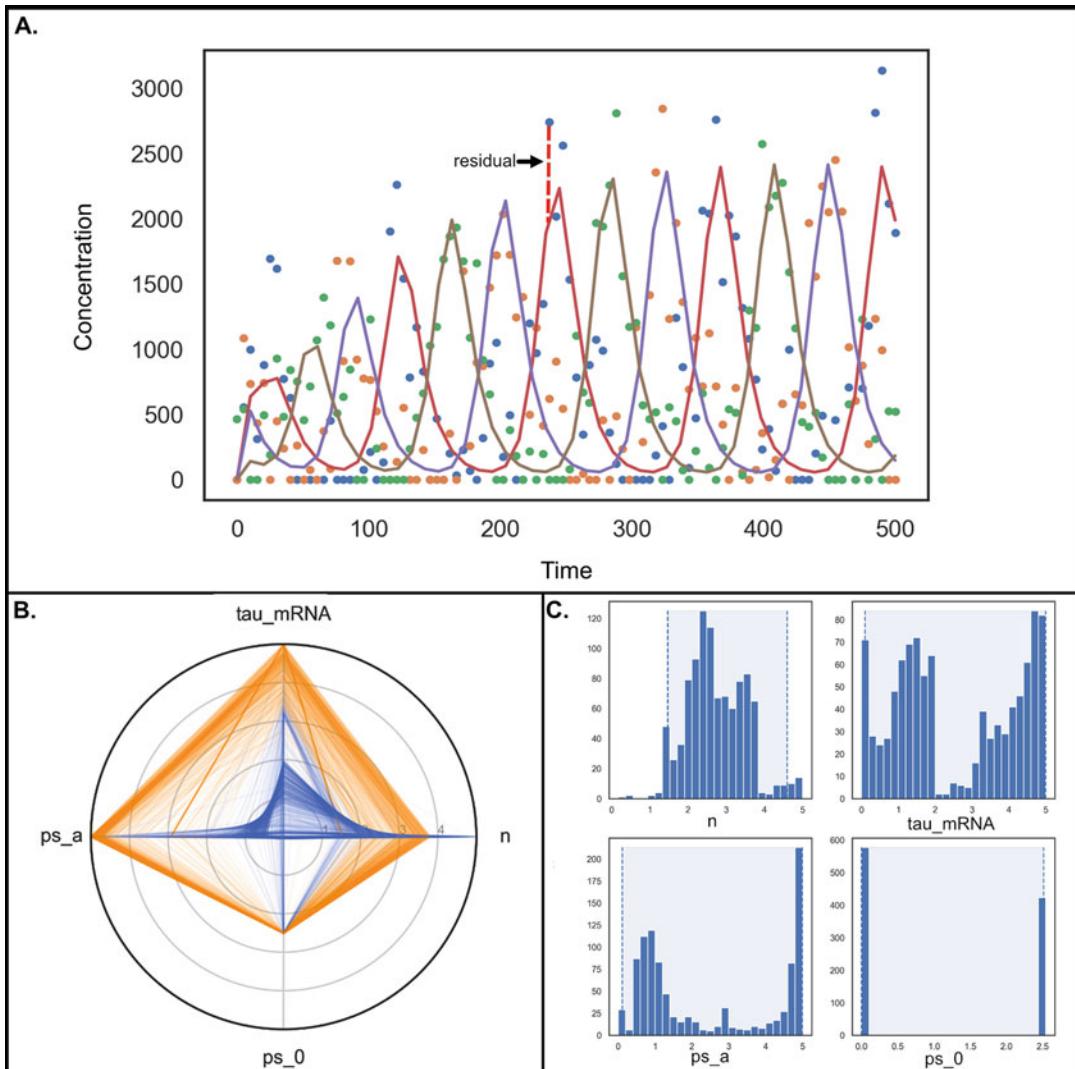


Fig. 5 Parameter estimation. **(a)** An example fit of the repressor protein species to synthetic data. In the example optimization, residuals between the data and simulation are minimized while four parameters (n , ps_a , ps_0 , and τ_{mRNA}) are estimated within user-specified ranges. **(b)** Families of parameters may arise if the optimization is performed multiple times, due to the stochastic nature of the global optimization routine employed in this work. Here, two families or clusters of parameter sets are roughly identified, indicated by the orange and blue traces on the radar plot. It may be difficult to determine which set of parameters is representative of the true system and is therefore useful to report multiple sets of parameters to indicate this uncertainty. **(c)** It is also best practice to perform uncertainty quantification and provide confidence intervals on the estimated parameters. Here, histograms for each of the parameters shows the distributions of values determined by performing a Monte Carlo simulation with 100 iterations. The 95% confidence intervals are shaded in light blue and suggest that the parameter values estimated in this work are not readily identifiable, given the large confidence intervals over the possible range of values. The parameter estimation algorithm, synthetic data generation algorithm, and Monte Carlo algorithm used for this work are included in the GitHub repository associated with this chapter

minimizing the least squares error between time course measurements and the model predictions and offers an additional method called profile likelihood estimation [63]. SBML-PET, PESTO, AMIGO, and SBstoat all estimate parameters for diverse types of experimental measurements. PyBioNetFit provides both parameter estimation and uncertainty quantification functionality [78]. Recently, a useful tool for specifying parameter estimation problems in biochemical network modeling was created, called PEtab, which can be supported by tools as a standardized format [73]. This standard allows the modeler to define their observation model, experimental data, and parameters to be estimated in separate and well-defined TSV files, making PEtab the best option for reproducible parameter estimation. PEtab is supported by several tools that provide parameter estimation capabilities, including COPASI and PESTO [73].

While the use of these parameter estimation tools is growing, it is still common for modelers to write custom code for parameter estimation. For custom programs to perform estimation, state the parameter estimation algorithm and all input values used to update the protocol. If a custom algorithm is created, provide its code and documentation that explicitly states how the algorithm performs the optimization. The optimization library provided by SciPy is especially useful for writing custom Python code for parameter estimation, providing a suite of gradient-based and global optimization methods [74]. Because this approach is so common, a simple example using Tellurium to simulate the model is demonstrated in the Jupyter Notebook (*see* supplemental files) associated with this chapter. This example uses optimization routines from the optimization library in SciPy and adds an additional step to check that the optimized model parameters produce oscillatory dynamics.

5.2 Uncertainty Quantification and Reporting Parameter Distributions with Monte Carlo Sampling

Often, a single optimization will be insufficient to provide reliable parameter values. In these cases, sets of estimated parameters should be reported in machine-readable formats, to identify clusters of similar sets of parameter values and to ensure that uncertainty quantification is performed. An example of these clusters of parameter values is shown in Fig. 5b for the parameters estimated in the repressilator model example. Biochemical measurements are imprecise, and many biochemical properties, such as species concentrations, exhibit natural variability. Uncertainty quantification allows the modeler to estimate the distributions of model input parameters, propagate these distributions through model simulations, and finally to quantify their impacts on model predictions by computing statistics that describe the resulting distribution. It is useful to initialize simulations by sampling inputs from their estimated parameter distributions and execute multiple simulations to obtain the distributions of predictions. All code produced to perform parameter estimation and uncertainty quantification should

be shared alongside the estimated parameter values and results of the uncertainty quantification analysis. Monte Carlo sampling can be used to estimate sets of parameters that satisfy the objective for optimization [81], and confidence intervals can be computed on the distribution for each parameter to determine whether the parameter is identifiable. In the associated Jupyter Notebook (*see* supplemental files), we perform Monte Carlo sampling and generate confidence intervals for four parameters in the repressilator network model. The histograms for these four parameters and their confidence intervals are shown in Fig. 5c.

6 Model Testing

Developing methods to thoroughly test the validity of a model, as well as check that all software associated with the modeling study are implemented correctly, is critical to the reproducibility of the study. Currently, there are not many libraries which can validate the biological “correctness” of a model. This is because many models are domain-specific and require a large amount of expert knowledge in the subject area. There are ongoing efforts to produce test suites relevant to biochemical network models, such as SBviper [75], and these should be consulted. However, these will be limited to testing rules that are broadly consistent across biochemical domains and may not be specialized enough for a given study. In this case, it will be necessary to design and implement a test suite for a new modeling study. Here, we focus on methods commonly used in the software development community and propose possible frameworks for creating and using a test suite. Additionally, to enable replicability, we recommend testing the model and its simulation experiments in an independent computing environment.

6.1 Automated Model Verification and Validation Using a Test Suite

Model verification uses a series of tests to determine whether the model and other software produced for a modeling study are implemented correctly in code, and model validation confirms whether the model behavior is credible when compared to known behavior of the biological system. It is useful to generate a test suite which can check extensively, both by performing static diagnostics on the code through linting and by checking the dynamic behavior of the biochemical model. Creating a test suite makes it easy to repeatedly verify and validate updates to the model, to check that they have not altered the model behavior.

Such test suites will vary considerably depending on the domain and biological question of interest, but some common tests may be relevant for many models across biochemical modeling. For example, you could implement a check for degradation reactions in which there are reactants with no products, but with a constant rate (i.e., the rate is not dependent on the concentration of

the species). If this error is left uncorrected, negative concentrations of the reactant species could result, which is not a meaningful result given that negative concentrations do not exist in the physical world. An example of a test suite for biochemical networks, designed for genome-scale metabolic model checking, is MEMOTE [83]. This test suite provides automatic testing, generates results files and history reports to offer informed feedback to the modeler, and makes it feasible to incrementally update and test these genome-scale metabolic models. We recommend using MEMOTE as a template for generating comprehensive test suites for biochemical networks.

In the case of the repressilator, we expect our time series simulation to generate oscillatory dynamics under the conditions we are interested in, and previously noted that our simulation with estimated parameters did achieve the expected result. In Fig. 6a, we show the result of a simple check on the repressilator model to ensure that the current parameter regime generates complex eigenvalues, which are required for a system to generate oscillatory dynamics. Here, we see that the oscillatory dynamics are lost in a simple time series simulation after an update to the model in the associated Jupyter Notebook (*see* supplemental files). The test result in Fig. 6a shows that the check for complex eigenvalues has failed. This check is implemented using the unittest framework in Python, which is a flexible method for implementing test suites [76]. SciUnit is another framework which can be employed for test-driven model validation which may give some additional structure for biochemical modeling [77]. In this package, modelers write a set of scientific unit tests that compare the model predictions with experimental measurements of the system, and SciUnit executes the tests. A useful tool for statically scanning SBML strings for mass balance errors, common in biochemical modeling, is SBMLLint [75]. An example showing the use of SBMLLint is also included in the test suite produced for the repressilator model in the associated Jupyter Notebook (*see* supplemental files). Figure 6b shows a workflow for iteratively editing and testing the model to ensure that changes to the model do not alter the expected behavior.

6.2 Replication in an Independent Computing Environment to Check Portability

The computing environment used to produce the original modeling study may contain dependencies that are required for proper execution of the model and simulations. A first check to improve reproducibility would be to execute the modeling study using a new computing environment from the one used to produce the study. For example, it may be useful to execute the study on a different operating system, using different versions of the software dependencies, or using a distinct simulator before distributing the study as a check for robustness. Typically, during the process of model construction, modelers will install all software dependencies before beginning the work. It is best to automate the software



Fig. 6 Verification, validation, and packaging. **(a)** The output from an example test implemented using the unittest framework in Python is shown. The time course indicates that the oscillatory dynamics expected for the repressor circuit have been lost in the simulation due to updated parameter values. The test to check for complex eigenvalues has failed, indicating that the parameter update is generating unexpected dynamic behavior which can be automatically identified through the test suite. **(b)** A workflow demonstrating the use of a test suite to enable model checking after updates to the model and associated software is shown. **(c)** After generating a standardized format for representing the model (SBML), simulation (SED-ML), or the full package for the modeling study (COMBINE archive), the simulations can be recreated in a unique simulator. In this case, the COMBINE archive saved in the associated Jupyter notebook was imported into the COPASI GUI. COPASI was used to replicate the simulation, originally performed using Tellurium

installation process and check that this automated installation is portable to other computing environments, given that the dependencies are required for the rest of the modeling study to be executed. In Fig. 6c we show the same simulation reproduced using the COMBINE archive in a distinct simulation tool, COPASI.

7 Packaging and Publishing

Packaging and publishing the model and associated data and software is the final stage in the modeling workflow and could prevent reproducibility without adequate transparency. Standalone packaging, which contains all model components, can improve the ease of sharing the model, data, software, and documentation for the study. There are several model repositories which can be used to distribute the study at the time of publication.

7.1 *Standalone Packaging*

Ideally, when a modeling study is prepared for publication and distribution, it should be self-contained in a package with all data and software necessary to recreate the modeling study. In principle, this will at minimum enable repeatability, as the study should be able to be regenerated in an independent computing environment if all software dependencies (including the appropriate versions) are present. A self-contained solution designed for standardized biochemical models encoded in SBML with simulation experiments encoded in SED-ML is the COMBINE archive (or the OMEX file format) [60]. The COMBINE archive is a zip archive that can contain the model, simulation experiment descriptions, and data, as well as a manifest file to describe the contents of the archive and a metadata file to explicitly identify all components of the archive [60]. Many tools support the COMBINE archive, allowing complete modeling studies to be imported for simulation and analysis by independent researchers. An example of using Tellurium to export a COMBINE archive is provided in the associated Jupyter Notebook (*see* supplemental files). This archive can be imported in new simulator such as COPASI to reproduce the simulations, indicating that the COMBINE archive enables portability, as shown in Fig. 6c. However, the COMBINE archive must be provided to a simulator to repeat a study – therefore the software used to run the study is not contained within the archive. If a container or virtual machine is distributed for the study, all dependencies can be explicitly specified, and it may be possible to rerun the study in a single click to generate the results. Modelers could consider distributing their study in a container or virtual machine to ensure that the simulation experiments can be replicated by independent researchers.

Docker and Amazon machine images are common options for containerized solutions to distribution, while VMware, Parallels, and VirtualBox may be used to construct virtual machines. We will briefly discuss how Docker can be used to generate standalone packaging for a biochemical network modeling study. Docker images can be built such that the operating system, software libraries, applications, and other dependencies are all collected for dissemination [87]. Building a Docker image involves writing a Dockerfile which specifies the parent images (existing Docker images that can be used as a template), sets the working directory, installs all software dependencies, copies files required for the modeling study into the image, and executes the modeling study. For the biochemical network models discussed here, the files which should be copied to the image include the data used by the model (e.g., to inform parameter values or parameter estimation algorithms, specify perturbations in the simulation, etc.), the model description (e.g., an SBML file), the simulation experiments (e.g., SED-ML files), and any analysis programs. In addition, a program which indicates how the study should be executed and how the results will be formatted and visualized should be included. The simulator which will execute the simulation experiments will be installed as a software dependency. For a study performed using python, the command “`pip install tellurium`” could be included in the Dockerfile to provide access to Tellurium and the libRoadRunner library for simulation. The command line in the Dockerfile must specify the programming language (e.g., Python) needed to run the program which will execute the study (e.g., a Python script that loads the SBML and SED-ML files or COMBINE archive and uses the tellurium simulation commands to generate simulation results). Once the Docker image is built, it is robust to changes to files and version updates of software and can be distributed on Docker Hub for others to install and run to repeat the study. An example of a containerized modeling study that uses standardized formats and standard-compatible software is the executable simulation model (EXSIMO) [88], which we recommend exploring to understand how many of the practices described in this chapter can be combined to create a reproducible study.

7.2 Model Repositories

Once the modeling study is complete, and the study is prepared for publication, modelers should find an appropriate place to store the components of their model. Ideally, store the model and all software and data needed to reproduce the study in public, open-source, and version-controlled repositories. With the self-contained preparation described in the previous section, this simply involves depositing the container to an appropriate repository. If it is not feasible to share all software and data, the model should minimally be published with a persistent identifier alongside the research article. This solution will not provide the dependencies to repeat

the study and may not reflect design decisions made, while the model was constructed but could still enable reproducibility if the model is documented with sufficient detail. The key is to minimize any barriers to accessibility, as these will impede efforts of other researchers to reproduce the modeling study. As a result, we advise using an open-source license [89] to publish the model, data, and software. The model study should be shared with a URL or DOI reference to ensure longstanding accessibility. Zenodo is a good resource to provide a readily accessible and persistent identifier for software and data produced in the modeling study.

Using a repository that is version-controlled allows the modeler to record design decisions frequently as updates are documented, and this also allows the modeler to return to previous iterations of the model if a given line of inquiry does not produce meaningful results. We recommend using a version-controlled repository throughout model development, rather than only storing the model, code, and data after the study is complete. When these studies are made publicly available, it provides other researchers with access to design decisions and failed lines of inquiry which may warrant additional exploration. The repository can also reflect model updates made after the publication is complete, which enhance our understanding of the system or the predictive power of the model, illustrating that modeling is performed iteratively and development may be ongoing. GitHub, Bitbucket, and Zenodo are widely used public repositories that support version control and are accessible within the scientific community. GitHub has become a particularly popular platform among modelers and software developers, is integrated with many integrated development environments, and supports collaboration in teams by providing functionality to branch repositories and use continuous integration to merge changes. We recommend using GitHub during development to track changes in your model and interface with collaborators.

While these repositories are very useful during development, it may help to deposit the model in repositories that are frequently used within a particular biological domain. This will increase visibility and reuse within the biochemical network modeling community. The BioModels database is well-established in this domain and offers a curation service to ensure that the models deposited in the repository can be readily implemented and that they generate the results discussed in the published research article [90]. While this already guarantees repeatability in a distinct computational environment (as recommended in the model testing section of this chapter), the curation service also checks that the model follows the MIRIAM guidelines to ensure that there is enough information for the model to be understood [53]. Many of the models provided to the BioModels database are encoded in SBML format, which aids in exchangeability and reuse. We encourage modelers to

deposit their SBML models in the BioModels database. Alternative options include the FAIRDOMHub [91] and JWS Online [59, 68] for sharing model components in a domain-specific repository. JWS Online is an integrated and standards-compliant model storage and simulation platform. It can be used to store model components with data, assumptions, parameter values, and simulation results. The public platform provides access to other researchers, allowing them to execute simulations of shared models directly on the JWS online platform or to download model artifacts. The BiGG Models knowledge base is a good choice for genome-scale metabolic models [92]. The Physiome Model Repository provides an alternative location to publish models with an emphasis on those which are annotated, encouraging reproducible methods [93].

8 Discussion of Guiding Principles

The foundation of many of these practical recommendations for building a reproducible biochemical network model can be summarized by useful guiding principles, namely, the FAIR principles, the use of standardized formats and ontologies for developing models, model versioning, and software engineering principles which emphasize automation, version-control, containerized solutions, and continuous integration.

8.1 FAIR Principles

We encourage modelers to follow the FAIR principles to make their work findable, accessible, interoperable, and reusable [94]. To do this, it is useful to distribute materials publicly with a persistent identifier, as recommended in the packaging and publishing section of this chapter. Additionally, including a summary of the aims of a modeling study, the design and design decisions, all assumptions and limitations of those assumptions, and documentation of the model structure and components will make the model easier to find, understand and reuse. Providing additional documentation to describe how components of the modeling study are related – for example, how dependencies in the data, model, and software are used, and how the predictions of the model result from the computational framework – will make the study more transparent and improve interoperability and reuse.

8.2 Standardized Formats

Many of the practical recommendations provided involve using standardized formats for the model description and simulation experiments, and standardized ontologies for naming, annotation, and documentation. While it may not always be possible to use established standards due to concerns about limited coverage of biochemical processes, limited functionality to create novel mathematical representations or analyses, and with the limited compatibility with existing software, we recommend using the standardized

formats for biochemical research problems that are possible under the specification. Some tasks are more amenable to standardization – for example, ontologies like the SBO or KiSAO can be used to annotate or name model components and make the model more comprehensible, even if the SBML format is not sufficient to represent the interactions of biochemical species in the model.

Incorporating more standardized formats will reduce the effort required to build reproducible models and will also make the model more interoperable and reusable. There are many software tools designed to support the standards described in this chapter, which means that a model constructed in standardized formats can be interpreted, simulated, and analyzed across multiple distinct platforms. This not only serves as a useful method to check that the model is robust through its portability but also increases the likelihood that researchers will be able to modify the published model or use it as a component in a more complex model, given that the components are described in compatible standardized formats. Support for existing standards is growing, with new tools consistently being developed and the coverage of these standards expanding with input from the greater modeling community.

8.3 Model Versioning

In the software development community, versioning software with a common system for naming each version, called semantic versioning, and frequent deployment of updated versions is common practice. This is not typical in the computational biology community – instead, a published modeling study is typically viewed as a completed line of inquiry. The model may be reused or enhanced for a new study but is typically viewed as an independent model, perhaps with a record to show it was inherited or modified from a previous model. There is not a common system for creating model versions, but a future of biochemical network modeling involving such iterative versioning could change how we view modeling. As described in the section on model repositories which described version control, keeping distinct versions of models available for investigation can open new lines of inquiry and necessarily tracks model development with documentation of changes. This model versioning maintains information about model provenance to document how the model has been updated over time and which components of the model may have been inherited from distinct sources. It may be worthwhile to consider the first published version of a model only as a starting point for further construction and investigation.

8.4 Automated Testing, Version Control, Containerized Solutions, and Continuous Integration

As we seek to make our computational models, simulation experiments, and data analysis more robust, it is useful to adopt principles from the software engineering community for systems biology [95, 96]. Testing, verification and validation, version control and iterative development, and containerized solutions can all improve the landscape of biochemical network modeling.

The software engineering industry has a longstanding history of rigorous testing, including unit testing and validation that the system performs as expected. Frequently, these procedures are automated by generating test suites as described in the model testing section of this chapter and calling a program to execute the tests when changes are made and before deploying the model. Full-coverage testing of both the model and the software is a critical procedure in computational modeling of biochemical networks. Version control provides an additional level of security for any changes made to the model and software, ensuring that the changes are documented and that it is possible to revert to earlier versions when an error does result. Thankfully, the software development community has constructed infrastructure through sites like GitHub to easily perform version control even as a beginner in biochemical network modeling.

The containerized solutions described in the packaging section of the chapter also demonstrate the utility of developments made by software engineers. It is uncommon for models to be published along with all the necessary dependencies and data to repeat the study. The practice of distributing all necessary dependencies for tool to run is common in software development, and the full system undergoes stringent checks to ensure that the software can be deployed to distinct computing environments without error. The same should be attempted for biochemical network models, to at minimum ensure that the results reported in publication can be regenerated by independent researchers.

Finally, as the scope of biochemical network models continues to expand, it is likely that teams of researchers will increasingly work together to perform modeling investigations. As a result, tools like continuous integration and iterative model development will become very useful. Continuous integration involves automated merging of the changes from each developer into a single copy of the model or software. It allows for rapid checks to ensure that the code between the two versions is consistent and requires approval to merge changes when there is a discrepancy. This will ensure that teams can seamlessly integrate their changes without introducing significant human error.

9 Conclusions

Reproducible modeling practices may seem like an added constraint on the progress of a modeling study but are well worth the effort, given the understanding that reproducible models give us greater confidence in the predictions made and knowledge acquired from the study. Transparency in the methods used throughout the modeling workflow is paramount, which makes version control, documentation, and publicly sharing all resources used to construct the model essential steps for reproducibility. For biochemical network modeling, the recommendations provided for each core stage of the modeling workflow can guide new modelers through reproducible modeling practices. The tools recommended are good resources to begin each task, and the suggested standards offer a template and structure to ensure the task can be readily understood by other modelers in the field. By following the FAIR principles and engaging in open-source sharing of the resources generated to create a modeling study, we can ensure our work can be accessed, modified, and reused by other researchers. Further, if we take on the rigorous testing and versioning practices which are already common practice in software development communities, we can elevate the reproducibility of work as computational biologists and modelers in the biochemical network domain. If more researchers engage in reproducible modeling practices, we can expand the use of models to accompany experimental methods and expedite discoveries in biology and medicine.

Acknowledgments

The authors would like to acknowledge the members of the Center for Reproducible Biomedical Modeling for their generous support in constructing educational resources like this chapter for dissemination to the greater modeling community. They would also like to thank Dr. Lucian P. Smith and Dr. Joseph L. Hellerstein for fruitful discussion and critical feedback on the computational resources produced for this chapter.

Funding

The content of this publication was supported by the National Institute of Biomedical Imaging and Bioengineering of the National Institutes of Health under award number P41GM109824 and by the National Science Foundation under award number NSF 1933453. It is solely the responsibility of the authors and does not necessarily represent the official views of the

National Institutes of Health, the National Science Foundation, the University of Washington, or the Center for Reproducible Biomedical Modeling.

References

- Mobley A, Linder SK, Braeuer R, Ellis LM, Zwellling L (2013) 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. <https://doi.org/10.1371/journal.pone.0063221>
- Prinz F, Schlange T, Asadullah K (2011) Believe it or not: how much can we rely on published data on potential drug targets? *Nat Rev Drug Discov* 10(9):712–712. <https://doi.org/10.1038/nrd3439-cl>
- Golub TR et al (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 286(5439):531–537. <https://doi.org/10.1126/science.286.5439.531>
- De Schutter E (2010) Data publishing and scientific journals: the future of the scientific paper in a world of shared data. *Neuroinformatics* 8(3):151–153. <https://doi.org/10.1007/s12021-010-9084-8>
- Woelfle M, Olliari P, Todd MH (2011) Open science is a research accelerator. *Nat Chem* 3(10):745–748. <https://doi.org/10.1038/nchem.1149>
- Casadevall A, Fang FC (2010) Reproducible science. *Infect Immun* 78(12):4972–4975. <https://doi.org/10.1128/IAI.00908-10>
- Elofsson A, Hess B, Lindahl E, Onufriev A, van der Spoel D, Wallqvist A (2019) Ten simple rules on how to create open access and reproducible molecular simulations of biological systems. *PLoS Comput Biol* 15(1):e1006649. <https://doi.org/10.1371/journal.pcbi.1006649>
- Sandve GK, Nekrutenko A, Taylor J, Hovig E (2013) Ten simple rules for reproducible computational research. *PLoS Comput Biol* 9(10):1–4. <https://doi.org/10.1371/journal.pcbi.1003285>
- Peng RD (2011) Reproducible research in computational science. *Science* (New York, NY) 334(6060):1226–1227. <https://doi.org/10.1126/science.1213847>
- Medley JK, Goldberg AP, Karr JR (2016) Guidelines for reproducibly building and simulating systems biology models. *IEEE Trans Biomed Eng* 63(10):2015–2020. <https://doi.org/10.1109/TBME.2016.2591960>
- Waltemath D, Wolkenhauer O (2016) How modeling standards, software, and initiatives support reproducibility in systems biology and systems medicine. *IEEE Trans Biomed Eng* 63(10):1999–2006. <https://doi.org/10.1109/TBME.2016.2555481>
- Porubsky VL, Goldberg AP, Rampadarath AK, Nickerson DP, Karr JR, Sauro HM (2020) Best practices for making reproducible biochemical models. *Cell Syst*. <https://doi.org/10.1016/j.cels.2020.06.012>
- Porubsky V, Smith L, Sauro HM (2020) Publishing reproducible dynamic kinetic models. *Brief Bioinform* 22(3). <https://doi.org/10.1093/BIB/BBA152>
- Tiwari K et al (2021) Reproducibility in systems biology modelling. *Mol Syst Biol* 17(2). <https://doi.org/10.15252/MSB.20209982>
- Papin JA, Mac Gabhann F, Sauro HM, Nickerson D, Rampadarath A (2020) Improving reproducibility in computational biology research. *PLoS Comput Biol* 16(5):e1007881. <https://doi.org/10.1371/JOURNAL.PCBI.1007881>
- Association for Computing Machinery (2018) Artifact review and badging. [Online]. Available: <https://www.acm.org/publications/policies/artifact-review-badging>
- Elowitz MB, Leibler S (2000) A synthetic oscillatory network of transcriptional regulators. *Nature* 403(6767):335–338. <https://doi.org/10.1038/35002125>
- Bandrowski A et al (2016) The ontology for biomedical investigations. *PLoS One* 11(4):e0154556. <https://doi.org/10.1371/journal.pone.0154556>
- Kazic T (2015) Ten simple rules for experiments' provenance. *PLoS Comput Biol* 11(10):e1004384. <https://doi.org/10.1371/journal.pcbi.1004384>
- Orchard S et al (2007) The minimum information required for reporting a molecular interaction experiment (MIMIx). *Nat Biotechnol* 25(8):894–898. <https://doi.org/10.1038/nbt1324>
- Deutsch EW et al (2008) Minimum information specification for in situ hybridization and immunohistochemistry experiments (MIS-FISHIE). *Nat Biotechnol* 26(3):305–312. <https://doi.org/10.1038/nbt1391>

22. Bustin SA et al (2009) The MIQE guidelines: minimum information for publication of quantitative real-time PCR experiments. *Clin Chem* 55(4):611–622. <https://doi.org/10.1373/clinchem.2008.112797>
23. Brazma A et al (2001) Minimum information about a microarray experiment (MIAME)—toward standards for microarray data. *Nat Genet* 29(4):365–371. <https://doi.org/10.1038/ng1201-365>
24. Taylor CF et al (2007) The minimum information about a proteomics experiment (MIAPE). *Nat Biotechnol* 25(8):887–893. <https://doi.org/10.1038/nbt1329>
25. Goldberg AP, Szigeti B, Chew YH, Sekar JA, Roth YD, Karr JR (2018) Emerging whole-cell modeling principles and methods. *Curr Opin Biotechnol* 51:97–102. <https://doi.org/10.1016/J.COPBIO.2017.12.013>
26. Karp PD et al (2017) The BioCyc collection of microbial genomes and metabolic pathways. *Brief Bioinform.* <https://doi.org/10.1093/bib/bbx085>
27. Schomburg I, Chang A, Schomburg D (2002) BRENDA, enzyme data and metabolic information. *Nucleic Acids Res* 30(1):47. <https://doi.org/10.1093/NAR/30.1.47>
28. Hastings J et al (2016) ChEBI in 2016: improved services and an expanding collection of metabolites. *Nucleic Acids Res* 44(D1): D1214–D1219. <https://doi.org/10.1093/NAR/GKV1031>
29. Kanehisa M, Goto S (2000) KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res* 28(1):27. <https://doi.org/10.1093/NAR/28.1.27>
30. Berman HM et al (2000) The Protein Data Bank. *Nucleic Acids Res* 28(1):235–242. <https://doi.org/10.1093/NAR/28.1.235>
31. Jassal B et al (2020) The reactome pathway knowledgebase. *Nucleic Acids Res* 48(D1): D498–D503. <https://doi.org/10.1093/NAR/GKZ1031>
32. Alcántara R et al (2012) Rhea—a manually curated resource of biochemical reactions. *Nucleic Acids Res* 40(Database issue):D754. <https://doi.org/10.1093/NAR/GKR1126>
33. Wittig U et al (2012) SABIO-RK – Database for biochemical reaction kinetics. *Nucleic Acids Res* 40:D1. <https://doi.org/10.1093/NAR/GKR1046>
34. Bateman A et al (2021) UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res* 49(D1):D480–D489. <https://doi.org/10.1093/NAR/GKA1100>
35. Creasy DM, Cottrell JS (2004) Unimod: protein modifications for mass spectrometry. *Proteomics* 4(6):1534–1536. <https://doi.org/10.1002/PMIC.200300744>
36. Deelman E, Berrian GB, Chervenak AL, Corcho O, Groth PT, Moreau L (2010) Metadata and provenance management. In: Semantic data management: challenges, technology, and deployment. CRC Press, pp 433–467
37. White GH (2008) Basics of estimating measurement uncertainty. *Clin Biochem Rev* 29 (Suppl 1):S53–S60
38. Mišković L, Hatzimanikatis V (2011) Modeling of uncertainties in biochemical reactions. *Biotechnol Bioeng* 108(2):413–423. <https://doi.org/10.1002/bit.22932>
39. Cokelaer T, Pultz D, Harder LM, Serra-Musach J, Saez-Rodriguez J, Valencia A (2013) BioServices: a common Python package to access biological Web Services programmatically. *Bioinformatics* (Oxford, England) 29(24):3241–3242. <https://doi.org/10.1093/BIOINFORMATICS/BTT547>
40. Lubitz T, Hahn J, Bergmann FT, Noor E, Klipp E, Liebermeister W (2016) SBtab: a flexible table format for data exchange in systems biology. *Bioinformatics* 32(16):2559–2561. <https://doi.org/10.1093/bioinformatics/btw179>
41. Karr JR, Liebermeister W, Goldberg AP, Sekar JAP, Shaikh B (2020) ObjTables: structured spreadsheets that promote data quality, reuse, and integration. *arXiv preprint arXiv:2005.05227*
42. Hucka M et al (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 19(4):524–531. <https://doi.org/10.1093/bioinformatics/btg015>
43. Smith LP, Bergmann FT, Chandran D, Sauro HM (2009) Antimony: a modular model definition language. *Bioinformatics* 25(18): 2452–2454. <https://doi.org/10.1093/bioinformatics/btp401>
44. Choi K et al (2016) Tellurium: a python based modeling and reproducibility platform for systems biology. *bioRxiv:054601*. <https://doi.org/10.1101/054601>
45. Demir E et al (2010) The BioPAX community standard for pathway data sharing. *Nat Biotechnol* 28(9):935–942. <https://doi.org/10.1038/nbt.1666>
46. Harris LA et al (2016) BioNetGen 2.2: advances in rule-based modeling. *Bioinformatics* 32(21):3366–3368. <https://doi.org/10.1093/bioinformatics/btw469>
47. Lopez CF, Muhlich JL, Bachman JA, Sorger PK (2013) Programming biological models in

- Python using PySB. Mol Syst Biol 9:646. <https://doi.org/10.1038/MSB.2013.1>
48. Wolstencroft K et al (2011) RightField: embedding ontology annotation in spreadsheets. Bioinformatics 27(14):2021–2022. <https://doi.org/10.1093/bioinformatics/btr312>
 49. Wolstencroft K et al (2011) The SEEK: A platform for sharing data and models in systems biology. In: Methods in enzymology, vol 500. Academic Press Inc, pp 629–655. <https://doi.org/10.1016/B978-0-12-385118-5.00029-3>
 50. Le Novère N et al (2009) The systems biology graphical notation. Nat Biotechnol 27(8):735–741. <https://doi.org/10.1038/nbt.1558>
 51. König M libsbgn-python documentation. <https://readthedocs.org/projects/libsbgn-python/downloads/pdf/latest/>. Accessed 29 Nov 2021
 52. Bergmann FT, Keating SM, Gauges R, Sahle S, Wengler K (2018) SBML Level 3 package: render, version 1, release 1. J Integr Bioinform 15(1). <https://doi.org/10.1515/jib-2017-0078>
 53. Laibe C, Le Novère N (2007) MIRIAM resources: tools to generate and resolve robust cross-references in systems biology. BMC Syst Biol 1:58. <https://doi.org/10.1186/1752-0509-1-58>
 54. Courtot M et al (2011) Controlled vocabularies and semantics in systems biology. Mol Syst Biol 7(1):543. <https://doi.org/10.1038/msb.2011.77>
 55. GitHub – matthiaskoenig/sbmlutils: Python utilities for SBML. <https://github.com/mattiaskoenig/sbmlutils>. Accessed 05 Dec 2021
 56. Hucka M et al (2018) The Systems Biology Markup Language (SBML): language specification for level 3 version 2 core. J Integr Bioinform 15(1). <https://doi.org/10.1515/jib-2017-0081>
 57. Welsh C, Nickerson DP, Rampadarath A, Neal ML, Sauro HM, Gennari JH (2021) libOmxMeta: enabling semantic annotation of models to support FAIR principles. Bioinformatics (Oxford, England). <https://doi.org/10.1093/BIOINFORMATICS/BTAB445>
 58. Hoops S et al (2006) COPASI--a COmplex PAthway SImulator. Bioinformatics 22(24):3067–3074. <https://doi.org/10.1093/bioinformatics/btl485>
 59. Peters M, Eicher JJ, van Niekerk DD, Waltemath D, Snoep JL (2017) The JWS online simulation database. Bioinformatics: btw831. <https://doi.org/10.1093/bioinformatics/btw831>
 60. Brindescu C, Codoban M, Shmarkatiuk S, Dig D (2014) How do centralized and distributed version control systems impact software changes? <https://doi.org/10.1145/2568225.2568322>
 61. Choi K, Smith LP, Medley JK, Sauro HM (2016) phraSED-ML: a paraphrased, human-readable adaptation of SED-ML. J Bioinform Comput Biol 14(06):1650035. <https://doi.org/10.1142/S0219720016500359>
 62. Somogyi ET et al (2015) libRoadRunner: a high performance SBML simulation and analysis library. Bioinformatics (Oxford, England) 31(20):3315–3321. <https://doi.org/10.1093/bioinformatics/btv363>
 63. Bergmann FT et al (2018) Simulation experiment description markup language (SED-ML) level 1 version 3 (L1V3). J Integr Bioinform 15(1). <https://doi.org/10.1515/jib-2017-0086>
 64. Bergmann FT et al (2014) COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project. BMC Bioinform 15(1):369. <https://doi.org/10.1186/s12859-014-0369-z>
 65. Waltemath D et al (2011) Reproducible computational biology experiments with SED-ML – the simulation experiment description markup language. BMC Syst Biol 5(1):198. <https://doi.org/10.1186/1752-0509-5-198>
 66. Zhukova A, Zhukova A, Waltemath D, Juty N, Laibe C, le Novère N (2011) Kinetic simulation algorithm ontology. Nat Preced. <https://doi.org/10.1038/npre.2011.6330.1>
 67. Bergmann FT et al (2017) COPASI and its applications in biotechnology. J Biotechnol. Elsevier B.V. 261:215–220. <https://doi.org/10.1016/j.jbiotec.2017.06.1200>
 68. Olivier BG, Snoep JL (2004) Web-based kinetic modelling using JWS Online. Bioinformatics 20(13):2143–2144. <https://doi.org/10.1093/bioinformatics/bth200>
 69. Somogyi ET et al (2015) LibRoadRunner: a high performance SBML simulation and analysis library. Bioinformatics 31(20):3315–3321. <https://doi.org/10.1093/bioinformatics/btv363>
 70. Choi K et al (2018) Tellurium: an extensible python-based modeling environment for systems and synthetic biology. Biosystems 171:74–79. <https://doi.org/10.1016/j.biosystems.2018.07.006>
 71. Brown SA, Folk M, Goucher G, Rew R, Dubois PF (1993) Software for portable

- scientific data management. *Comput Phys* 7: 304. <https://doi.org/10.1063/1.4823180>
72. Wolstencroft K et al (2015) SEEK: a systems biology data and model management platform. *BMC Syst Biol* 9(1):33. <https://doi.org/10.1186/s12918-015-0174-y>
 73. Ashyraliyev M, Fomekong-Nanfack Y, Kaandorp JA, Blom JG (2009) Systems biology: parameter estimation for biochemical models. *FEBS J* 276(4):886–902. <https://doi.org/10.1111/j.1742-4658.2008.06844.x>
 74. Zi Z, Klipp E (2006) SBML-PET: a Systems Biology Markup Language-based parameter estimation tool. *Bioinformatics* 22(21): 2704–2705. <https://doi.org/10.1093/bioinformatics/btl443>
 75. Mitra ED et al (2019) PyBioNetFit and the biological property specification language. arXiv. ArXiv ID: 1903.07750
 76. Stapor P et al (2018) PESTO: Parameter ESTimation TOolbox. *Bioinformatics* 34(4): 705–707. <https://doi.org/10.1093/bioinformatics/btx676>
 77. Balsa-Canto E, Banga JR (2011) AMIGO, a toolbox for advanced model identification in systems biology using global optimization. *Bioinformatics* 27(16):2311–2313. <https://doi.org/10.1093/BIOINFORMATICS/BTR370>
 78. GitHub – sys-bio/SBstoat: parameter optimization using Tellurium. <https://github.com/sys-bio/SBstoat>. Accessed 04 Dec 2021
 79. Schmiester L et al (2020) PEtab-interoperable specification of parameter estimation problems in systems biology. *PLoS Comput Biol*. <https://doi.org/10.5281/zenodo.3732958>
 80. Optimization and Root Finding (scipy.optimize) — SciPy v1.3.0 reference guide. [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/optimize.html>
 81. Valderrama-Bahamóndez GI, Fröhlich H (2019) MCMC techniques for parameter estimation of ODE based models in systems biology. *Front Appl Math Stat* 5:55. <https://doi.org/10.3389/FAMS.2019.00055> /BIBTEX
 82. GitHub – sys-bio/SBviper: unit tests for models in systems biology. <https://github.com/sys-bio/SBviper>. Accessed 29 Nov 2021
 83. Lieven C et al (2020) MEMOTE for standardized genome-scale metabolic model testing. *Nat Biotechnol*. <https://doi.org/10.5281/zenodo.2636858>
 84. unittest — Unit testing framework — Python 3.10.0 documentation. <https://docs.python.org/3/library/unittest.html>. Accessed 29 Nov 2021
 85. Omar C, Aldrich J, Gerkin RC (2014) Collaborative infrastructure for test-driven scientific model validation. [Online]. Available: <http://www.cs.cmu.edu/~aldrich/papers/sciunit-icse14.pdf>
 86. GitHub – ModelEngineering/SBMLLint: model checker for SBML compliant models. <https://github.com/ModelEngineering/SBMLLint>. Accessed 29 Nov 2021
 87. Docker: lightweight Linux containers for consistent development and deployment. *Linux J* 2014(239). <https://dl.acm.org/doi/10.5555/2600239.2600241>. Accessed 30 Nov 2021
 88. König M (2020) Executable simulation model of the liver. *bioRxiv*:2020.01.04.894873. <https://doi.org/10.1101/2020.01.04.894873>
 89. Rosen L (2004) Open source licensing. Software freedom and intellectual property law, pp 255–268. [Online]. Available: https://books.google.com/books/about/Open_Source_Licensing.html?id=HGokAQAAIAAJ. Accessed 30 Nov 2021
 90. Li C et al (2010) BioModels database: an enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Syst Biol* 4(1):92. <https://doi.org/10.1186/1752-0509-4-92>
 91. Wolstencroft K et al (2017) FAIRDOMHub: a repository and collaboration environment for sharing systems biology research. *Nucleic Acids Res* 45(D1):D404–D407. <https://doi.org/10.1093/nar/gkw1032>
 92. Norsigian CJ et al (2020) BiGG Models 2020: multi-strain genome-scale models and expansion across the phylogenetic tree. *Nucleic Acids Res* 48(D1):D402–D406. <https://doi.org/10.1093/NAR/GKZ1054>
 93. Sarwar DM et al (2019) Model annotation and discovery with the Physiome Model Repository. *BMC Bioinform* 20(1):1–10. <https://doi.org/10.1186/S12859-019-2987-Y-FIGURES/5>
 94. Wilkinson MD et al (2016) The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* 3:160018. <https://doi.org/10.1038/sdata.2016.18>
 95. Goldberg AP, Szigeti B, Chew YH, Sekar JAP, Roth YD, Karr JR (2018) Emerging whole-cell modeling principles and methods. *Curr Opin Biotechnol*. Elsevier Ltd. 51:97–102. <https://doi.org/10.1016/j.copbio.2017.12.013>
 96. Hellerstein JL, Gu S, Choi K, Sauro HM (2019) Recent advances in biomedical simulations: a manifesto for model engineering. *F1000Research* 8. <https://doi.org/10.12688/F1000RESEARCH.15997.1>



Chapter 6

Integrating Multi-Omics Data to Construct Reliable Interconnected Models of Signaling, Gene Regulatory, and Metabolic Pathways

Krishna Kumar, Debaleena Bhowmik, Sapan Mandloi, Anupam Gautam, Abhishake Lahiri, Nupur Biswas, Sandip Paul, and Saikat Chakrabarti

Abstract

Alteration of the status of the metabolic enzymes could be a probable way to regulate metabolic reprogramming, which is a critical cellular adaptation mechanism especially for cancer cells. Coordination among biological pathways, such as gene-regulatory, signaling, and metabolic pathways is crucial for regulating metabolic adaptation. Also, incorporation of resident microbial metabolic potential in human body can influence the interplay between the microbiome and the systemic or tissue metabolic environments. Systemic framework for model-based integration of multi-omics data can ultimately improve our understanding of metabolic reprogramming at holistic level. However, the interconnectivity and novel meta-pathway regulatory mechanisms are relatively lesser explored and understood. Hence, we propose a computational protocol that utilizes multi-omics data to identify probable cross-pathway regulatory and protein-protein interaction (PPI) links connecting signaling proteins or transcription factors or miRNAs to metabolic enzymes and their metabolites using network analysis and mathematical modeling. These cross-pathway links were shown to play important roles in metabolic reprogramming in cancer scenarios.

Key words Multi-omics, Microbiome, Metabolic reprogramming, Cross-pathway links, Protein-protein interaction

1 Introduction

Since the establishment of Warburg Effect in 1927 [1, 2], metabolic reprogramming is considered to play important role in cancer formation and progression. Numerous studies have elucidated altered state of metabolic pathways/enzymes such as glycolysis, glutaminolysis, amino acid, sugar and lipid metabolism, etc. in various types of cancers. Many studies have shown the causal roles of these altered metabolic pathways in cancer leading to the establishment of metabolic reprogramming as one of the important hallmarks of cancers [3, 4]. However, it is fair to assume that many more metabolic pathways and/or enzymes might be altered

and play important roles in cancer formation and progression. Hence, a systemic exploration of all the metabolic pathways that could in principle be altered or reprogrammed during cancer progression is required. With the advent of high-throughput experiments, different types of omics data such as genomics, epigenomics, RNA and miRNA transcriptomics, proteomics, phosphoproteomics, metabolomics, lipidomics, and pharmacogenomics are nowadays available from large cohort of cancer patients. Utilization of these multi-omics data in unraveling the hitherto unknown metabolic alterations could be extremely helpful to identify individualistic “precision oncology”-based altered metabolic pathways in cancer patients.

Since, aberrations of signaling and regulatory (transcription factors) genes/proteins are primarily abundant in cancer, it is critical and challenging to investigate the impact of these aberrations on metabolic adaptation in cancer cells. Few studies have already shown the link between metabolic alterations and oncogenic changes in signaling proteins, transcription factors, and miRNAs [5–13]. However, a much more exhaustive excavation of the inter-pathway landscape is due to chart out global connectivity patterns between metabolic enzymes and signaling and other regulatory modules of the cells.

The human microbiota can also produce or transform a wide variety of metabolites, which play a major role in the maintenance of the systemic balance of circulating levels of nutrients and metabolites and can modulate various biological functions including the immune and nervous systems [14, 15]. Several studies have revealed the taxonomic and metabolic alterations either from the cancerous tumors or in the other body sites distinct from the tumors [16–18]. An analytical framework to integrate the microbial taxonomy and metabolic profile thus can elucidate the potential effect of microbiota towards regulation of metabolic fate of the cancer cells. This framework estimates metabolome alterations via reconstructing the metagenome profile from taxonomic shifts associated with pathogenesis.

Here, we are presenting a computational pipeline for development of a platform of cellular meta-interactome networks constituted by genes, miRNA, proteins, and metabolites along with their interconnectivity formed by protein-protein interactions and gene-regulatory interactions. Connectivity between systemic metabolite pool and the resident microbiome diversity within and across cancer patients is also considered. Using this meta-interactome, meta-pathway connectivity network, one can extract cross-pathway links that connects signaling pathway proteins, miRNA, and transcription factor (TF) to metabolic pathway proteins through protein-protein interactors (PPIs). The implementation of a mathematical model-based approach for identification of important cross-pathway links and further, the application of in vitro perturbation

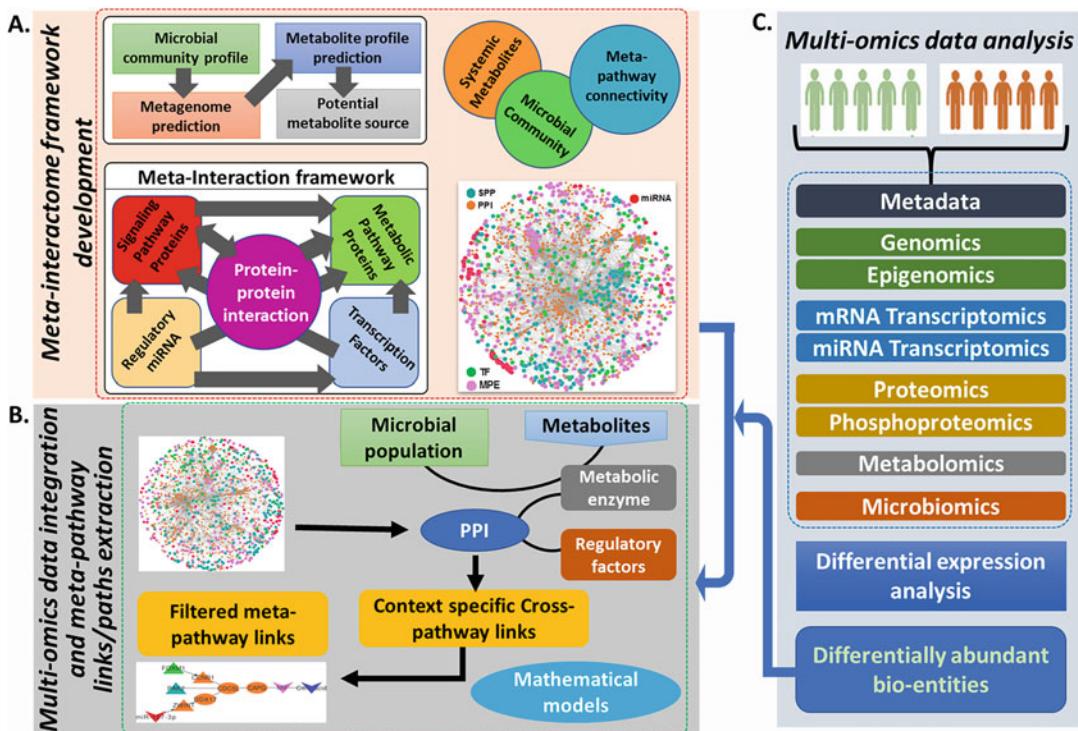


Fig. 1 Overview of the workflow. (a) provides an overview of the concepts for construction of cross-pathway connectivity and associated meta-interaction networks. (b) outlines the multi-omics data processing and subsequent analysis procedures. (c) shows a flowchart of multi-omics and metagenomic data integration and subsequent extraction of critical meta-pathway connectivity links

analysis can find out probable critical signaling or regulatory factors and their associated paths, which in turn may impact the functional status of metabolic enzymes in cancer cells. Figure 1 provides a pictorial overview of our approach, while the following section describes the datasets and the methods that were proposed in our approach.

2 Materials

2.1 Dataset

2.1.1 Dataset for Knowledgebase Meta-Interactome Framework Development

A. Protein-Protein Interaction Data

Structural and functional interaction patterns provide a platform for cross-talk among the bio-entities. Human protein-protein interaction (HPI) data was collected by extracting experimentally verified (score ≥ 700) protein-protein interactions available in the STRING v11.0 [19] database.

B. Pathway Data

Signaling and metabolic pathway information were collected and collated from cellular pathway resources including KEGG [20], Reactome [21, 22], Signalink [23], and NetPath [24] for better coverage of the steps involved in the pathways. Metabolic pathway information datasets were collected from KEGG. Transcriptional regulatory data including transcription factor (TF)-target gene interaction were retrieved from HTRIdb (Human Transcriptional Regulation Interactions database) [25] and TRRUST (Transcriptional Regulatory Relationships Unraveled by Sentence-based Text mining) [26] databases. Post-transcriptional regulatory data consists of miRNA-target gene interactions were retrieved from the mirTarbase [27] and Tarbase [28] databases.

Apart from the structural and functional interaction data, information regarding the connectivity among the data types is also important. Hence, interconnectivity information among the bio-molecules was retrieved from the relevant literature and databases using text and data mining approaches [20–24, 29–33].

2.1.2 Dataset for Multi-Omics Data Analysis

A. Multi-Omics Data

With the help of modern high-throughput technologies, it has become possible to perform global profiling of genes (genomics and epigenomics), proteins (proteomics and phospho-proteomics), RNAs (RNA transcriptomics), miRNAs (miRNA transcriptomics), and metabolites (metabolomics) from the same/similar biological sources. Cellular regulatory mechanisms are distributed across different types of bio-entities (e.g., carbohydrates, lipids, nucleic acids, and proteins) and in different forms (e.g., DNA, RNA, protein, miRNA, metabolites, etc.). Multi-omics analysis provides the path of information flow from one omics layer to other omics layer. Because of the involvement of a large number of entities, these omics data appear as “big” data in the biological context.

One of the primary challenges of multi-omics analysis is the integration of primary omics data and other non-omics biological information. TCGA [34] and ICGA [35] are examples of popular multi-omics data sources.

2.1.3 B. Microbiome Data

The advent of culture-free high-throughput sequencing also paved the exploration of the microbial communities present at different body sites of the human host and understanding their role in human health and disease. Most of the studies conducted the low-cost amplicon-based sequencing using different primer sets in compare to whole shotgun metagenomic sequencing. The primary sequence data generation for any group of individuals for any specific body site or multiple body sites is nowadays a routine process. In this regard the human microbiome project data coordination center (HMPDACC) hosts several human microbiome resources at www.hmpdacc.org. The primary amplicon sequencing

data can further be processed for differential analyses between any groups of individuals for microbial community profiling, predicted functional annotation, and pathway profiling.

3 Methods

3.1 Meta-Interactome Framework Development

3.1.1 Construction of Meta-Interaction Networks

5048 proteins with 18,044 interactions formed the human protein-protein interaction network (HPPIN). Protein interactome network was generated by considering the proteins as nodes and edges as interaction. 22,480 experimentally verified TF-target gene interactions among 697 TFs, and 12,407 target genes were collected to form the transcriptional regulatory network. The TFs and the corresponding target genes were further mapped onto experimentally verified human protein-protein interactions to form TF-TG-PPI network. Similarly, post-transcriptional regulatory network formed by miRNA-target gene interactions were also mapped onto the PPI network. 8407 miRNA-TG interactions formed by 743 miRNA, and 2891 target genes were used to generate the miRNA-TG-PPIN network (Fig. 1a).

3.2 Multi-Omics Data Analysis

3.2.1 Differential Expression Analysis

Differential expression analysis (DEA) is a standard procedure to estimate the quantitative changes of expression of bio-entities (e.g., mRNA, miRNA, proteins, microbial components, metabolites etc.) that are statistically significant between experimental groups. There are different methods available for differential expression analysis such as edgeR [36], DESeq [37], baySeq [38], and EBSeq [39], which are Bayesian approaches based on a negative binomial model. In our case we performed differential expression analysis of each dataset using the processed data available at the Gene Expression Omnibus (GEO) database [40]. Genes having $\log_2\text{FC} \geq +1.5$ and $\log_2\text{FC} \leq -1.5$ were considered as upregulated and downregulated genes, respectively. The genes with $\log_2\text{FC}$ value between -1.5 and $+1.5$ were considered as neutrally expressed genes. The Benjamini-Hochberg method [41] was used to control the false discovery rate. Genes with adjusted p-value ≤ 0.05 were considered as significantly altered. For differential miRNA expression analysis, $\log_2\text{FC} \geq +1.0$ and $\log_2\text{FC} \leq -1.0$ were considered as thresholds for the identification of up and downregulated miRNAs, respectively. The miRNAs with $\log_2\text{FC}$ value between -1.0 and $+1.0$ were considered as neutrally expressed miRNAs (Fig. 1b).

Differential abundance of other types of omics data such as proteomics, phosphoproteomics, genomics, metagenomics, epigenomics, and metabolomics was also integrated. Raw omics data can be processed and filtered using any standard DEA tools. We have used processed format of the omics data where differential expression/abundance of corresponding biomolecules are provided with logFC and threshold probability or p-value for

defining up and downregulation of genes/miRNAs/proteins/metabolites. For genomics, epigenomics, and phosphoproteomics data, genes that are mutated and/or methylated and proteins, which are phosphorylated are considered for further analysis, respectively. Metagenomics data analyses can be performed from raw data to identify differentially abundant microbial components (the detailed analysis method is described in next section) (Fig. 1b).

3.2.2 Microbiome Data Processing and Analysis

Due to the wide usage of the amplicon sequencing, we will discuss here the amplicon sequencing data processing and analysis steps (Fig. 1a). The first objective of amplicon (16S rDNA metagenomics) analysis is to create a feature table from the unprocessed, raw reads generated from the sequencing machine. Illumina is the most prevalent platform, generally with 250×2 or 300×2 paired-end chemistry for short-reads, and the output sequences are in fastq file format. A number of pipelines and tools are available for processing amplicon sequencing data including MEGAN [42], mothur [43], QIIME [44], USEARCH [45], VSEARCH [46], and MICCA [47], among which QIIME and USEARCH were quite popular. The basic steps involved merging paired-end reads, primer trimming, filtering, and finally OTU (operational taxonomic unit) clustering, either closed, open, or de novo. This yielded a final OTU abundance table, usually in the form of a biom file. OTU is a functional representation of closely related microorganisms that could be a species, a group, or at any level of taxonomic hierarchy. But recently an alternative concept emerged called denoising, which was incorporated in QIIME2 [48], one of most widely used pipelines for the downstream processing of amplicon sequencing data, at present, and described ahead. DADA2 [49] is a method which can be used for trimming, filtering, denoising, and chimera removal, and it is built-in QIIME2. The other denoising method available with the QIIME2 pipeline is Deblur [48]. Alongside these, QIIME2 also has VSEARCH incorporated that allows feature clustering (using both de novo and closed methods) and chimeral removal. After successfully processing the raw reads, QIIME2 creates a feature count table, which has the number of reads assigned toward each feature sequence, or ASVs (amplicon sequence variants). Unlike OTUs in clustering methods, denoising results in ASVs, which are basically single DNA sequences and more accurate than OTUs, as they are able to incorporate even a single nucleotide change in the variants. The next step is to assign the taxonomy (at the levels of kingdom, phylum, class, order, family, genus, and species) to the feature sequences or ASVs, and this is where databases come into the picture. Greengenes [50], SILVA [51], and RDP (Ribosomal Database Project) [52] are few 16S-specific databases. For taxonomic classification in QIIME2, first, the database is filtered based upon the 16S region-specific primers (for example V3-V4) with extract-reads; second, database

sequences are fitted with fit-classifier-sklearn (fit-classifier-naive-bayes) and assignment of representative sequences by classify-sklearn [53]. All the three plugins are inherent to QIIME2 in addition to other options for taxonomic classification depending upon the needs of the user. There are other QIIME2-compatible community plugins available, for example, RESCRIPt, which could modify the database and curate taxonomic data, before taxonomic classification of the ASVs [54]. SILVA is generally the most preferred database, but certain downstream analysis of the amplicon sequencing data requires Greengenes-based taxonomic classification exclusively. Further statistical analysis including ANOVA, t-test, etc. at different taxonomic levels can be performed using different packages in R.

3.2.3 Reconstruction of Metagenome Content from Amplicon Data

A predictive metagenomic approach, PICRUSt, is quite well established for the purpose [55]. The prime function of this tool is to infer the functional profile of the microbial community from amplicon sequencing data (Fig. 1a). PICRUSt predicts the ancestral gene content of the microbes using an ancestral state reconstruction algorithm based upon the gene families of their closely related ancestors, while the OTU abundance table is normalized by predicted 16S copy number of each OTU followed by metagenomic function prediction in the “gene content inference step.” Normalized abundance table is then used in the “metagenome inference” step. The first step provides an annotated table of the predicted gene family KO (KEGG orthology) counts for each sample. Subsequently, the next step provides the values representing the extent of contribution by all mapped OTUs to their respective gene families. The final gene count table has the abundance of the genes present in all the metagenomics samples in the OTU table. One limiting factor for using PICRUSt is that it only works on OTUs generated by closed reference OTU-picking (specifically against Greengenes: last updated in 2013). PICRUSt 2.0, introduced in 2019, tries to overcome this issue by incorporating ASV (which has a finer resolution than OTUs) placement on the reference tree made from 20,000 bacterial and archaeal genomes full 16S rRNA genes [56].

3.2.4 Estimation of Community-Wise Metabolic Potential

The community-based metabolite potential (CMP) score is calculated using the tool MIMOSA [57]. The CMP score is a modified extension of PRMT which predicts the community-wise metabolic potential, or the relative capacity of the microbial community to produce or to consume each metabolite [58]. In MIMOSA the input files include a normalized table of predicted abundances of KEGG Orthologs in each sample, a Greengenes-based contribution table with the predicted copy number and abundance of KEGG orthology groups for each feature sequence for each sample and a metadata file describing sample groups. Apart from these, files

describing reactions annotated in KEGG pathway maps, reference information for each KEGG reaction ID, and the linking between KEGG reactions and KEGG genes (KOs) are required for MIMOSA analysis. The tool will finally generate the CMP score matrix for each of the metabolites (or compounds) in the corresponding samples, and much like in PRMT, the CMP score is unit-less (Fig. 1a).

3.2.5 Correlation Analysis Between Predicted and User-Supplied Metabolites

MIMOSA is able to correlate predicted metabolic information with real-time metabolic data (LC-MS, GC-MS feature table). Along with OTU and metadata files, the metabolic features file (with sample IDs in the columns and the compound IDs—KEGG IDs in the row) can be used to look for the compounds or metabolites that are common between the predicted ones and the one obtained after real-time analyses. It then applies the mantel test on the distance matrices of the matched compounds and checks if the predicted ones are correlating positively or negatively with the real-time analyzed one. The tool can also output the potential source of metabolites in terms of gene, reaction, and species (Fig. 1a).

3.3 Multi-Omics Data Integration and Meta-Interaction Network Extraction

A meta-interactome network consisting of human protein-protein interactions, miRNA-target gene regulatory interactions, and transcription factor-target gene regulatory relationships was constructed via superimposition of regulatory networks of transcription factors (TFs) and miRNAs on HPPIN (Fig. 1a). Context-specific interactome networks were extracted from this parent meta-interactome network using the genes, mRNAs, miRNAs, proteins, and metabolites that are either deregulated or altered according to the supplied single or multiple omics data (Fig. 1c). It creates a filtered meta-interactome network comprising of deregulated or altered nodes and their first- or second-level interactors and/or regulators. For metabolomics data, the proteins linked with metabolites are used to integrate the deregulated metabolites (Fig. 1c).

3.3.1 Identification of Topologically Important Nodes (TINs)

Once the context-specific meta-interactome network is formed via utilization of single or multiple omics data, topologically important nodes (TIN), namely, hubs, central nodes (CNs) [59], and bottlenecks (BNs) [60], were identified. To find the important nodes, network and node indices like degree, betweenness, closeness, and clustering coefficients were calculated from the extracted meta-interactome network. These nodes were calculated using our previously reported methods and protocols [59]. For transcriptomics and proteomics data, TINs were identified from the expressed nodes only. For phosphoproteomics, genomics, epigenomics, and metabolomics data, TINs were identified from phosphorylated,

mutated, methylated proteins/genes, and metabolic enzymes, respectively.

Hubs are nodes that have high degrees. The probability distribution function (PDF) of degree z-scores for all nodes in network was plotted, and based on a threshold value of degree, hub nodes were identified. Centrality parameters like, betweenness, closeness, and clustering coefficients were calculated, and the cumulative centrality scores (CCS) were estimated by summing over the combined scores for first layer interactors. Bottleneck nodes were characterized based on their betweenness values.

3.3.2 Metabolic Enzyme Cross-Connecting Paths and Networks

Signaling-metabolic interconnection network was constructed by connecting signaling pathways genes/proteins with the metabolic pathways genes/proteins via protein-protein interactions. All possible unique connections (maximum three proteins involved in between) from a signaling (S) protein to a metabolic pathway protein (M) were established through PPIs (up to second level), considering a signaling pathway protein (S) as a starting point in the HPPIN. Four different types of linking paths were established where signaling proteins were connected to metabolic pathway proteins either directly (S-M) or via one (S-P-M), two (S-P-P-M), or three (S-P-P-P-M) PPIs, respectively. These paths/connections were converted into network to construct signaling-metabolic interaction network (SMIN).

Similarly, transcription factors (TF) and metabolic pathways genes/proteins were connected through TF-target gene interactions and PPIs of TF target genes considering TFs as a source. Similar to signaling-metabolic enzymes connection, different numbers of PPI (up to four) were considered for TF-metabolic links. The resulting paths/links were converted into network to construct TF-metabolic interaction network (TFMIN).

MicroRNA (miR) to metabolic pathway proteins (M) interconnecting paths were established using miRNA-Target gene interactions and PPIs (up to second level) of miRNA target genes considering miRNA as source. NetworkX program [61] was used to form microRNA-metabolic interconnecting network (miRMIN).

3.3.3 Contextualization and Customization of the Cross-Connecting Networks

Deregulated (upregulated and downregulated) and expressed genes and miRNAs were further mapped onto all possible paths/connections to filter context-specific regulatory molecules (signaling pathway proteins, TF, and miRNA) to metabolic enzymes cross-connecting paths. The probability of interactions of a gene/protein with its interactors in the sub-network was determined by using the principle of mass action to define the local entropy of a gene/protein. Normalized expression values of sub-network nodes were used to calculate the interaction probabilities.

Further, node weight of every node was defined based on its biological properties (signaling cross-talk protein (SC) and rate-limiting enzyme (RLE)), differentially expressed gene (DEG), and network topological properties (e.g., hubs, CNs, BNs, etc.).

3.3.4 Identification of Significant Interconnecting Pairs and Paths

Inter-pathway cross-connecting paths were scored using a hidden Markov model (HMM)-based mathematical model established by our group [62, 63]. Two models were used to identify the significant pairs and interconnecting paths, respectively. Edge and node weight of genes/proteins/miRNAs involved in paths were used to calculate the path scores. A threshold of path score was applied to select the significant S/TF/miR-M pairs, whereas paths having path score $\geq 80\%$ of the highest path score for every S/TF/miR-M pair were considered as significant S/TF/miR-M interconnecting paths.

3.3.5 In Silico Perturbation Analysis

To identify the key nodes in the final paths/networks of every module, each of the nodes present in the significant paths/network was removed individually from the HPPIN, and the path score was recalculated for the resulting paths/network by using HMM models. The perturbation score was calculated by using average path score before and after perturbation. Again, a threshold of the average path score deviation (before vs after perturbation) for each perturbed node was applied to select key nodes in significant paths/network.

Figure 1c provides a flowchart of the multi-omics data integration and meta-pathway links/paths extraction.

4 Conclusions

We provide a combinatorial approach and multi-omics data integration and subsequent analysis protocol to unravel the probable means of metabolic alterations that happens during cancer formation and progression. Metabolic alterations within cancer patients may happen due to combination of alteration of the genomic/transcriptomic/proteomic status of the coding and non-coding RNAs together with alteration of the metabolite pool resulted from an altered population of residing microbiome.

Our approach yielding cross-pathway regulatory links to metabolic enzymes is aimed to shed light on the mechanism of altered functional status of the metabolic enzymes in cancer scenario. It is well established that altered transcriptional and post-transcriptional regulations are responsible for mediating the changes in biological processes which ultimately shape in complex patho-physiological situations like cancer. Some of the regulations are perhaps maintained through the systemic coordinated interaction of proteins as a

complex system. Therefore, identification of such protein-interaction network regulating structural and functional status of the interacting partners (enzymes) that are not directly linked is important. The dynamics and flexible nature of the protein-protein interactions might play crucial roles in regulating the interacting partners, and the influence of the alteration of one or more interaction could be transgressed to distal interacting partners in a cascading manner.

In complementation of the above approach, we have developed tools and protocols to analyze the microbial population data collected from cancer and non-cancer patients [64–66]. The primary objective of this approach is to connect the statistically significant differential population of microbes residing within the cancer patients to their altered status of metabolite pool. We hypothesize that the altered microbiome population observed within the cancer patients could influence their altered metabolite pool, which in turn may contribute toward metabolic reprogramming.

References

1. Warburg O (1925) The metabolism of carcinoma cells. *J Cancer Res* 9:148–163
2. Warburg O (1956) On the origin of cancer cells. *Science* 123:309–314
3. Hanahan D, Weinberg RA (2011) Hallmarks of cancer: the next generation. *Cell* 144:646–674
4. Patrick S, Ward CBT (2013) Metabolic reprogramming: a cancer hallmark even Warburg did not anticipate. *Cancer Cell* 21:297–308
5. Ward CBT, Patrick S, Thompson CB (2012) Signaling in control of cell growth and metabolism. *Cold Spring Harb Perspect Biol* 4: a006783
6. Mo Y, Wang Y, Zhang L et al (2019) The role of Wnt signaling pathway in tumor metabolic reprogramming. *J Cancer* 10:3789–3797
7. Papa S, Choy PM, Bubici C (2019) The ERK and JNK pathways in the regulation of metabolic reprogramming. *Oncogene* 38:2223–2240
8. Martin-Martin N, Carracedo A, Torrano V (2017) Metabolism and transcription in cancer: merging two classic tales. *Front Cell Dev Bio* 5:119
9. Dong Y, Tu R, Liu H et al (2020) Regulation of cancer cell metabolism: oncogenic MYC in the driver's seat. *Sig Transduct Target Ther* 5: 124
10. Machida K (2018) Pluripotency transcription factors and metabolic reprogramming of mitochondria in tumor-initiating stem-like cells. *Antioxid Redox Signal* 28:1080–1089
11. Rottiers V, Naar AM (2012) MicroRNAs in metabolism and metabolic disorders. *Nat Rev Mol Cell Biol* 13:239–250
12. Chen B, Li H, Zeng X et al (2012) Roles of microRNA on cancer cell metabolism. *J Transl Med* 10:228
13. Singh PK, Mehla K, Hollingsworth MA et al (2011) Regulation of aerobic glycolysis by microRNAs in cancer. *Mol Cell Pharmacol* 3: 125–134
14. Fung TC, Olson CA, Hsiao EY (2017) Interactions between the microbiota, immune and nervous systems in health and disease. *Nat Neurosci* 20:145–155
15. Zheng D, Liwinski T, Elinav E (2020) Interaction between microbiota and immunity in health and disease. *Cell Res* 30:492–506
16. Gopalakrishnan V, Helmink BA, Spencer CN et al (2018) The influence of the gut microbiome on cancer, immunity, and cancer immunotherapy. *Cancer Cell* 33:570–580
17. Helmink BA, Khan MAW, Hermann A et al (2020) The microbiome, cancer, and cancer therapy. *Nat Med* 25:377–388
18. Xavier JB, Young VB, Skufca J et al (2020) The cancer microbiome: distinguishing direct and indirect effects requires a systemic view. *Trends Cancer* 6:192–204
19. Szklarczyk D, Gable AL, Lyon D et al (2019) STRING v11 : protein –protein association

- networks with increased coverage , supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Res* 47:D607–D613
20. Kanehisa M, Sato Y, Kawashima M et al (2016) KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Res* 44:D457–D462
 21. Fabregat A, Sidiropoulos K, Garapati P et al (2016) The Reactome pathway knowledgebase. *Nucleic Acids Res* 44:D481–D487
 22. Croft D, Mundo AF, Haw R et al (2014) The Reactome pathway knowledgebase. *Nucleic Acids Res* 42:D472–D477
 23. Fazekas D, Koltai M, Türei D et al (2013) SignaLink 2 – a signaling pathway resource with multi-layered regulatory networks. *BMC Syst Biol* 7:7
 24. Kandasamy K, Mohan SS, Raju et al (2010) NetPath: a public resource of curated signal transduction pathways. *Genome Biol* 11:R3
 25. Bovolenta LA, Acencio ML, Lemke N (2012) HTRIdb: an open-access database for experimentally verified human transcriptional regulation interactions. *BMC Genomics* 13:405
 26. Han H, Cho JW, Lee S et al (2018) TRRUST v2: an expanded reference database of human and mouse transcriptional regulatory interactions. *Nucleic Acids Res* 46:D380–D386
 27. Huang HY, Lin YCD, Li J et al (2020) miRTarBase 2020: updates to the experimentally validated microRNA–target interaction database. *Nucleic Acids Res* 48:D148–D154
 28. Karagkouni D, Paraskevopoulou MD, Chatzopoulos S et al (2018) DIANA-TarBase v8:a decade-long collection of experimentally supported miRNA-gene interaction. *Nucleic Acids Res* 46:D239–D245
 29. Mandloi S, Chakrabarti S (2015) PALM-IST: pathway assembly from literature mining--an information search tool. *Sci Rep* 5:10021
 30. Hoffmann R, Valencia A (2004) A gene network for navigating the literature. *Nat Genet* 36:664
 31. Shah PK, Jensen LJ, Boue S, Bork P (2005) Extraction of transcript diversity from scientific literature. *PLoS Comput Biol* 1(1):e10
 32. Horn F, Lau AL, Cohen FE (2004) Automated extraction of mutation data from the literature: application of MuteXt to G protein-coupled receptors and nuclear hormone receptors. *Bioinformatics* 20:557–568
 33. Hu ZZ, Narayanaswamy M, Ravikumar KE, Vijay-Shanker K, Wu CH (2005) Literature mining and database annotation of protein phosphorylation using a rule-based system. *Bioinformatics* 21:2759–2765
 34. <https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga>
 35. Zhang J, Baran J, Cros A et al (2011) International Cancer Genome Consortium Data Portal-a one-stop shop for cancer genomics data. *Database* 2011:bar026
 36. Robinson MD, McCarthy DJ, Smyth GK (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* 26:139–140
 37. Anders S, Huber W (2010) Differential expression analysis for sequence count data. *Genome Biol* 11:R106
 38. Hardcastle TJ (2021) baySeq: Empirical Bayesian analysis of patterns of differential expression in count data. R package version 2.26.0
 39. Leng N, Kendziorski C (2021) EBSeq: an R package for gene and isoform differential expression analysis of RNA-seq data. R package version 1.32.0
 40. Barrett T, Wilhite SE, Ledoux P et al (2013) NCBI GEO: archive for functional genomics data sets--update. *Nucleic Acids Res* 41:D991–D995
 41. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J Royal Stat Soc Ser B* 57:289–300
 42. Huson DH, Auch AF, Qi J et al (2007) MEGAN analysis of metagenomic data. *Genome Res* 17:377–386
 43. Schloss PD, Westcott SL, Ryabin T et al (2009) Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Appl Environ Microbiol* 75:7537–7541
 44. Caporaso JG, Kuczynski J, Stombaugh J et al (2010) QIIME allows analysis of high-throughput community sequencing data. *Nat Methods* 7:335–336
 45. Edgar RC (2010) Search and clustering orders of magnitude faster than BLAST. *Bioinformatics* 26:2460–2461
 46. Rognes T, Flouri T, Nichols B et al (2016) Vsearch: a versatile open source tool for metagenomics. *PeerJ* 4:e2584
 47. Albanese D, Fontana P, De Filippo C et al (2015) MICCA: a complete and accurate software for taxonomic profiling of metagenomic data. *Sci Rep* 5:1–7
 48. Bolyen E, Rideout JR, Dillon MR et al (2019) Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. *Nat Biotechnol* 37:852–857

49. Callahan BJ, McMurdie PJ, Rosen MJ et al (2016) DADA2: high-resolution sample inference from Illumina amplicon data. *Nat Methods* 13:581–583
50. DeSantis TZ, Hugenholtz P, Larsen N et al (2006) Greengenes, a Chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Appl Environ Microbiol* 72: 5069–5072
51. Quast C, Pruesse E, Yilmaz P et al (2013) The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Res* 41:D590–D596
52. Cole JR, Wang Q, Fish JA et al (2014) Ribosomal Database Project: data and tools for high throughput rRNA analysis. *Nucleic Acids Res* 42:D633–D642
53. Pedregosa F, Varoquaux G, Gramfort A et al (2011) Scikit-learn: machine learning in Python. *J Machine Learn Res* 12:2825–2830
54. Robeson MS, O'Rourke DR, Kaehler BD et al (2021) RESCRIPt: reproducible sequence taxonomy reference database management for the masses. *PLoS Comput Biol* e1009581
55. Langille MGI, Zaneveld J, Caporaso JG et al (2013) Predictive functional profiling of microbial communities using 16S rRNA marker gene sequences. *Nat Biotechnol* 31: 814
56. Douglas GM, Maffei VJ, Zaneveld JR et al (2020) PICRUSt2 for prediction of metagenome functions. *Nat Biotechnol* 38:685–688
57. Noecker C, Eng A, Srinivasan S et al (2016) Metabolic model-based integration of microbiome taxonomic and metabolomic profiles elucidates mechanistic links between ecological and metabolic variation. *MSystems* 1:e00013–e00015
58. Larsen PE, Collart FR, Field D et al (2011) Predicted Relative Metabolomic Turnover (PRMT): determining metabolic turnover from a coastal marine metagenomic dataset. *Microb Inform Exp* 1:1–11
59. Bhattacharyya M, Chakrabarti S (2015) Identification of important interacting proteins (IIPs) in plasmodium falciparum using large-scale interaction network analysis and in-silico knock-out studies. *Malar J* 14:1–17
60. Yu H, Kim PM, Sprecher E et al (2007) The importance of bottlenecks in protein networks: correlation with gene essentiality and expression dynamics. *PLoS Comput Biol* 3:713–720
61. Hagberg A, Swart PS, Chult D (2008) Exploring network structure, dynamics, and function using networkx. N. p. Web, USA
62. Bag AK, Mandloi S, Jarmalavicius S et al (2019) Connecting signaling and metabolic pathways in EGF receptor-mediated oncogenesis of glioblastoma. *PLoS Comput Biol* 15: e1007090
63. Kumar K, Bose S, Chakrabarti S (2021) Identification of cross-pathway connections via protein-protein interactions linked to altered states of metabolic enzymes in cervical cancer. *Front Med* 8:1949
64. Biswas N, Kumar K, Bose S et al (2020) Analysis of Pan-Omics Data in Human Interactome Network (APODHIN). *Front Genet* 11: 589231
65. Biswas N, Chakrabarti S (2020) Artificial Intelligence (AI)-based systems biology approaches in multi-omics data analysis of cancer. *Front Oncol* 10:588221
66. Bose S, Kumar K, Chakrabarti S (2021) System biology and network analysis approaches on oxidative stress in cancer. In: Chakraborti S, Ray BK, Roychowdhury S (eds) *Handbook of oxidative stress in cancer: mechanistic aspects*. Springer, Singapore



Chapter 7

Efficient Quantification of Extrinsic Fluctuations via Stochastic Simulations

Tagari Samanta and Sandip Kar

Abstract

At the molecular level, all the biological processes are exposed to fluctuations emanating from various sources in and around the cellular system. Often these fluctuations dictate the outcome of a cell-fate decision-making event. Thus, having an accurate estimate of these fluctuations for any biological network is extremely important. There are well-established theoretical and numerical methods to quantify the intrinsic fluctuation present within a biological network arising due to the low copy numbers of cellular components. Unfortunately, the extrinsic fluctuations arising due to cell division events, epigenetic regulation, etc. have received very little attention. However, recent studies demonstrate that these extrinsic fluctuations significantly affect the transcriptional heterogeneity of certain important genes. Herein, we propose a new stochastic simulation algorithm to efficiently estimate these extrinsic fluctuations for experimentally constructed bidirectional transcriptional reporter systems along with the intrinsic variability. We use the Nanog transcriptional regulatory network and its variants to illustrate our numerical method. Our method reconciled experimental observations related to Nanog transcription, made exciting predictions, and can be applied to quantify intrinsic and extrinsic fluctuations for any similar transcriptional regulatory network.

Key words Stochastic simulation, Intrinsic noise, Extrinsic noise, Computational modeling, Nanog transcription, MicroRNA

1 Introduction

In any living organism, biological processes are normally organized by a complex set of molecular events [1, 2]. These processes happen in a highly precise manner despite withstanding various sources of fluctuations at the cellular level [3]. At the same time, several studies in literature had demonstrated that cellular heterogeneities definitively alter the cell fate decision-making events [4, 5]. To understand how such biological precision is achieved under a fluctuating cellular environment, we need to efficiently quantify the effect of such cellular heterogeneities present in the corresponding system [5]. These cellular heterogeneities are broadly classified into two distinct kinds, namely, intrinsic [6] and extrinsic fluctuations

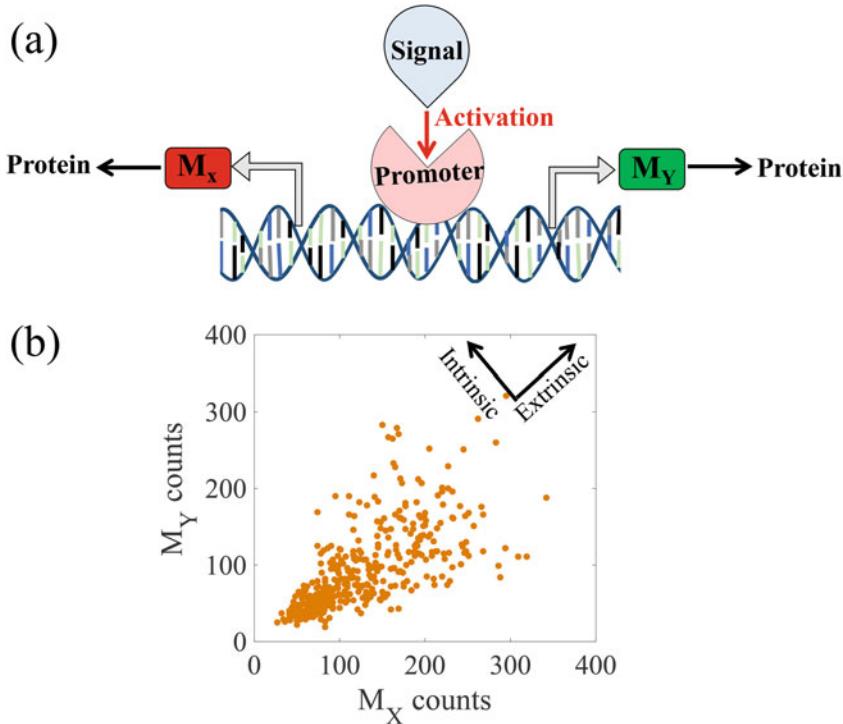


Fig. 1 Schematic representation of the experimental system to quantify the intrinsic and extrinsic noise contribution associated with the transcription process. (a) The external signal-driven bidirectional transcriptional reporter system is used to measure the fluctuation. (b) The scatter plot of the respective transcripts (M_Y and M_X) in different cells (each dot represents a cell within a population of cells). (Adapted with modifications from Elowitz et al. [9] and Ochiai et al. [5])

[7]. The molecular fluctuation due to gene expression variability is commonly known as intrinsic fluctuation. Any other source of fluctuation, other than the intrinsic fluctuation is defined as extrinsic fluctuation, which usually represents the effect of cell-to-cell variability within a cellular population [8].

Experimentally it is quite challenging to quantify the effect of intrinsic and extrinsic fluctuations. Elowitz et al. first developed a protocol [9] to measure these heterogeneities by constructing a bidirectional promoter system (Fig. 1a) in *E. coli* and quantified the intrinsic variability due to the gene expression fluctuations (Fig. 1b). Herein, any other fluctuations other than gene expression noise are considered as the extrinsic fluctuations to the system. A similar kind of system has been developed by Ochiai et al. [5] in a reporter cell line (NMP cell line) to understand the contribution of intrinsic and extrinsic noise sources related to transcription of Nanog (a transcription factor related to stem cell regulation) gene (Fig. 1a). They revealed that at the level of Nanog transcription, the intrinsic and extrinsic noise is partitioned in the ratio of ~45:55 under normal wild-type as well as in different inhibitory culture

conditions (Fig. 1b). This demonstrates that extrinsic sources of fluctuations significantly affect the biological events along with intrinsic variabilities. Theoretical methods [7, 10] have been developed to separately quantify intrinsic and extrinsic fluctuations. However, it turns out to be difficult to segregate the effect of intrinsic and extrinsic fluctuations for a specific biological network.

Intrinsic or molecular fluctuations can be computed effectively for a biological network by simply employing Gillespie's stochastic simulation algorithm (SSA) [10]. Over the year, SSA has been significantly modified to make it more efficient and computationally less expensive [11–13]. However, by employing SSA, it is not possible to accurately measure the various extrinsic variabilities present within a biological system. Among these extrinsic variabilities, cell division events and the related processes associated with the cell cycle regulation can be a major source of fluctuation to any other biological network taking place within the corresponding cell. During the cell division process, the cell can divide unequally [14], the proteins and mRNAs can get distributed equally or unequally [15], transcription rates of any gene vary during different phases of the cell cycle [16, 17], and many other such things can alter the extrinsic variabilities. How to investigate the effect of these extrinsic sources of fluctuations for any biological network without explicitly having a cell cycle regulation model? Can there be a generalized stochastic framework that will deal with the bidirectional promoter systems to effectively segregate the contributions of intrinsic and extrinsic fluctuations?

2 Nanog Transcriptional Regulatory Network as Model System

To address the abovementioned questions, we have taken the help of a simple transcriptional regulatory network of Nanog (a homeobox transcription factor of stem cells) (Fig. 2) to develop a stochastic simulation method based on SSA to quantify the ratio of intrinsic and extrinsic fluctuations for any biological system [18]. We began with a deterministic gene regulatory network of Nanog regulation by considering a bidirectional promoter system of Nanog following Ochiai et al. (Fig. 2a) [5]. We developed the deterministic model of Nanog transcription (including Oct4 and Sox2) in a phenomenological manner [18] with experimentally known degradation rates of proteins and mRNA present in the network and maintained the expression levels of these species as observed experimentally. Ochiai et al. [5] kept the NMP cells under different culture conditions and measured the expression level of the two transcripts of Nanog (with MS2 and PP7 repeats) after 4 h by performing SmFISH (single-molecule fluorescent in situ hybridization) experiment.

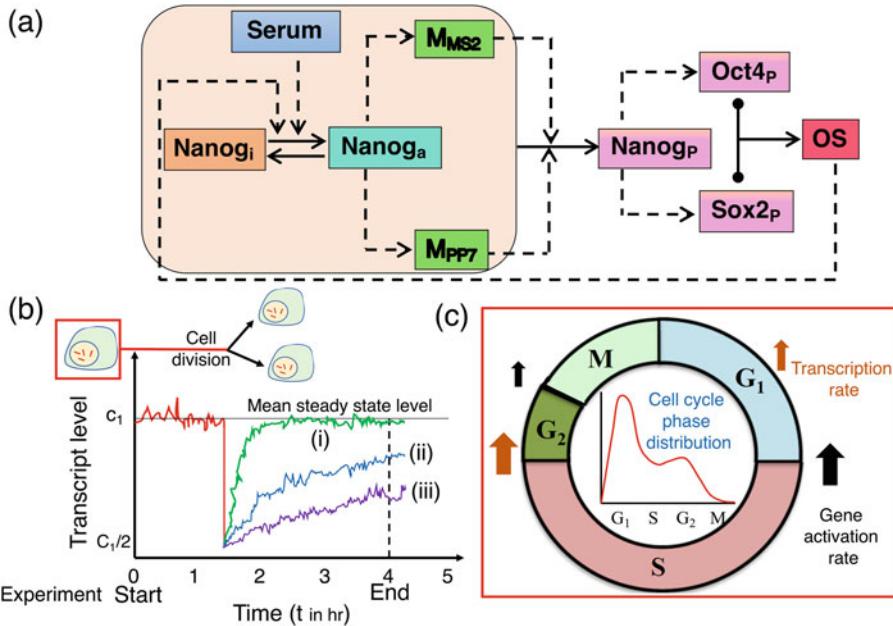


Fig. 2 Nanog transcriptional regulatory network as the model system. (a) The Nanog-Oct4-Sox2 transcription network [18]. The shaded region models how the inactive Nanog promoter (Nanog_i) gets activated (Nanog_a) and leads to either of the Nanog transcripts ($M_{\text{MS}2}$ and $M_{\text{PP}7}$) in the experimental setup. Once Nanog protein (Nanog_P) gets produced from Nanog transcripts, it activates the production of Oct4 and Sox2 proteins (Oct4_P and Sox2_P) which produces a heterodimeric complex (OS) that positively regulates Nanog transcription. (b) Schematic representation of how cell division can create a high degree of extrinsic heterogeneity depending on the inherent time scales following which the transcript comes back around the natural steady-state level after the cell division. In situation (i), the system quickly comes back to the mean steady-state level but for situations (ii), and (iii) the system cannot reach the steady-state level within the experimental observation time, which can lead to higher extrinsic variability. (c) Schematic depiction of the factors related to cell cycle regulation, which influence the transcriptional heterogeneity for the cells under observation

They observed that the ratio of the intrinsic and extrinsic noise at the Nanog transcription level is maintained around ~45:55 under a diverse set of culture conditions [18]. We performed an SSA simulation of our Nanog transcriptional network and examined our simulation results in a manner by which Elowitz et al. [9] analyzed their experimental data by employing the theory of covariance to quantify the extent of intrinsic and extrinsic noises separately. Our analysis failed to reconcile the experimentally observed fluctuation ratio and underestimated the contribution of extrinsic fluctuations. This suggests that the SSA can capture the molecular fluctuations within a biological network adequately, but the contribution of the extrinsic variabilities is not well represented in this method.

In our model, we wanted to avoid an explicit cell cycle network as this will make our analysis more complicated. However, if a biological network and its components are indirectly related to

cell cycle regulation, the cell cycle regulation affects that network and the corresponding components in few specific ways. For example, in the case of the Nanog transcriptional network, where the proteins and mRNAs related to the network are highly stable, perturbation such as cell division happening in between the measurement time can cause a high degree of heterogeneity in the system (Fig. 2b). This is because all the cellular material even if gets distributed equally among the daughter cells during cell division, the system will take a certain time to come back to the original steady states after the cell division event resets the initial levels of various species associated with Nanog regulation [4]. Moreover, these experiments are performed in an asynchronous population of ESCs, where cells are in different phases of the cell cycle [19], and depending on the phase, the transcription rates of various genes vary substantially to produce greater fluctuations in the ESC developmental dynamics [16, 17]. Thus, our simulation method must include these sources of fluctuations in a proper manner.

3 Method

In our method, we extensively modified Gillespie's stochastic simulation and introduced a systematic way to incorporate various extrinsic noise sources [18]. The nonthermal noise sources (extrinsic noise) such as the noise due to equal or non-equal partitioning of cellular material during cell division, the variation of promoter activation and transcription rates during the different phases of the cell cycle, etc. were added in our stochastic simulation algorithm [18]. In this method, we simulated individual cells within an embryonic stem cell (ESC) population under various culture conditions. This allowed us to quantify the ratio of intrinsic and extrinsic fluctuation in Nanog transcription for such ESCs by exactly knowing their next cell division time and the current cell cycle phase in which they were in. In each stochastic simulation run, we wanted to simulate a single-cell present within an asynchronous population of ESCs. In this regard, we need to know two crucial pieces of information about that particular cell: (i) the next cell division time and (ii) the current cell cycle phase of that corresponding cell. Thus, to perform our stochastic simulation, we proceed in the following manner (Fig. 3):

Step 1

We considered that the cell cycle time period of ESCs follows a Gaussian distribution with ~16 h of the mean cell cycle time (for NMP cell line [5]) with a 20% coefficient of variation (CV) in the cell cycle time. We converted a machine-generated uniformly distributed random number to a random number corresponding to this Gaussian distribution. This will eventually allow us to create

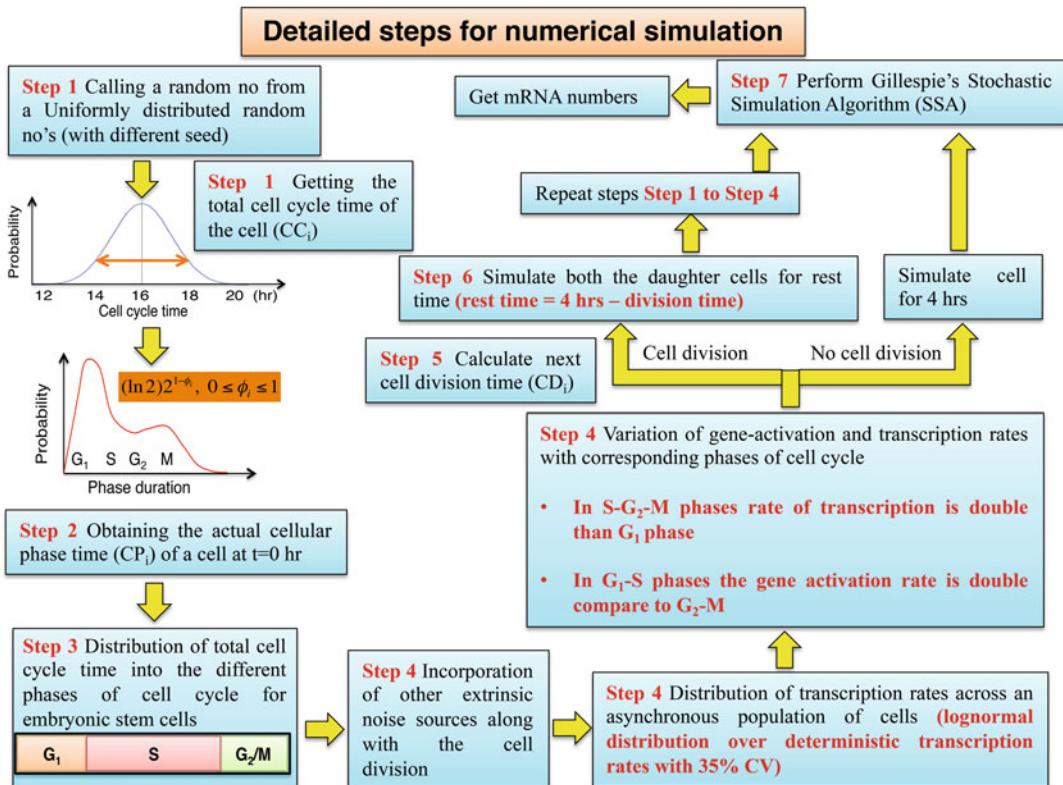


Fig. 3 The detailed steps involved in the proposed stochastic simulation. We have incorporated other extrinsic noise sources along with the cell division. (This figure is taken from the publication Samanta and Kar [18] with copyright permission)

an asynchronous population of cells as per experiments (see Note 1). For each run of stochastic simulation, now we can have a random cell cycle time for a single cell picked from this Gaussian distribution. To begin with, we pick a random cell cycle time for the i th cell (CC_i) in the ESC population.

Step 2

Next, we figure out the current cell cycle phase of the single cell picked from Gaussian cell cycle distribution time with the help of a distribution given as $(\ln 2)2^{1-\varphi_i}$, where φ_i is a random number that varies between 0 and 1 (see Note 2) [19]. Multiplication of φ_i with CC_i gives the current cell cycle phase time (CP_i) for the chosen i th single cell at the beginning of the simulation time (t_0).

Step 3

Depending on the cell-type under consideration (ESCs in this case), we get a rough estimate of the time spent in different cell cycle phases by that i th cell based on the total cell cycle time for it (see Note 3). With the knowledge of CC_i and CP_i , we can get the current cell cycle phase of the i th cell accurately using this step.

Step 4

The gene activation (the conversion rate of genes from its “OFF” (inactive) state to “ON” (active) state) and transcription rates (the transcription of mRNA from its corresponding active gene (“ON” state)) vary a lot during the overall period of the cell cycle (*see Note 4*) [16, 17]. We assumed that the transcription rates of all the network components can be obtained using a log-normal distribution around the mean deterministic transcription rate with a 35% of the coefficient of variation (for ESCs) [20]. We further modified these gene activation and transcription rates according to the different phases of the cell cycle (*see Note 4*), which act as additional sources of extrinsic noise for the system.

Step 5

Now, we have to calculate the time when the next cell division time (CD_i) will occur. This is given as $CD_i = CC_i - CP_i$. Depending on the CD_i and the experimental observation time (in this case 4 h for the SmFISH experiment), the chosen i th cell may or may not undergo cell division (*see Note 5*).

Step 6

If the i th cell undergoes cell division within the experimental observation period (t_{obs}), then we calculate the rest of the observation time (t_{rest}) for the two newborn daughter cells as ($t_{rest} = CD_i - t_{obs}$). We will simulate both these daughter cells for t_{rest} time by implementing **Steps 1–4**.

Step 7

Once all these abovementioned noise sources are included, we will carry out the conventional Gillespie’s stochastic simulation for t_{obs} time for the i th single cells with or without cell division event.

Let us consider two different simulation scenarios, where we simulate single cells with or without cell division event chosen (Fig. 4). First, we initialize the system at the time t_0 ($t_0 = 0$ in this case) and provide the observation time t_{obs} . We set our t_{obs} as 4 h following the experimental condition related to Nanog transcriptional network. We simulated about 400 cells (following **Steps 1–7**) as considered in the experiments to replicate the smFISH experimental scenario. At the end of simulation time, we counted the number of each type of Nanog transcripts from each cell to quantify the corresponding allele-specific mRNA expressions. Within this t_{obs} time, some of the cells within the population might undergo cell division. For this, we calculated the cell cycle time for a particular cell as well as the current phase time of the cell at a time t_0 . Let’s say for a cell (**Cell 1**), the cell cycle time (CC_1) is 18 h, and the current cell cycle phase time (CP_1) comes out to be

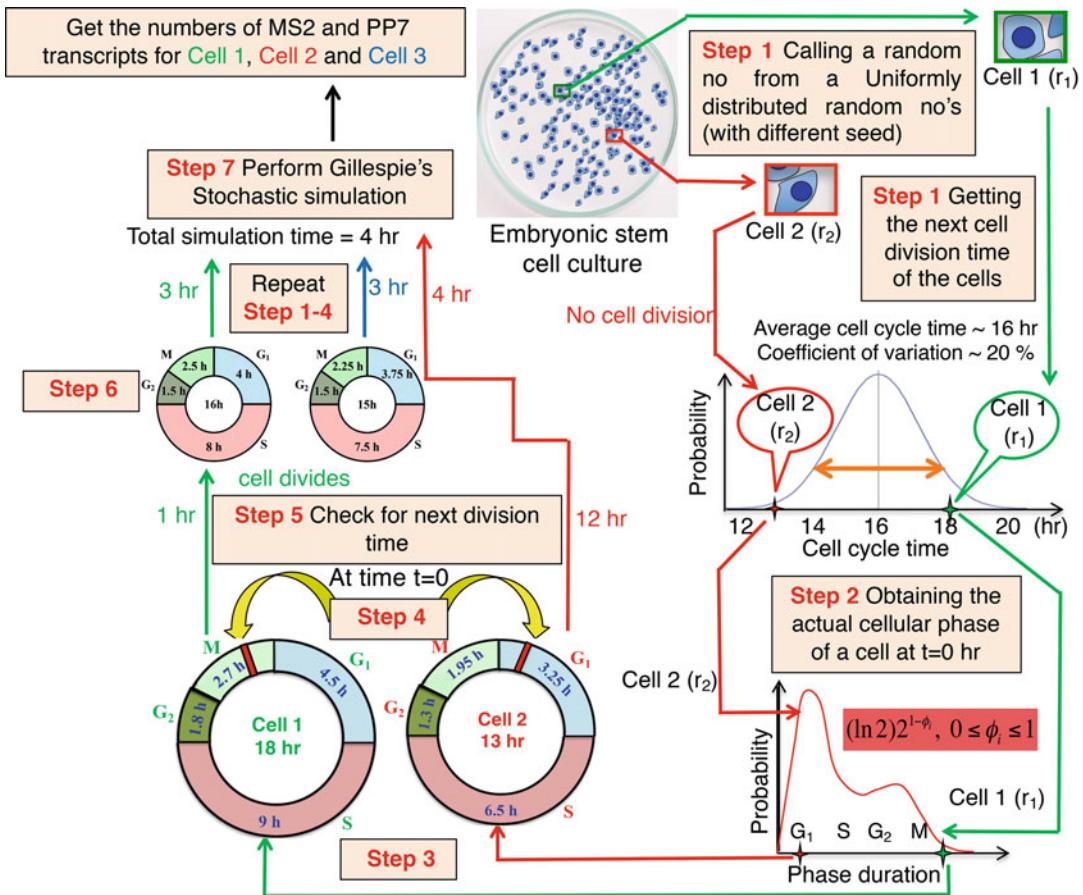


Fig. 4 Proposed numerical recipe to calculate the extrinsic and intrinsic fluctuation accurately for a biological network. Here we have taken the example of Nanog transcriptional regulatory network. We have demonstrated two different scenarios schematically: (i) stochastic simulation with cell division (**Cell 1**) and (ii) stochastic simulation without cell division (**Cell 2**). (This figure is taken from the publication Samanta and Kar [18] with copyright permission)

17 h (Fig. 4). That means **Cell 1** will undergo cell division (CD_1) in 1 h time. Thus, in 1 h time, **Cell 1** will divide into two daughter cells, and we will follow these two daughter cells for the t_{rest} time (3 h). In another situation, say for **Cell 2**, the cell cycle time (CC_1) randomly comes out as 13 h, and the current cell cycle phase time (CP_1) is 1 h (Fig. 4). Therefore, at the end of the experiment, i.e., $t_{obs} = 4$ h, **Cell 2** will not divide.

Notes

1. In an asynchronous population of cells, all the cells will not have the same cell cycle time, and they will be in different phases of the cell cycle [21]. **Step 1** allows us to implement

the idea that each cell within an asynchronous population have different cell cycle time and are in different state of their cell cycle at any given point of time.

2. This cell cycle phase distribution is based on the assumption that in any cellular population, the number of daughter cells is twice that of the mother cells [19]. Hence, we will obtain a greater number of cells in the earlier phases of the cell cycle.
3. The first 25% of the total cell cycle is considered as G₁ phase, and the next 50% is the S phase. After the S phase, 10% of the total cell cycle time is taken as G₂ phase, and the last 15% of the cell cycle is the M phase. These % phase timings can differ for different cell types [22, 23].
4. The gene activation rate is double [16] in the G₁-S phases compared to the G₂-M phase, whereas the transcription rate is known to become double [17] during S-G₂-M phases compare to the G₁ phase.
5. The i th cell will undergo cell division if the cell division time (CD_i) is less than the observation time (t_{obs}), and for those cells, the end time of simulation will be t_{obs} .

4 Applications

The method described above proved quite efficient to interpret the experimental observations related to Nanog transcriptional dynamics [18]. The stochastic simulation for the Nanog transcriptional network (Fig. 2a) under serum (Fig. 5a) as well as different inhibitory (Fig. 5b) conditions can reconcile the experimentally observed intrinsic and extrinsic noise ratio (45:55). Moreover, we obtain additional insight into the underlying mechanism by which this ratio is maintained in such a robust manner. Our analysis revealed that the dynamics of various Nanog transcriptional events are regulated in a specific way under serum and different inhibitory conditions to maintain the intrinsic to extrinsic fluctuation ratio around 45:55 [18]. This raises the important question: How one can alter this ratio to vary the proportion of intrinsic and extrinsic noise contribution in Nanog transcription? Interestingly, our stochastic simulation study predicted that we can increase the intrinsic noise contribution to Nanog transcription by only using a higher level of CHIR99021 inhibitor (Fig. 5c). This depicts the predictive power of the method.

However, employing a high level of inhibitors can be quite toxic to the cells. Is there any alternative way to manipulate the Nanog transcriptional heterogeneity? Recently, it has been demonstrated experimentally that by introducing transcript-specific microRNAs (miRNA), one can modify the heterogeneity observed

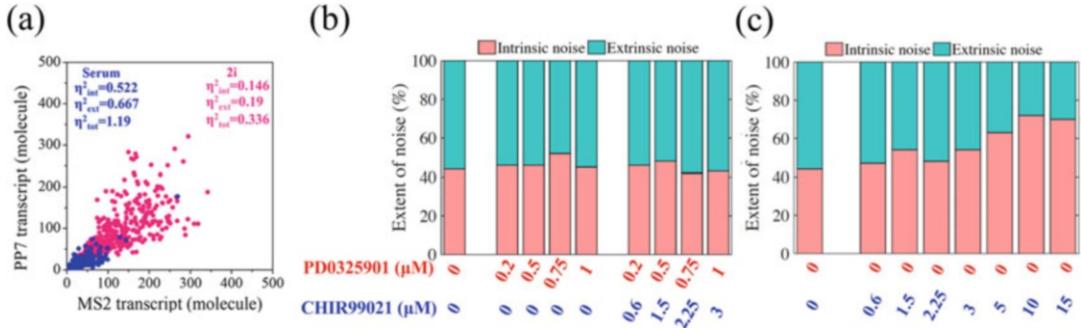


Fig. 5 Understanding the origin of robust Nanog transcriptional heterogeneity maintenance. **(a)** The simulation technique reproduces the 45:55 ratio of intrinsic to extrinsic noise under serum and bi-conditions (presence of both PD0325901 and CHIR99021 inhibitors in the different ratio). **(b)** Stochastic simulations under different inhibitory conditions reconciled the robust maintenance of the intrinsic to extrinsic fluctuations. **(c)** The model predicts that the intrinsic noise contribution can be increased by only employing the CHIR99021 inhibitor. (This figure is taken from the publication Samanta and Kar [18] with copyright permission)

at the level of transcription for a particular gene [24]. For the Nanog gene, there exists a miRNA (miR-296 [25]), which specifically degrades Nanog transcripts. Thus, by altering the 3'UTR region of the corresponding Nanog transcript, it is possible to produce transcript-specific miRNAs [26], which can specifically bind with a particular Nanog transcript and degrade it with varied efficiency. The 3'UTR regions can be designed so that the miR-296 can only bind with any one of these transcripts (MS2 (Fig. 6a), or PP7 (Fig. 6c)), or it binds to both of them [26].

Our stochastic simulations reveal that by introducing MS2-specific miR-296 into the system numerically with varied binding efficiencies, the proportions of the extrinsic noise in Nanog transcription can be further elevated (Fig. 6b). Whereas, by employing a PP7-specific miR-296 with altered binding efficiencies, the intrinsic noise can be increased in a significant amount (Fig. 6d). This implies that by suitably designing the 3'UTR region of Nanog transcripts, the intrinsic to extrinsic noise ratio can be influenced as per need. In this regard, our simulation methodology will come handy to envisage the possible outcome for such kinds of scenarios.

5 Conclusion

In biological systems, fluctuations do affect the dynamical outcome of important decision-making events [5]. Thus, estimating the effect of fluctuations on these events is becoming extremely crucial. With the experimental advances in cellular biology, it is getting easier to quantify the exact contributions of the intrinsic and extrinsic fluctuations associated with the transcriptional events of

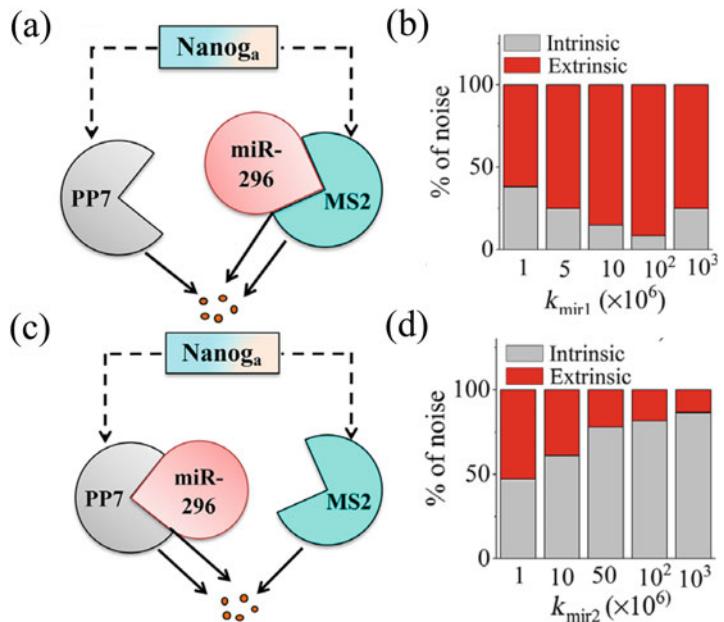


Fig. 6 Influencing the Nanog transcriptional heterogeneity by employing transcript-specific miR-296. (a) The miR-296 is specifically designed to inhibit the MS2-specific transcript of Nanog. (b) The percent contribution of intrinsic and extrinsic noise is plotted for different miR-296-MS2 transcript binding rates (k_{mir1}). (c) The miR-296 is specifically designed to inhibit the PP7-specific transcript of Nanog. (d) The percent contribution of intrinsic and extrinsic noise is plotted for different miR-296-PP2 transcript binding rates (k_{mir2}). (Parts of this figure are taken from the publication Samanta and Kar [26] with copyright permission)

any gene of interest [4, 5, 9]. In these experiments, it turns out that extrinsic fluctuations play a governing role in dictating the cell-fate decision-making processes under different circumstances [5]. However, these processes are extremely complicated, and one needs a reliable methodology to estimate the contribution of the extrinsic fluctuations related to these biological events. We put forward a new methodology [18] that takes into account most of the extrinsic fluctuations that can influence the transcriptional dynamics of any gene by taking an example of Nanog transcription.

Our method implicitly added most of the extrinsic sources of fluctuations by considering all the heterogeneities that come with a cell cycle event for a cell and reconciled various critical observations made in the context of Nanog transcriptional heterogeneity. The stochastic simulations further predicted how the ratio of intrinsic and extrinsic noise can be altered by either influencing the Nanog transcriptional machinery with specific inhibitors or by modulating the miR-296 interactions with the Nanog transcripts. Importantly, this method is quite generic and can be employed to decipher the

origin of heterogeneity in any gene of interest. However, like every other numerical method, this method also has few limitations. First, if the gene of interest is interacting highly with the cell cycle regulators, then it will be unreal to assume that the corresponding gene's dynamics are independent of the cell cycle regulation. Secondly, the method needs to be improved if there are more than one cell cycle events for a specific cell during the experimental time frame. These things can be systematically included in this method to make it more robust. However, the method in its current form is quite suitable to handle the problem as discussed by taking the Nanog transcriptional system.

Acknowledgments

Thanks are due to IIT Bombay for providing the RA-fellowship to (TS). This work is supported by the funding agency SERB, India (Grant no. CRG/2019/002640 and Grant no. MTR/2020/000261).

References

- Pedazra JM, Van OA (2005) Noise propagation in gene networks. *Science* 307:1965–1969. <https://doi.org/10.1126/science.1109090>
- McAdams HH, Arkin A (1999) It's a noisy business! Genetic regulation at the nanomolar scale. *Trends Genet* 15:65–69. [https://doi.org/10.1016/S0168-9525\(98\)01659-X](https://doi.org/10.1016/S0168-9525(98)01659-X)
- Ozbudak EM, Thattai M, Kurtser I et al (2002) Regulation of noise in the expression of a single gene. *Nat Genet* 31:69–73. <https://doi.org/10.1038/ng869>
- Kalmar T, Lim C, Hayward P et al (2009) Regulated fluctuations in Nanog expression mediate cell fate decisions in embryonic stem cells. *PLoS Biol* 7:33–36. <https://doi.org/10.1371/journal.pbio.1000149>
- Ochiai H, Sugawara T, Sakuma T, Yamamoto T (2014) Stochastic promoter activation affects Nanog expression variability in mouse embryonic stem cells. *Sci Rep* 4:1–9. <https://doi.org/10.1038/srep07125>
- Raser JM, O'Shea EK (2004) Control of stochasticity in eukaryotic gene expression. *Science* 304:1811–1814. <https://doi.org/10.1126/science.1098641>
- Swain PS, Elowitz MB, Siggia ED (2002) Intrinsic and extrinsic contributions to stochasticity in gene expression. *Proc Natl Acad Sci* 99:12795–12800. <https://doi.org/10.1073/pnas.162041399>
- Fu AQ, Pachter L (2016) Estimating intrinsic and extrinsic noise from single-cell gene expression measurements. *Stat Appl Genet Mol Biol* 15:447–471. <https://doi.org/10.1515/sagmb-2016-0002>
- Elowitz MB, Levine AJ, Siggia ED, Swain PS (2002) Stochastic gene expression in a single cell. *Science* 297:1183–1186. <https://doi.org/10.1126/science.1070919>
- Gillespie DT (1976) A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J Comput Phys* 22:403–434. [https://doi.org/10.1016/0021-9991\(76\)90041-3](https://doi.org/10.1016/0021-9991(76)90041-3)
- Gillespie DT (2001) Approximate accelerated stochastic simulation of chemically reacting systems. *J Chem Phys* 115:1716–1733. <https://doi.org/10.1063/1.1378322>
- Chatterjee A, Vlachos DG, Katsoulakis MA (2005) Binomial distribution based τ -leap accelerated stochastic simulation. *J Chem Phys* 122:1–7. <https://doi.org/10.1063/1.1833357>
- Anderson DF (2008) Incorporating postleap checks in tau-leaping. *J Chem Phys* 128. <https://doi.org/10.1063/1.2819665>
- Huh D, Paulsson J (2011) Random partitioning of molecules at cell division. *Proc Natl Acad Sci* 108:15004–15009. <https://doi.org/10.1073/pnas.1013171108>

15. Dessalles R, Fromion V, Robert P (2020) Models of protein production along the cell cycle: an investigation of possible sources of noise. *PLoS One* 15:1–25. <https://doi.org/10.1371/journal.pone.0226016>
16. Skinner SO, Xu H, Nagarkar-Jaiswal S et al (2016) Single-cell analysis of transcription kinetics across the cell cycle. *eLife* 5:1–24. <https://doi.org/10.7554/eLife.12175.001>
17. Zopf CJ, Quinn K, Zeidman J, Maheshri N (2013) Cell-cycle dependence of transcription dominates noise in gene expression. *PLoS Comput Biol* 9:1–12. <https://doi.org/10.1371/journal.pcbi.1003161>
18. Samanta T, Kar S (2019) Dynamical reorganization of transcriptional events governs robust Nanog Heterogeneity. *J Phys Chem B* 123: 5246–5255. <https://doi.org/10.1021/acs.jpcb.9b03411>
19. Kar S, Baumann WT, Paul MR, Tyson JJ (2009) Exploring the roles of noise in the eukaryotic cell cycle. *Proc Natl Acad Sci* 106: 6471–6476. <https://doi.org/10.1073/pnas.0810034106>
20. Niepel M, Spencer SL, Sorger PK (2009) Non-genetic cell-to-cell variability and the consequences for pharmacology. *Curr Opin Chem Biol* 13:556–561. <https://doi.org/10.1016/j.cbpa.2009.09.015>
21. Dowling MR, Kan A, Heinzel S, Zhou JH et al (2014) Stretched cell cycle model for proliferating lymphocytes. *Proc Natl Acad Sci U S A* 111(17):6377–6382. <https://doi.org/10.1073/pnas.1322420111>
22. Hindley C, Philpott A (2013) The cell cycle and pluripotency. *Biochem J* 451:135–143. <https://doi.org/10.1042/BJ20121627>
23. White J, Dalton S (2005) Cell cycle control of embryonic stem cells. *Stem Cell Rev* 1:131–138. <https://doi.org/10.1385/SCR:1:2:131>
24. Schmiedel JM, Klemm SL, Zheng Y et al (2015) MicroRNA control of protein expression noise. *Science* 348:128–132. <https://doi.org/10.1126/science.aaa1738>
25. Tay Y, Zhang J, Thomson AM et al (2008) MicroRNAs to Nanog, Oct4 and Sox2 coding regions modulate embryonic stem cell differentiation. *Nature* 455:1124–1128. <https://doi.org/10.1038/nature07299>
26. Samanta T, Kar S (2020) Fine-tuning Nanog expression heterogeneity in embryonic stem cells by regulating a Nanog transcript-specific microRNA. *FEBS Lett* 1–15. <https://doi.org/10.1002/1873-3468.13936>



Chapter 8

Meta-Dynamic Network Modelling for Biochemical Networks

Anthony Hart and Lan K. Nguyen

Abstract

ODE modelling requires accurate knowledge of parameter and state variable values to deliver accurate and robust predictions. Parameters and state variables, however, are rarely static and immutable entities, especially in a biological context. This observation undermines the predictions made by ODE models that rely on specific parameter and state variable values and limits the contexts in which their predictions remain accurate and useful. Meta-dynamic network (MDN) modelling is a technique that can be synergistically integrated into an ODE modelling pipeline to assist in overcoming these limitations. The core mechanic of MDN modelling is the generation of a large number of model instances, each with a unique set of parameters and/or state variable values, followed by the simulation of each to determine how parameter and state variable variation affects protein dynamics. This process reveals the range of possible protein dynamics for a given network topology. Since MDN modelling is integrated with traditional ODE modelling, it can also be used to investigate the underlying causal mechanics. This technique is particularly suited to the investigation of network behaviors in systems that are highly heterogenous or systems wherein the network properties can change over time. MDN is a collection of principles rather than a strict protocol, so in this chapter, we have introduced the core principles using an example, the Hippo-ERK crosstalk signalling network.

Key words Meta-dynamic network modelling, ODE modelling, Heterogeneity, Protein dynamics, Signalling networks, Hippo pathway, ERK pathway

1 Introduction

Ordinary differential equation (ODE) models of protein networks are well suited to investigating protein dynamics in response to perturbations, such as targeted drug treatment or growth factor stimulation [1, 2]. Like all models however, they are limited and ODE models are limited in two major ways. The first limitation regards knowledge of the “true” values of the parameters and state variables, and the second is the treatment of parameter and state variable values as static entities. Meta-dynamic network (MDN) modelling is a top-down modelling approach that can be synergistically integrated into traditional ODE modelling approaches to address these limitations.

Very rarely do we have knowledge of the “true” value of a parameter. Even the concept of a “true” value of a parameter is problematic. Parameters generally refer to the strength of a facet of a protein interaction, and there are many factors that can influence protein interactions: pH, temperature, post-translational modifications, catalytic enzymes, etc. While it is certainly possible to measure catalytic constants and substrate saturation concentrations utilizing pure reactants and enzyme solutions, there is no guarantee that this measured value will hold over the vast range of cellular heterogeneity [3, 4]. Additionally, the direct measurement of biological parameters might be feasible for a very small network, with few components and interactions, but would be prohibitively expensive, time-consuming, and risky for larger networks.

When investigating large networks, many network components and interactions are abstracted to make the model more tractable. When components are abstracted, information about the “hidden” components must be integrated into the remaining parameter values. This eliminates the ability to use biologically derived parameter values, mentioned above, and even reduces the ability to constrain the range of a parameter value. In addition, when we have incomplete knowledge about a network’s components and interactions, which is more often than not, this abstraction occurs implicitly even if this is not our intent. Parameter estimation is a tool that has been developed to overcome this problem [5]. In this method, biological data and optimization algorithms are used to tune a model’s parameters such that the model’s output matches biological data, creating a context-specific fitted parameter set. While this technique has become increasingly more popular, and frequently confers a model with a high degree of predictive capacity, it too fails to address the second major limitation of ODE modelling, i.e., the dynamic nature of network properties.

Evolution and heterogeneity are two prime examples of phenomena that can generate a wide range of parameter and state variable values for supposedly identical protein networks [6]. Evolution can alter the values of parameters over time through mutations that change a protein’s expression/interactions, and random chance can drive a large degree of heterogeneity in protein expression, even amongst cells of the same type [7, 8]. Particularly in the context of diseases like cancer, the prediction of parameters from an experiment testing an acute period of time and a limited number of cells, is thus only useful in this limited context. Generating parameter values, either through biological experiments or computational estimation, is critical for conferring predictive capacity to a model. However, this also limits an understanding of the ways in which a particular network topology behaves under different circumstances, i.e., the range of possible behaviors the topology can facilitate.

MDN modelling has been developed to address both of these limitations. It is not a separate, stand-alone technique. Instead, should be integrated into the traditional ODE modelling pipeline when it cannot be assumed that the constituent parameter and state variable values are homogenous or constant. The principal idea behind MDN modelling is to explore how the variation of parameter and state values affects protein behavior. This is then followed by further analysis to identify useful and informative patterns about the network being studied. MDN can thus be split into three phases: generation, measurement, and analysis. While there are common core steps in a given MDN model, the exact procedure will be dictated by the biological question that stimulated the production of the model in the first place. In this chapter we will explain each of these phases, using an example model to demonstrate the principles of MDN modelling.

2 Overview of Meta-Dynamic Network (MDN) Modelling

A distilled overview of MDN modelling can be seen in Fig. 1, where, as with all experimental techniques, the first stage of MDN modelling is to define the question that the model will attempt to answer. For MDN modelling this will generally be a question regarding a network phenomenon where network heterogeneity is thought to be important, or where the network can change over time. Once this question has be defined and the relevant network(s) identified, the construction of a network schematic can begin.

The construction of a network schematic entails selecting which proteins to include and identifying the relationships between them. Often only a subset of all the known proteins and interactions will be included in the network schematic as the computational cost of analyzing networks grows exponentially with size. When the network schematic has been defined, the next step is to convert it into mathematical equations.

Biochemical reaction rate laws are mathematical templates that describe how specific protein interactions drive changes in protein concentrations over time. The two most common reaction rate laws used in ODE modelling are mass action, describing complex formation and dissolution, and Michaelis Menten kinetics, describing enzymatic conversion of reactants into products. To convert a network schematic to a series of mathematical equations, one selects the appropriate rate law and applies it to the relevant protein interactions. This process continues until all protein interactions have been described by mathematical equations. This step can be quite laborious and error-prone, and a number of software tools have be developed to somewhat automate this process, including

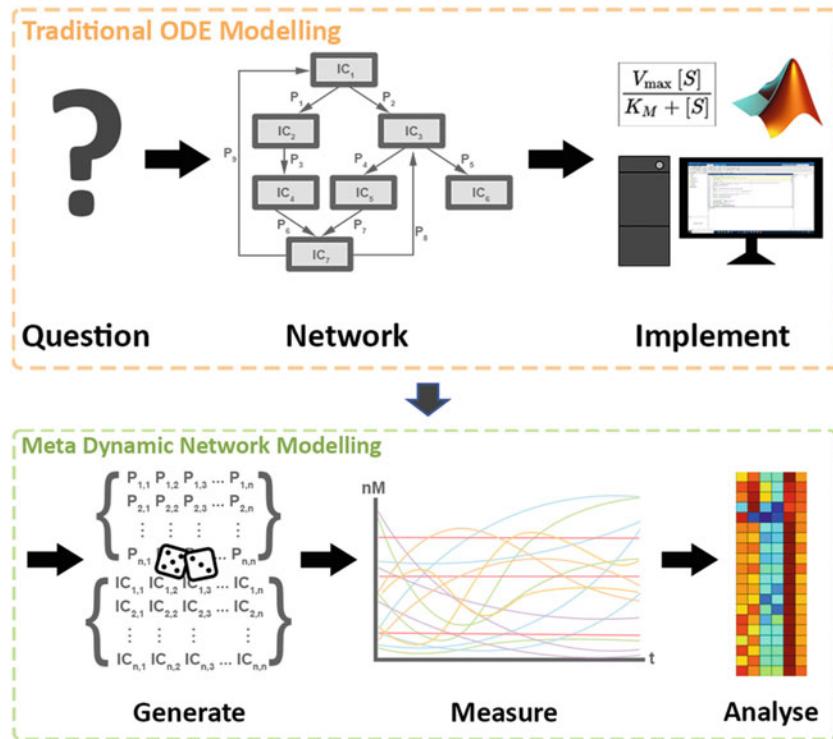


Fig. 1 Meta-dynamic network (MDN) modelling overview. The first three stages of MDN are identical to that of traditional ODE modelling. The last three stages broadly cover the novel principles of MDN modelling. The first step is the generation of a biological question. The second is the construction of a network schematic for the system being explored in the question. The third is the conversion of the network into ODEs and implementation in an appropriate software. The fourth is the generation of model instances. The fifth is measuring the networks response to stimulation/perturbation and generation of the MDN data/map. The last step is the analysis of the data/map in order to answer the initial biological question and hopefully stimulate new questions for further study

Copasi, PottersWheel, and the IntiQuan Tools MatLab Package [9–11]. Once the network schematic has been converted to ODEs, it is ready to be simulated.

A newly constructed mathematical ODE model should contain the following components. Input(s) and output(s), a set of parameters describing the strength of protein interactions, a set of state values that describe the concentration of proteins at any given time and a set of initial conditions that represent their basal concentrations. In MDN modelling, each of these components can be varied. Choosing which components to vary will depend on the biological question(s) being asked. When the components being varied have been selected, their ranges of variation must then be determined. Additionally, values for the components not being varied must also be carefully determined, as they will influence the final results. When this is complete, model instances can be generated to explore the defined ranges.

Model instances can be created by randomly generating values for the component(s) being varied within the defined ranges. Each model instance can then be simulated and the behavior of its constituent proteins stored. Typically, the recorded behavior will be time-course or dose-response data. This process is then continuously repeated until the aggregate data set is large enough to be analyzed. There is no upper limit to the number of model instances that can be tested; however as more and more data is collected, it becomes increasingly computationally expensive to analyze. One way to help identify when enough is enough is to employ error analysis, as seen later in Fig. 4e. As MDN modelling produces distributions of behavior, error analysis can be employed to identify the number of model instances needed for the distributions to stabilize. When a large number of model instances have been generated and simulated, the resultant data is ready for analysis.

The analysis of the MDN modelling data typically begins by qualitatively describing and classifying the behavior of each protein in each model instance. Qualitative behavior descriptors include increasing (INC), decreasing (DEC), biphasic (BIP), rebound (REB), no response (NRP), and other (ETC). To be useful, qualitative descriptors must be mathematically definable so that they can be applied to the collected dataset. When each protein in each model instance has been assigned a qualitative label, the dataset can be analyzed further. This is where MDN modelling begins to diverge significantly depending on the biological question being asked. One may simply wish to assess the distribution of behaviors for a particular protein or protein subset within the modelled network, or determine whether a particular behavior can be facilitated by the network topology. One may wish to extract all model instances that show a particular behavior and try to identify the causal mechanisms that drive this behavior. In the following sections, we will walk through an example question and MDN model to demonstrate its application and then introduce some useful additional analyses.

3 Case Study: The Hippo-ERK Crosstalk Network

The best way to illustrate MDN modelling and its potential applications is through a real-world example. Throughout the rest of this chapter, we will study a crosstalk network linking the Hippo and ERK MAPK signalling pathways via MDN modelling. A schematic of this network is given in Fig. 2. The Hippo pathway is known to play a role in the regulation of cell proliferation, differentiation, survival, and apoptosis [12]. The ERK signalling pathway is a mitogenic pathway that also drives cell growth and proliferation and has many interactions, or crosstalk, with the Hippo pathway

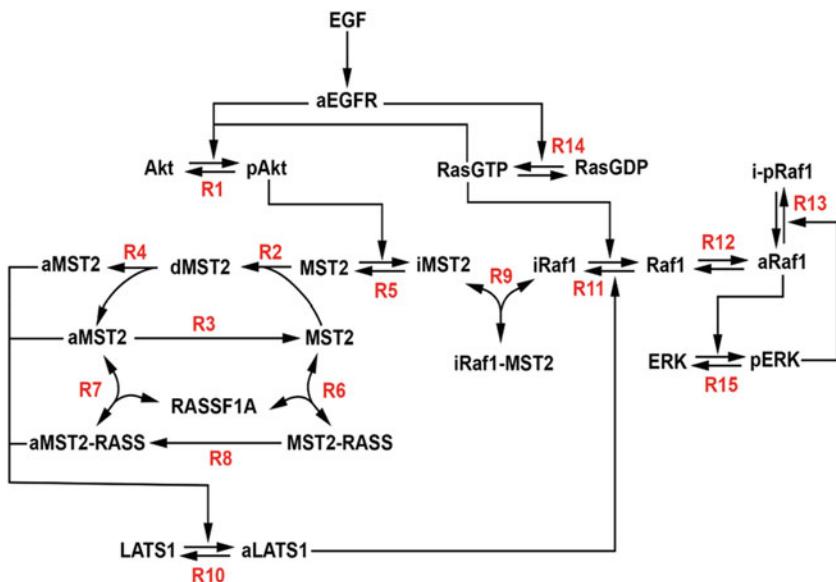


Fig. 2 Reaction schematic of the Hippo-ERK signalling crosstalk network model. The model input is EGF, and its major outputs are ERK phosphorylation (double-phosphorylated ERK or pERK) and active LATS1 (aLATS1). ERK phosphorylation leads to downstream events that drive cell growth and survival. Activation of LATS1 leads to the downregulation of cell survival genes and the downregulation of apoptotic genes. The model reaction rates and ODEs are given in Appendices A and B

[13–15]. Together these two signalling pathways can produce fine-tuned and integrated signals to facilitate efficient decision-making by cells.

The Hippo pathway consists of Mammalian Ste20-like kinase 1/2 (MST1/2), Ras association domain family 1 isoform (RASSF1A), large tumor suppressor kinase 1/2 (LATS1/2), yes-associated protein (YAP), and transcriptional co-activator with a PDZ-binding domain (TAZ). Activated MST1/2, promoted by RASSF1A, in turn activates LATS1/2 which then phosphorylates YAP and TAZ, inhibiting them through nuclear exclusion [14]. As YAP and TAZ are typically pro-survival transcriptional regulators, activating proliferation and anti-apoptosis genes, their inhibition by LATS1/2 suppresses growth and can potentially sensitize a cell to programmed cell death.

The ERK pathway is typically activated by the stimulation of a number of receptor tyrosine kinases, including EGFR. This results in the well-known protein cascade that ultimately activates MAPK proteins: Ras GTP → aRaf-1 → pMEK → pERK (Fig. 2). Phosphorylated ERK then activates a number of transcriptional regulators which upregulate growth and survival genes. Active ERK also negatively regulates Raf-1, providing a feedback loop that creates an upper limit to this pathway's activation. Finally, EGFR also stimulates the phosphorylation of Akt, which is further enhanced by Ras GTP creating a positive feedforward motif for Akt.

It can be seen from Fig. 2 that the Hippo and ERK pathways possess parallel but inverse roles in controlling cell survival and growth, wherein activated Hippo signalling suppresses growth and survival and activated ERK signalling promotes it. Likely due to this relationship, the two pathways are tightly integrated. The first crosstalk between the pathways involves the inhibitory phosphorylation of MST2 by activated Akt [14]. This phosphorylation causes MST2 to sequester phosphorylated Raf-1 into a MST2-Raf1 complex and inhibit ERK signalling, but it also reduces the availability of unbound MST2 and thus negatively regulates Hippo signalling. The second point of crosstalk involves active LATS1 serine-phosphorylating Raf-1, leading to its inactivation [14]. This contributes to the formation of the MST2-Raf1 complex and the sequestering of MST2 from the Hippo pathway and Raf-1 from the ERK pathway. Thus, this crosstalk also acts as a negative feedback loop for both pathways.

4 Application of MDN Modelling to the Hippo-ERK Crosstalk Network

4.1 MDN Setup and Implementation

Cells display a significant degree of heterogeneity in their protein expression, even healthy cells that are from the same tissue or cell line [16, 17]. If we restricted our analysis of the Hippo-ERK network to a single set of initial conditions, our model would only represent one possible cellular context. MDN modelling allows us to map an extensive range of possible behaviors that can be facilitated by a particular network topology. By performing MDN modelling, we can reveal the range of possible behaviors that can be facilitated by the Hippo-ERK network and even describe the distribution of each constituent protein's behavior. With this information we could perform further analyses to reveal patterns in this distribution and learn how to manipulate network components to achieve a desired signalling outcome. While the analyses can be as complex and deep as the researcher's imagination will allow, for the sake of understanding, we will keep our example question, and consequential analysis, relatively simple. The question we will attempt to address is as follows:

What is the effect of protein expression heterogeneity on the signalling outputs: active LATS1 and phosphorylated ERK, in response to EGF stimulation?

Defining this question allows us to create the network schematic seen in Fig. 2. We can then convert this network schematic to an ODE model using well-established methods that we will not go over here. Shin and Nguyen have previously investigated a very similar Hippo-ERK network, and we are going to utilize some of their results in our model [15]. Shin and Nguyen used Western blot time-course data to train their Hippo-ERK ODE model and

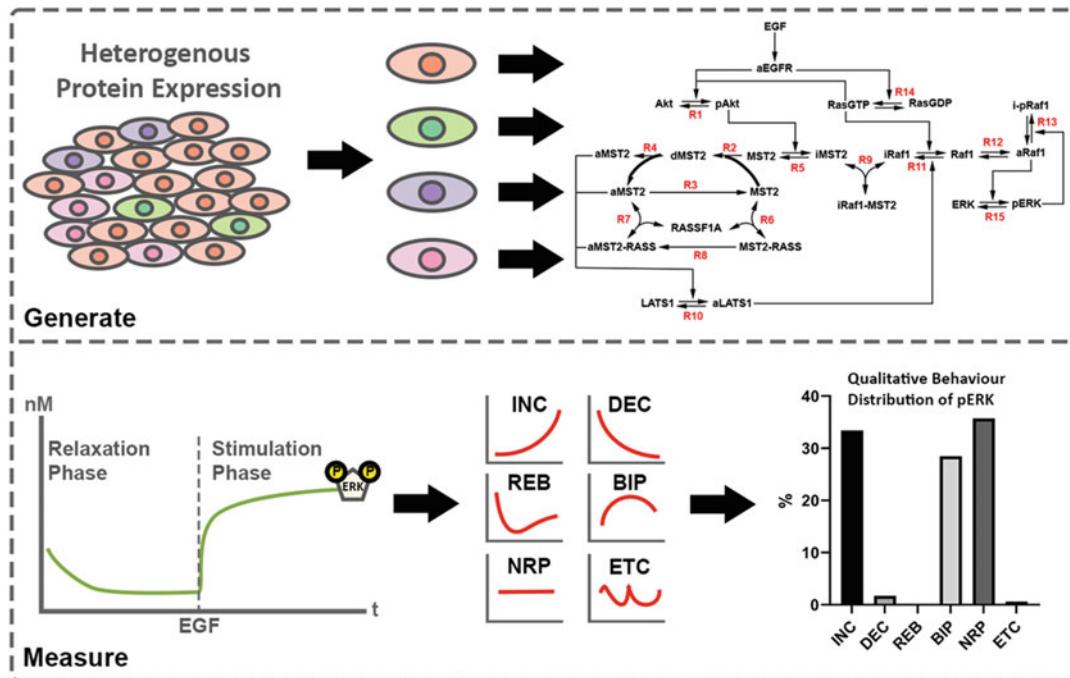


Fig. 3 Workflow for investigating the effect of protein expression heterogeneity on network dynamics. Here, pERK is used as an example model readout for illustration. First, model instances are created that represent a wide range of basal protein expression (1–10,000 nM). Using a parameter set obtained from the literature, these instances are stimulated with EGF and the time-course response of pERK recorded. The responses are then qualitatively sorted into six behavior profiles, allowing us to analyze the proportions of each behavior

generate a biologically relevant parameter set. In our MDN model, we will use this parameter set and vary the initial conditions for each protein over the range 1 to 10,000 nM to observe how protein expression affects active LATS1 and ERK behavior. An overview of this process can be seen in Fig. 3.

To test the range of protein concentrations, we must generate a large number of model instances, each with a unique set of initial conditions. There are multiple ways to do this, depending on the amount of information you have regarding a protein's abundance, but if you have no information, the best way is to generate model instances randomly. A simple way to do this is to generate an $M \times N$ matrix, where M is the number of instances you wish to test, N is the number of proteins in your network, and all the values within are random numbers within your defined range (1 and 10,000).

The next step is to simulate each model instance and record the time-course profile of each protein in each model instance. Here we have utilized the IQM software package for simulation and simply saved the outputs into a MATLAB file as a multidimensional array. Each unique set of initial conditions is combined with the core parameter set and then simulated. This process is then repeated until all of the initial conditions have been simulated.

Within each iteration the model is simulated in phases. Typically, the first phase is the relaxation phase, wherein no stimulations are given and the model reaches equilibrium. Following this are the stimulation phases, in which the model is stimulated, e.g., with ligands such as EGF in our case (see Fig. 3). When testing a very large number of model instances, the resulting file can become prohibitively large. One way to overcome this is only the relevant phases are recorded and only a subset of time-points within this phase is kept. It is often quite useful to select timepoints on a log scale to capture the critical early behaviors more comprehensively.

4.2 MDN Modelling-Based Analysis of Protein Expression Heterogeneity

To assess the distribution of the protein behaviors, we must convert the recorded time-course simulation data to qualitative descriptions. As mentioned earlier, a very general but useful set of qualitative behavioral descriptors are as follows: increasing (INC), decreasing (DEC), rebounding (REB), biphasic (BIP), no response (NRP), and other (ETC, including behaviors that do not fit into one of the previous categories, e.g., oscillation), illustrated in Fig. 3 (bottom panel). Each of these behaviors can be defined, and thus categorized automatically, using a set of heuristic quantitative criteria via quasi-mathematical code. For example, time-course profiles are defined as INC if the final concentration is at least 10% larger than the initial concentration, and at no point did the concentration drop below the initial concentration. NRP was defined as when no concentration value exceeds $+/- 10\%$ of the initial concentration.

When these definitions have been created for the behaviors being investigated, they can be applied to the recorded time-course simulation data. This results in another $M \times N$ matrix, where M is the number of model instances, N is the number of proteins and the values within are the qualitative descriptions of a protein's time-course profile within each model instance. Using this matrix, we can calculate the quantity (frequency) of each behavior for each protein in the network across all model instances; the results of which for the Hippo-ERK network can be seen in Fig. 4a.

Our initial question asks what the effects of protein expression heterogeneity are on active LATS1 (aLATS2) and phosphorylated ERK (pERK). In our MDN model, we generated 200,000 unique model instances with 200,000 unique initial condition sets that each represent a unique basal protein expression profile. We then stimulated each model instance with EGF and recorded the temporal response of each protein. Using this result we can now analyze the aggregate behaviors of aLATS1 and pERK to answer our question.

Looking at aLATS1, we can see that the dominant behavior is DEC. When the network is stimulated with EGF, LATS1 is inhibited across more than half of all the model instances (see Fig. 4b for examples of aLATS1 time-course profile). This indicates that,

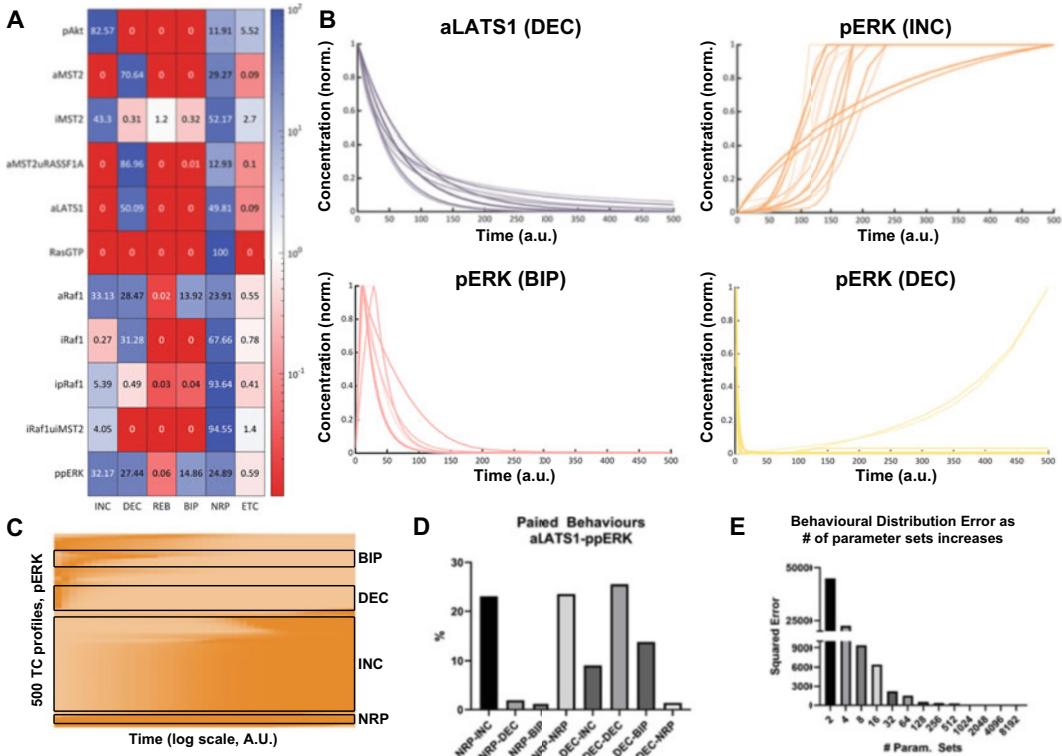


Fig. 4 Investigating the effect of protein expression heterogeneity on phosphorylated ERK (pERK) and activated LATS1 (aLATS1). Each model instance, representing a unique basal expression profile of the Hippo-ERK network protein components, is stimulated with EGF (100 nM) and the qualitative behavioral response of each protein is analyzed and recorded. **(a)** Heatmap of the proportion of qualitative behavior for key proteins within the Hippo-ERK network across the tested range, 1–10,000 nM. **(b)** Example time-course profiles, extracted from clusters generated in C, showing representative dynamic profiles for the dominant aLATS1 and pERK behaviors. **(c)** Clustering of 500 randomly selected time-course profiles of phosphorylated ERK. **(d)** Frequency of paired aLATS1 and pERK behaviors. **(e)** Error analysis showing how much the distribution changes when the number of model instances is doubled. The distribution converges after approximately 1000 model instances

because of the negative feedback from aLATS1 to Raf1 (Fig. 2), in the majority of cases LATS1 cooperates with Raf1-ERK to promote growth and survival. The second most common behavior of aLATS1 is that of no response (NRP), meaning in ~46% of the tested model instances EGF stimulation had little-to-no effect on aLATS1. Because LATS1 is responsible for inhibiting downstream pro-survival proteins/genes (via YAP/TAZ), any cells in which aLATS1 does not decrease in response to EGF would potentially display less growth and be more sensitive to apoptosis, relative to those cells that do experience decreasing aLATS1 [18].

Equally as important to commonly observed behaviors are seemingly impossible or exceedingly rare behaviors identified by MDN modelling. Active LATS1 never increases, rebounds, or shows biphasic behavior. This suggests that EGF is incapable of

stimulating a protein-expression-mediated increase in aLATS1. Similarly, there are no conditions among the 200,000 model instances, in which LATS1 decreases in activity initially in response to EGF but recovers due to adaptive network alterations.

Moving to pERK, we can see that it has four major behaviors: increasing, decreasing, biphasic, and no response. A subset of increasing, decreasing, and biphasic behaviors can be seen in Fig. 4b. Phosphorylated ERK stimulates cell proliferation and both INC and BIP behaviors display an initial increase in pERK [19]. The difference between these behaviors is that the increase in ERK phosphorylation is sustained in INC but is attenuated at some point in time in BIP. It is known that the duration of ERK signalling has differential effects on cellular outcomes, and this result demonstrates that the basal expression profile of a cell can dictate the length of time a cell will respond to EGF stimulation [20].

Perhaps counter-intuitively, there are many model instances that result in little to no response in ERK phosphorylation and even a large number of model instances where EGF causes ERK phosphorylation to decrease (Fig. 4c). NRP is often a large portion of the total model instances tested, and this is likely due to the creation of upstream bottlenecks in signalling that disconnect downstream signalling events. On the other hand, a decrease in EGF-induced ERK phosphorylation can be explained by overly strong negative feedback mechanisms. If the proteins that negatively regulate ERK signalling are more strongly activated in response to EGF stimulation than the proteins that activate ERK, the result will be an overall decrease in pERK. One possible scenario is as follows. EGF stimulation of Akt leads to the inhibition of MST2 which in turn can promote the sequestration of Raf1. If this mechanism sequesters too much Raf1, ERK will no longer be phosphorylated and will decrease over time. Cells possess many mechanisms that shutdown growth and proliferation, and this result suggests that one such mechanism is the manipulation of protein expression.

Looking at the behaviors individually provides useful insight into possible behavioral landscape, but it is also quite useful to see which behaviors may occur together. We can see in Fig. 4d that there are two highly enriched combinations for unresponsive aLATS1 and three for decreasing LATS1: NRP-INC at 23% and NRP-NRP at 23.5%, and DEC-INC at 9%, DEC-DEC at 25.5% and DEC-BIP at 13.7%. In the contexts where aLATS1 is unresponsive, it appears that pERK is split between either INC or NRP. This shows that there are protein expression profiles wherein unresponsive LATS1 activity is associated with ERK to displaying sustained activation but that there are also expression profiles where the whole network fails to transduce information to the level of LATS1 and ERK. It is also quite interesting to note that there are

very few contexts in which aLATS1 is unresponsive but pERK decreases or is biphasic. It seems as if when the Hippo network is disconnected from EGF stimulation, the ERK pathway does not possess the ability to effectively self-regulate.

When aLATS1 decreases, the largest subset of ERK response is to also decrease. This is quite a counterintuitive result but can be explained by the double negative regulation of ERK by EGF stimulation. Akt-mediated inactivation of MST1/2 leads to decreasing LATS1, but this also happens to sequester Raf-1 from the ERK signalling pathway, shutting it down. The next largest subset of ERK response is biphasic. In this scenario LATS1 activity decreases, while ERK increases initially and then decreases after a period of time. The decrease in LATS1 activity attenuates its inhibitory effect on Raf-1 and can thus stimulate ERK phosphorylation, but as ERK becomes increasingly active this turns on its auto-negative regulatory loop, leading to a BIP response. It also seems that there are contexts in which the negative self-regulation of ERK is not very strong, allowing for the sustained activation of ERK seen in the third largest combination, DEC-INC (Fig. 4d).

Analyzing the individual behaviors and combinations facilitates our understanding of what is possible within the network. Furthermore, we can begin to identify interesting behavioral correlations or even causal mechanisms. The obvious next step in MDN is to delve deeper and ask what causes these behaviors, what is responsible for their differences and what underlies the correlations between them. In the next section, we will show how this can be done by exploring the effects of parameter variation on network behavior and how patterns in these parameters can be targeted to manipulate the dynamic behavior of pERK.

4.3 Identification of Parameter Patterns Controlling Particular pERK Dynamics

ERK activity promotes cellular growth, proliferation, and survival, and as such the MAPK signalling pathway is often implicated in cancer [21]. Cancer is also an incredibly heterogeneous disease, and controlling it in a sustained manner will require a deep understanding of how signalling networks behave and change over time [22]. Differing cell contexts, epigenetic status, and mutations all give rise to heterogeneity in parameter values, and so it is useful to investigate the effect of parameter variation on key signalling outputs, such as pERK in this example study. In doing so we can gain a broad view of how the network acts under different parameter contexts and begin to identify patterns in these behaviors that can be exploited. The knowledge gained then allows us to manipulate protein dynamics in a precise, rational, and robust manner.

Similar to the previous *in silico* experiment, the first stage is the generation of model instances that represent a wide range of parameter values. In this experiment, parameter values were randomly generated between 10^{-4} and 10^4 , for 200,000 unique model

instances. These model instances were then simulated and the qualitative behaviors of each protein was assessed and recorded. Previously, Romano et al. performed time-course experiments in which HeLa cells were stimulated with serum over a 2-h time period, and output protein concentrations were measured using Western blots [14]. The measured proteins were phosphorylated Akt, active MST2, and hyperphosphorylated ERK, and their qualitative behaviors were increasing, decreasing, and biphasic, respectively. To give the investigation biological relevance, approximately 6000 model instances that qualitatively matched these behaviors were extracted for further analysis.

In the context of cancer, along with the biological observation of biphasic ERK phosphorylation, it may be therapeutically beneficial to reduce both the maximum concentration of pERK and its steady-state concentration post-drug treatment. However, we know that cancer is highly heterogeneous and that a good therapeutic target in one context may not be effective in another. But what drives these differences? If we could understand what makes a good target in one context and a bad one in another, we could differentially target the networks to achieve an effective therapeutic outcome. MDN modelling can be employed to produce a broad picture of how a wide range of network conditions will respond to perturbations. In this case, we want to identify patterns of parameter perturbations that reduce maximum and steady-state concentrations of pERK.

To identify patterns in the relationship between parameters and ERK phosphorylation, we can perform a global perturbation analysis. A workflow of the process can be seen in Fig. 5. In the first step, we perform an *in silico* knockdown of each individual parameter in each model instance and measure the effect on pERK. If we see an increase in the peak and steady state of pERK, we assign that parameter in that model instance a value of 1. If we see a reduction, we assign a -1, and if there is no change, we assign a value of 0. We then repeat this process until we have tested all of the model instances. The result is an $M \times N$ matrix, where M is the number of model instances, and N is the number of parameters.

The matrix we have now generated contains information that describes whether or not targeting a particular protein has a positive or negative effect on pERK. To identify patterns within this matrix, we can perform hierarchical clustering to group similar effective targets together, the result of which we can see in Fig. 6a. From this clustering analysis, we can extract the clusters and analyze their contents. In this case we summed the total number of times each parameter was an effective target within each cluster, as shown in Fig. 6b. This produces two important results. The first is a unique profile of parameter contributions to maximum and steady-state

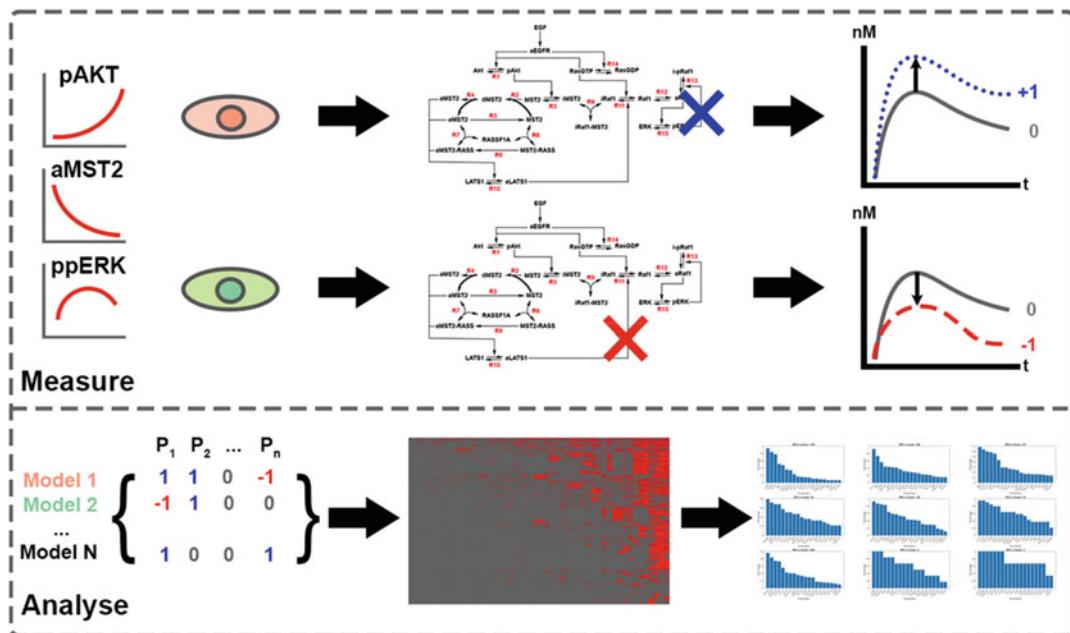


Fig. 5 Identifying parameter perturbation patterns resulting in reduced pERK. First, a subset of model instances is selected based on qualitative similarity to biological experiments. Next, each parameter in each model instance is knocked down by 20%, the modified model instance is simulated, and the time-course of pERK is recorded. If the parameter perturbation results in increased ERK phosphorylation, 1 is recorded, no change is recorded as a 0, and a decrease is recorded as a -1 . The resulting matrix is then clustered, allowing the identification of similarly responding groups. The parameter perturbations can then be ranked within each group to identify the parameters responsible for the decrease in pERK within each group

ERK phosphorylation that can potentially be used as a biomarker. The second is a ranked list of parameters, within each cluster, that could be used to select the most effective target(s) to reduce ERK phosphorylation.

The final step in this analysis is to test the effect of knocking down the top parameter(s) on ERK phosphorylation to computationally validate their efficacy. For this experiment, we selected the top parameter, Km13f1, in the largest cluster and performed an increasingly potent knockdown. We can see a subset of the effects on pERK in Fig. 6c. Km13f1 is a parameter involved in the inactivation of Raf-1 by pERK, and knocking it down increases the potency of this negative feedback regulation. While this parameter may not be directly targetable itself by pharmacological means, the results of this analysis suggest that targeting a protein that regulates this process would be highly effective in reducing ERK activity across a very broad range of cellular contexts.

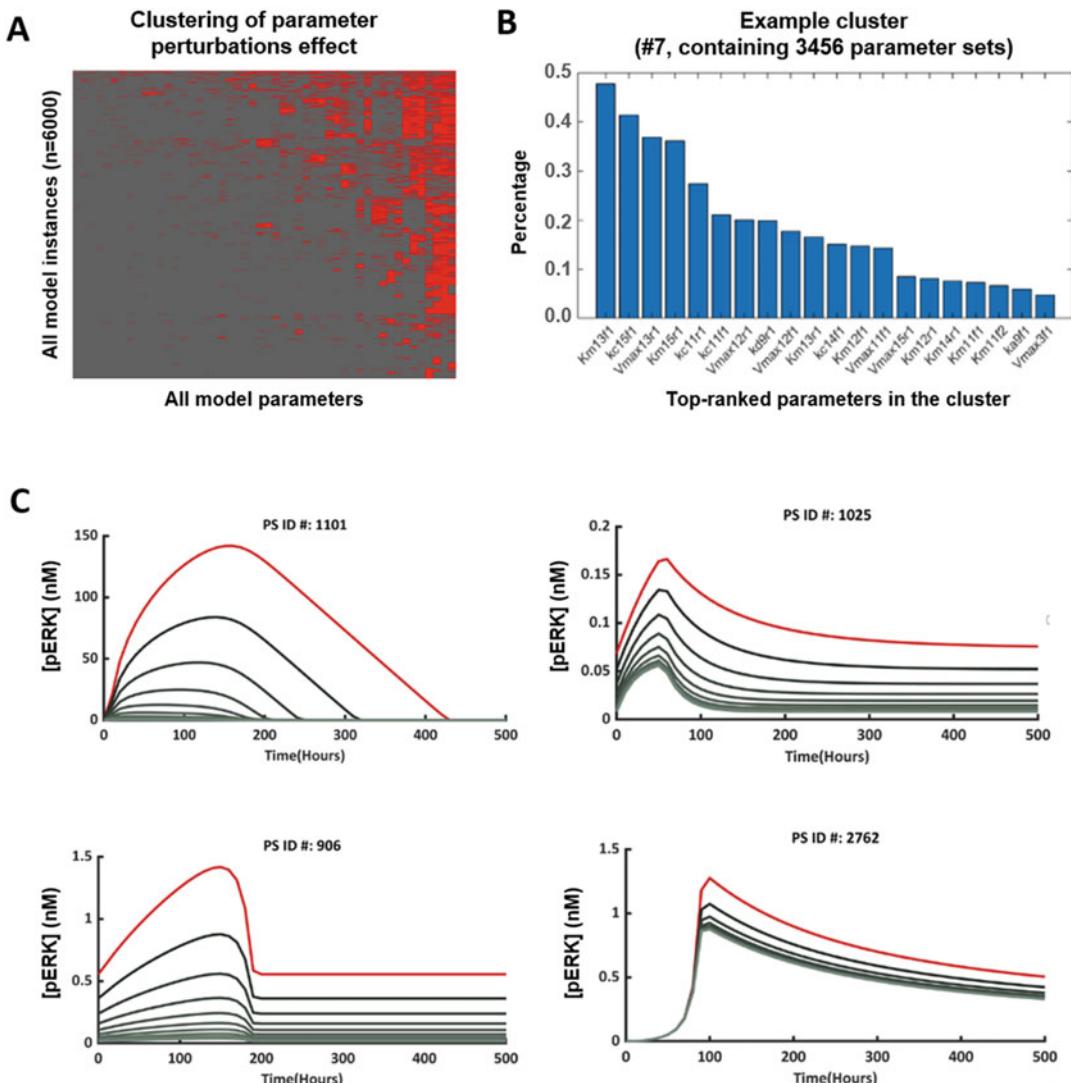


Fig. 6 Knockdown of driver parameters results in cluster-specific reduction of ERK phosphorylation. **(a)** Clustering of parameter perturbations that result in a reduction of both the maximum and an overall decrease in steady state levels of pERK. **(b)** Sum-rank of parameter perturbation frequency/proportion within each cluster, shown for cluster 7 as an example. **(c)** Sample knockdowns of driver parameter Km13f1 in cluster 7. Red line is wild type; dark to light grey represents an increasingly potent knockdown of the parameter

5 Discussion

The results of the analyses above require biological validation before they could be used to influence therapeutic strategies. However, even as is, they contain a wealth of useful and novel insights into the dynamic behavior of the Hippo-ERK crosstalk network. MDN modelling provides a map of the possible behaviors that can

be facilitated by a network topology and in doing so provides a quasi-upper limit to the range of behaviors we are likely to see in a heterogeneous cellular context.

Historically, we have only had aggregate experimental techniques available to us that pool information from a large population of cells into representative, bulk-level result. This may have generated the misleading idea that cells are all very similar and that network properties are conserved among cells of the same origin. The rise of single-cell techniques has well and truly demonstrated this idea to be false, but our understanding and appreciation of the implications is still catching up [23–25].

MDN is a technique that was aimed to understand how heterogeneity translates to the intracellular, protein network level. It is one thing to know that two cells can possess very different protein expression profiles; it is another to understand how this influences a cell’s ability to process internal and external information. For healthy cells, heterogeneity likely facilitates robustness and flexibility [26, 27]. The same network topology can be utilized in different cells, and, by simply altering expression levels, information can be processed in the appropriate manner for that cell type or situation. In the context of cancer, heterogeneity is what enables undesirable behaviors such as the development of drug-resistance and immune evasion. Understanding heterogeneity will enable us to have a much broader and systemic understanding of complex multicellular environments.

While capable of offering powerful and useful insight, this technique certainly has limitations; the largest of which is its scope. While we may not know the “true” value of a parameter or precisely know the concentrations of a network’s proteins, the range is likely much less than which has been presented in this analysis for any given context. By giving such broad ranges, we capture a similarly broad range of possibility, but not all possibilities are equally likely, and this can potentially skew our interpretations of MDN results. In this regard, when possible, biologically plausible ranges of kinetic parameters and protein concentrations should be utilized to inform MDN modelling.

MDN is an ensemble technique that roughly resembles a Monte Carlo method. At first glance they seem quite matched; however, MDN modelling is distinct in its application. Monte Carlo methods are those that rely on random sampling to gain insight into the behavior of complex systems, often numerical in nature [28, 29]. Monte Carlo methods are generally used to approximate an unknown value using random sampling. This method assumes that the expected value of a random variable can be approximated by taking the sample mean of independent samples of the variable. Ensemble modelling, in the context of machine learning, much more similarly resembles a Monte Carlo method. In the ensemble machine learning approach, averaging machine

predictions across multiple machine learning algorithms produces more accurate predictions, i.e., the averaging of the predictions across independent algorithms gets you closer to the true prediction [30, 31]. In MDN modelling however, we are not attempting to identify the “true” values of parameters or concentrations; we are simply evaluating the effect that their variation has on signalling outputs. In doing so, we are attempting to understand the possibilities contained within a network, but we are not making any comments regarding which one is true or more predictive.

To truly boost the analytical power of MDN modelling, we will need extensive knowledge regarding the biologically relevant ranges of parameters and protein concentrations. Experimental observations can be integrated into the MDN modelling pipeline to filter out less likely possibilities and test a much more biologically representative set of model instances. This also goes both ways however, with MDN modelling offering insight into the most fruitful targets for investigation or to identify the least amount of information required to separate two seemingly similar contexts.

In this chapter we introduced MDN modelling through an investigation of the Hippo-ERK crosstalk network. We explored the effects of protein heterogeneity on key signalling outputs and identified parameter perturbation patterns that allowed us to manipulate these outputs in a context-specific fashion. These are two possible applications of the principles of MDN, but there are many additional analyses that can be performed, such as the investigation of dose-response or the parameter changes required for the transformation of one behavior into another. Another obvious extension of the MDN principles demonstrated here would be to alter the network topology itself, while holding parameter and concentration values constant, to investigate the effects that specific protein relationships have on outputs. MDN modelling is a technique that is most effectively utilized when the cellular context is heterogeneous or changes over time and provides powerful mechanistic insight into how heterogeneity translates to network-level protein dynamics.

Acknowledgments

This work was supported by a Victorian Cancer Agency Mid-Career Research Fellowship (MCRF18026), an Investigator Initiated Research Scheme grant from National Breast Cancer Foundation (IIRS-20-094), and a Metcalf Venture Grant by Cancer Council Victoria, Australia, was awarded to L.K.N.

Appendix A: Ordinary Differential Equations of the MST2-ERK Crosstalk Model

The reaction rates are given in Appendix B.

Left-hand sides	Right-hand sides
$d[pAkt]/dt$	$v1 - v2$
$d[dMST2]/dt$	$v3$
$d[MST2]/dt$	$v4$
$d[aMST2]/dt$	$v5$
$d[iMST2]/dt$	$v6 - v7$
$d[MST2uRASSF1A]/dt$	$v8 - v9$
$d[aMST2uRASSF1A]/dt$	$v10 - v11$
$d[aMST2uRASSF1A]/dt$	$v12$
$d[iRafluiMST2]/dt$	$v13 - v14$
$d[aLATS1]/dt$	$v15 - v16$
$d[iRafl]/dt$	$v17 - v18$
$d[aRafl]/dt$	$v19 - v20$
$d[ipRafl]/dt$	$v21 - v22$
$d[RasGTP]/dt$	$v23 - v24$
$d[pERK]/dt$	$v25 - v26$

Appendix B: Reactions and Reaction Rates of the MST2-ERK Network Model

Reaction	Reaction rates
v1 $Akt \rightarrow pAkt$	$(kc1f1 * aEGFR * Akt) / (Km1f1 + Akt) + (kc1f2 * Akt * RasGTP) / (Km1f2 + Akt)$
v2 $pAkt \rightarrow Akt$	$(Vmax1r1 * pAkt) / (Km1r1 + pAkt)$
v3 $MST2 + MST2 \rightarrow dMST2$	$(2 * ka2f1 * MST2^2) / (Km3f1 + aMST2)$
v4 $aMST2 \rightarrow MST2$	$(Vmax3f1 * aMST2) / (Km3f1 + aMST2)$
v5 $dMST2 \rightarrow aMST2 + aMST2$	$(2 * kd4f1 * dMST2)$
v6 $MST2 \rightarrow iMST2$	$(kc5f1 * MST2 * pAkt) / (Km5f1 + MST2)$
v7 $iMST2 \rightarrow MST2$	$(Vmax5r1 * iMST2) / (Km5r1 + iMST2)$

(continued)

Reaction	Reaction rates
v8 MST2 + RASSF1A → MST2uRASSF1A	(ka6f1 * MST2 * RASSF1A)
v9 MST2uRASSF1A → MST2 + RASSF1A	(kd6r1 * MST2uRASSF1A)
v10 aMST2 + RASSF1A → aMST2uRASSF1A	(ka7f1 * aMST2 * RASSF1A)
v11 aMST2uRASSF1A → aMST2 + RASSF1A	(kd7r1 * aMST2uRASSF1A)
v12 MST2uRASSF1A → aMST2uRASSF1A	(Vmax8f1 * MST2uRASSF1A) / (Km8f1 + MST2uRASSF1A)
v13 iRaf1 + iMST2 → iRaf1uiMST2	(ka9f1 * iRaf1 * iMST2)
v14 iRaf1uiMST2 → iRaf1 + iMST2	(kd9r1 * iRaf1uiMST2)
v15 LATS1 → aLATS1	(kc10f1 * LATS1 * aMST2) / (Km10f1 + LATS1) + (kc10f2 * LATS1 * aMST2uRASSF1A) / (Km10f2 + LATS1)
v16 aLATS1 → LATS1	(Vmax10r1 * aLATS1) / (Km10r1 + aLATS1)
v17 Raf1 → iRaf1	(kc11f1 * aLATS1 * Raf1) / (Km11f1 + Raf1) + (Vmax11f1 * Raf1) / (Km11f2 + Raf1)
v18 iRaf1 → Raf1	(kc11r1 * iRaf1 * RasGTP) / (Km11r1 + iRaf1)
v19 Raf1 → aRaf1	(Vmax12f1 * Raf1) / (Km12f1 + Raf1)
v20 aRaf1 → Raf1	(Vmax12r1 * aRaf1) / (Km12r1 + aRaf1)
v21 aRaf1 → ipRaf1	(kc13f1 * aRaf1 * pERK) / (Km13f1 + aRaf1)
v22 ipRaf1 → aRaf1	(Vmax13r1 * ipRaf1) / (Km13r1 + ipRaf1)
v23 RasGDP→RasGTP	(kc14f1 * RasGDP * aEGFR) / (Km14f1 + RasGDP)
v24 RasGTP→RasGDP	(Vmax14r1 * RasGTP) / (Km14r1 + RasGTP)
v25 ERK→pERK	(kc15f1 * aRaf1 * ERK) / (Km15f1 + ERK)
v26 pERK→ERK	(Vmax15r1 * pERK) / (Km15r1 + pERK)

Appendix C: Parameter Values of the MST2-ERK Network Model

Parameter name	Parameter value
kclf1	0.001149
Km1f1	51.2083
kclf2	0.716985
Km1f2	0.744041
Vmax1r1	0.08687
Km1r1	0.014969
ka2f1	4472.19
Vmax3f1	1413.9
Km3f1	427.268
kd4f1	0.611723
kc5f1	6683.9
Km5f1	0.000831
Vmax5r1	7511.38
Km5r1	816.169
ka6f1	0.068398
kd6r1	0.113023
ka7f1	0.423669
kd7r1	1.22563
Vmax8f1	0.000569
Km8f1	6.7082
ka9f1	28.1235
kd9r1	0.000489
kc10f1	6189.56
Km10f1	3960.77
kclf2	0.000293
Km10f2	22.2638
Vmax10r1	2260.63
Km10r1	0.085035
kclf1	0.11768
Km11f1	10.6827
Vmax11f1	2.07105

(continued)

Parameter name	Parameter value
Km11f2	0.882094
kc11rl	0.920276
Km11rl	0.901533
Vmax12f1	317.336
Km12f1	3.19718
Vmax12rl	994.764
Km12rl	457.52
kc13f1	0.002742
Km13f1	207.086
Vmax13rl	254.671
Km13rl	0.076777
kc14f1	0.206094
Km14f1	120.52
Vmax14rl	1027.24
Km14rl	297.194
kc15f1	5.34199
Km15f1	0.036764
Vmax15rl	995.314
Km15rl	150.998
aEGFR_0	100
aEGFR_on	5000

References

1. Santra T (2018) Fitting mathematical models of biochemical pathways to steady state perturbation response data without simulating perturbation experiments. *Sci Rep* 8:11679–11679
2. Mishra SK, Bhowmick SS, Chua H, Zhang F, Zheng J (2015) Computational cell fate modelling for discovery of rewiring in apoptotic network for enhanced cancer drug sensitivity. *BMC Syst Biol* 9(Suppl 1):S4–S4
3. Kilpinen H, Goncalves A, Leha A, Afzal V, Alasoo K, Ashford S, Bala S, Bensaddek D, Casale FP, Culley OJ, Danecik P, Faulconbridge A, Harrison PW, Kathuria A, McCarthy D, McCarthy SA, Meleckyte R, Memari Y, Moens N, Soares F, Mann A, Streeter I, Agu CA, Alderton A, Nelson R, Harper S, Patel M, White A, Patel SR, Clarke L, Halai R, Kirton CM, Kolb-Kokociński A, Beales P, Birney E, Danovi D, Lamond AI, Ouwehand WH, Vallier L, Watt FM, Durbin R, Stegle O, Gaffney DJ (2017) Common genetic variation drives molecular heterogeneity in human iPSCs. *Nature* 546: 370–375
4. Jones EC, Uphoff S (2021) Single-molecule imaging of LexA degradation in *Escherichia coli* elucidates regulatory mechanisms and heterogeneity of the SOS response. *Nat Microbiol* 6:981–990

5. Degaspero A, Fey D, Kholodenko BN (2017) Performance of objective functions and optimisation procedures for parameter estimation in system biology models. *npj Syst Biol Appl* 3: 20
6. Bareche Y, Venet D, Ignatiadis M, Aftimos P, Piccart M, Rothe F, Sotiriou C (2018) Unravelling triple-negative breast cancer molecular heterogeneity using an integrative multiomic analysis. *Ann Oncol* 29:895–902
7. Romanos M, Allio G, Roussigné M, Combres L, Escalas N, Soula C, Médeville F, Steventon B, Trescases A, Bénazéraf B (2021) Cell-to-cell heterogeneity in Sox2 and Bra expression guides progenitor motility and destiny. *elife* 10:e66588
8. Lawson DA, Kessenbrock K, Davis RT, Pervolarakis N, Werb Z (2018) Tumour heterogeneity and metastasis at single-cell resolution. *Nat Cell Biol* 20:1349–1360
9. Sunnåker M, Schmidt H (2016) IQM tools-efficient state of the art modeling across pharmacometrics and systems pharmacology. *J Pharmacokinet Pharmacodyn* 43:S69–S70
10. Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, Singhal M, Xu L, Mendes P, Kummer U (2006) COPASI—a COMplex PAthway SImlator. *Bioinformatics* 22:3067–3074
11. Maiwald T, Eberhardt O, Blumberg J (2012) Mathematical modeling of biochemical systems with PottersWheel. *Methods Mol Biol* 880: 119–138
12. Ma S, Meng Z, Chen R, Guan KL (2019) The Hippo Pathway: biology and pathophysiology. *Annu Rev Biochem* 88:577–604
13. Nguyen LK, Matallanas DG, Romano D, Kholodenko BN, Kolch W (2015) Competing to coordinate cell fate decisions: the MST2-Raf-1 signalling device. *Cell Cycle* 14:189–199
14. Romano D, Nguyen LK, Matallanas D, Halasz M, Doherty C, Kholodenko BN, Kolch W (2014) Protein interaction switches coordinate Raf-1 and MST2/hippo signalling. *Nat Cell Biol* 16:673–684
15. Shin S-Y, Nguyen LK (2016) Unveiling hidden dynamics of Hippo Signalling: a systems analysis. *Genes* 7:44
16. Tikhonova AN, Dolgalev I, Hu H, Sivaraj KK, Hoxha E, Cuesta-Domínguez A, Pinho S, Akhmetzyanova I, Gao J, Witkowski M, Guillamot M, Gutkin MC, Zhang Y, Marier C, Diefenbach C, Kousteni S, Heguy A, Zhong H, Fooksman DR, Butler JM, Economides A, Frenette PS, Adams RH, Satija R, Tsirigos A, Aifantis I (2019) The bone marrow microenvironment at single-cell resolution. *Nature* 569:222–228
17. Norris D, Yang P, Shin SY, Kearney AL, Kim HJ, Geddes T, Senior AM, Fazakerley DJ, Nguyen LK, James DE, Burchfield JG (2021) Signaling heterogeneity is defined by pathway architecture and intercellular variability in protein expression. *iScience* 24:102118
18. Li D, Ji J-X, Xu Y-T, Ni H-B, Rui Q, Liu H-X, Jiang F, Gao R, Chen G (2018) Inhibition of Lats1/p-YAP1 pathway mitigates neuronal apoptosis and neurological deficits in a rat model of traumatic brain injury. *CNS Neurosci Ther* 24:906–916
19. Lavoie H, Gagnon J, Therrien M (2020) ERK signalling: a master regulator of cell behaviour, life and fate. *Nat Rev Mol Cell Biol* 21:607–632
20. Hiratsuka T, Bordeu I, Pruessner G, Watt FM (2020) Regulation of ERK basal and pulsatile activity control proliferation and exit from the stem cell compartment in mammalian epidermis. *Proc Natl Acad Sci U S A* 117:17796–17807
21. Maik-Rachline G, Hacohen-Lev-Ran A, Seger R (2019) Nuclear ERK: mechanism of translocation, substrates, and role in cancer. *Int J Mol Sci* 20:1194
22. Hastings JF, O'Donnell YEI, Fey D, Croucher DR (2020) Applications of personalised signalling network models in precision oncology. *Pharmacol Ther* 212:107555
23. Hou Y, Guo H, Cao C, Li X, Hu B, Zhu P, Wu X, Wen L, Tang F, Huang Y, Peng J (2016) Single-cell triple omics sequencing reveals genetic, epigenetic, and transcriptomic heterogeneity in hepatocellular carcinomas. *Cell Res* 26:304–319
24. Kinker GS, Greenwald AC, Tal R, Orlova Z, Cuoco MS, McFarland JM, Warren A, Rodman C, Roth JA, Bender SA, Kumar B, Rocco JW, Fernandes P, Mader CC, Keren-Shaul H, Plotnikov A, Barr H, Tsherniak A, Rozenblatt-Rosen O, Krizhanovsky V, Puram SV, Regev A, Tirosh I (2020) Pan-cancer single-cell RNA-seq identifies recurring programs of cellular heterogeneity. *Nat Genet* 52:1208–1218
25. Ho DW, Tsui YM, Sze KM, Chan LK, Cheung TT, Lee E, Sham PC, Tsui SK, Lee TK, Ng IO (2019) Single-cell transcriptomics reveals the landscape of intra-tumoral heterogeneity and stemness-related subpopulations in liver cancer. *Cancer Lett* 459:176–185
26. Billing U, Jetka T, Nortmann L, Wundrack N, Komorowski M, Waldherr S, Schaper F, Ditttrich A (2019) Robustness and information transfer within IL-6-induced JAK/STAT Signalling. *Commun Biol* 2:27

27. Kucharavy A, Rubinstein B, Zhu J, Li R (2018) Robustness and evolvability of heterogeneous cell populations. *Mol Biol Cell* 29:1400–1409
28. Ballnus B, Hug S, Hatz K, Görlitz L, Hasenauer J, Theis FJ (2017) Comprehensive benchmarking of Markov chain Monte Carlo methods for dynamical systems. *BMC Syst Biol* 11:63
29. Fallahi S, Skaug HJ, Alendal G (2020) A comparison of Monte Carlo sampling methods for metabolic network models. *PLoS One* 15: e0235393
30. Bannick MS, McGaughey M, Flaxman AD (2021) Ensemble modelling in descriptive epidemiology: burden of disease estimation. *Int J Epidemiol* 49:2065–2073
31. Poirion OB, Jing Z, Chaudhary K, Huang S, Garmire LX (2021) DeepProg: an ensemble of deep-learning and machine-learning models for prognosis prediction using multi-omics data. *Genome Med* 13:112



Chapter 9

Rapid Particle-Based Simulations of Cellular Signalling with the FLAME-Accelerated Signalling Tool (FaST) and GPUs

Gavin Fullstone

Abstract

Cellular signalling is a vital process in living organisms for coordinating highly diverse responses to various stimuli. Particle-based modelling excels in its ability to model complex features of cellular signalling pathways including stochasticity, spatial effects, and heterogeneity, thus improving our understanding of critical decision processes in biology. Yet, particle-based modelling is computationally prohibitive to implement. We recently developed FaST (FLAME-accelerated signalling tool), a software tool that harnesses the power of high-performance computation to reduce the computational burden of particle-based modelling. In particular, employing the unique massively parallel architecture of graphic processing units (GPUs) provided extreme speed ups of simulations by >650-fold. In this chapter, we provide a step-by-step walkthrough of how to use FaST to create GPU-accelerated simulations of a simple cellular signalling network. We further explore how the flexibility of FaST can be used to implement entirely customized simulations while still including the intrinsic speed up advantages of GPU-based parallelization.

Key words Systems biology, Particle-based modelling, Cell signalling, GPU parallelization

1 Introduction

Cellular signalling is essential in translating various stimuli into diverse cell responses, including proliferation, migration, and death. Cell responses can dramatically vary depending on the characteristics of the individual cell, the strength, and the duration of the stimuli. Cells employ various levels of regulation that determine how each individual cell will respond to a stimulus or multiple stimuli including by varying the concentration of signalling components, controlling subcellular spatial distribution of signalling components, feedback\feedforward mechanisms, and complex cross-talk between multiple signalling pathways. Systems modelling approaches have been used with great success to delineate cell signalling decision-making processes [1, 2]. However, commonly employed systems modelling approaches, such as ordinary

differential equation (ODE) modelling, do not always reflect the complex structural heterogeneity and stochastic effects that are intrinsic to many biological signalling pathways and can profoundly affect cell signalling outcomes. A number of particle-based modelling systems have recently been established that can encompass such spatiotemporal considerations and are intrinsically stochastic, providing an improved method for modelling such effects in biology [3–8]. However, these have often been limited by their complexity in implementation and slow computational times. In particular, the latter is a major hurdle preventing the wide-scale use of particle-based modelling in systems modelling of cell signalling. Not only does it restrict modelling to time scales that are of less relevance to biology, where meaningful time points are often in the order of hours and days, but it also is poorly compatible with application of well-established modelling techniques such as parameter estimation, sensitivity analysis, and uncertainty analysis.

We previously developed FaST, a software that addresses the long run times of particle-based modelling using either CPU or GPU-based parallelization strategies [9]. We demonstrated vast reductions in simulation times, particularly on GPU-based architecture. In subsequent tests, using state-of-the-art high-performance GPUs, we were able to reduce simulation times further to a matter of minutes, representing a >650-fold speed up (*see* Table 1). In this chapter, we will demonstrate the use of FaST to produce GPU-parallelized model of cellular signalling. We focus on the GPU-implementation due to its superior gain in performance compared to the CPU implementation. The theory behind FaST and our particle-based modelling approach are covered in detail previously [9]. Here, we focus on a step-by-step building of a cellular signalling pathway using FaST, covering model specification, compilation, simulation and analysis, and finally customization.

2 Materials

2.1 FaST Installation

The latest version of FaST is downloadable from <https://doi.org/10.5281/zenodo.2620047>. It was written using Matlab 2016b and is implementable from within Matlab itself or via available Linux, Mac, or Windows binaries located in the \bin folder. Binaries require the appropriate Matlab Runtime (MCR) installed through the included Web installer or downloadable from <https://www.mathworks.com/products/compiler/mcr>. FaST is also compatible with newer versions of Matlab, albeit with potential differences in the appearance of the graphical user interface.

Table 1

Specification of the Agent file. Properties to be included, identifier text and expected inputs for specifying the Agent file

Property	Identifier	Expected input
Name	Name	Alphanumeric
Unique identifier	Type_Identifier	Unique integer
Type of agent	Type	<i>Soluble</i> <i>Receptor</i>
Location within cell	Location	<i>Intracellular</i> <i>Extracellular</i> <i>cell_membrane</i>
Concentration	Concentration	Molar concentration (scientific or decimal format)
Diffusion coefficient	Diffusion_Coefficient	$\text{m}^2 \text{ s}^{-1}$ (scientific or decimal format)

2.2 Building, Compilation, and Running FaST Models for GPUs

The code generated by FaST requires building using FLAME GPU 1 and compilation using a C compiler. This requires the following:

1. A CUDA capable graphics card device (NVidia).
2. CUDA toolkit (CUDA 8.0 or later) and compatible developer driver, downloadable directly from NVidia.
3. The FLAME GPU software development kit (SDK) 1, downloadable from <https://flamegpu.com/download/>
4. Microsoft Visual Studios (for windows compilation).
5. A C compiler. For Windows, a C compiler can be downloaded alongside Microsoft Visual Studios. A C compiler is usually distributed as part of Linux.

The code generated by FaST is built and compiled through the FLAME GPU 1 SDK. It should be noted that FaST is currently incompatible with FLAME GPU 2.

3 Methods

For each step, example inputs and expected outputs are placed in the depository <https://doi.org/10.18419/darus-2175> for reference purposes. These can be used to troubleshoot issues that arise during the protocol.

3.1 Creating a Simulation with FaST

The first step in creating any FaST simulation is specification of the model components and their reactions. For the purposes of this protocol, we will use a simple receptor-ligand binding model as an example to illustrate the required steps and act as a tutorial model for increasing user familiarity with FaST-based simulations.

Importantly, the methods described hereafter are equally applicable to more complex networks of interactions. To generate the basic model, FaST requires two user defined input files, an *Agent* file and a *Reaction* file, which specify the components and interactions of the system.

3.1.1 Writing the Agents File

The agent input file is a simple text file (.txt) that contains basic information about each component (e.g., proteins, cofactors) in the model. For each component, FaST requires a name, a unique identifier, a type of protein (soluble or membrane-bound), a cellular location (e.g., extracellular or intracellular), a concentration at time 0, and a diffusion coefficient. The information required and the expected inputs are outlined in Table 1. For the receptor-ligand binding example, we will create three single agents, a soluble extracellular ligand, a cell membrane receptor *RECEPTOR*, and a cell membrane receptor-ligand complex *RL*. The resulting file should look like this:

```
Name=LIGAND
Type_Identifier=1
Type=soluble
Location=extracellular
Concentration=3e-8
Diffusion_Coefficient=3e-11
Name=RECEPTOR
Type_Identifier=10
Type=receptor
Location=cell_membrane
Concentration=1.2e-7
Diffusion_Coefficient=1e-12
Name=RL
Type_Identifier=11
Type=receptor
Location=cell_membrane
Concentration=0
Diffusion_Coefficient=1e-12
```

In FaST, concentrations are given as molar (M) concentrations, while diffusion coefficients are measured in $\text{m}^2 \text{s}^{-1}$. More information regarding picking a reasonable diffusion coefficient in the absence of experimental data is contained in *see Note 1*.

3.1.2 Writing the Reactions File

The reaction file is a simple text file (.txt) that contains information on reactions that take place within the simulation. All reactants and products must be declared in the Reactions file as well as the type of reaction and appropriate reaction rates. The type of reactions included in FaST are reversible, irreversible, catalysis, degradation,

Table 2

Specification of the Reaction file. Properties to be included, identifier text and expected inputs for specifying the Reaction file

Property	Identifier	Expected input
Reaction		$A + B > AB + C + D + E$ (0–2 reactants and 0–4 products, separated by $>$)
Type of reaction	Type	<i>Reversible</i> <i>Irreversible</i> <i>Catalysis</i> <i>Degradation</i> <i>Synthesis</i>
Forward reaction rate	k_f	$M^{-1} s^{-1}$ (second order) or s^{-1} (first order) or $M s^{-1}$ (zero order) (scientific or decimal format)
Reverse reaction rate	k_r	s^{-1} (scientific or decimal format)

or synthesis. The information required and the expected inputs are outlined in Table 2. For the receptor-ligand binding example, we will implement a single reversible reaction:



with a forward reaction rate (k_f) of $10^5 M^{-1} s^{-1}$ and a reverse reaction rate (k_r) of $0.001 s^{-1}$. The resulting file should look like this:

```
RECEPTOR+LIGAND>RL
Type=reversible
kf=1e5
kr=0.001
```

3.1.3 Running FaST

After writing the Agent and Reaction files, it is time to run FaST and create the basic simulation code. FaST can be launched directly from Matlab by running the FaST.m script located in the FaST folder or alternatively by running the appropriate binaries for your operating system in the FaST\bin directory. Running FaST should open the graphic user interface (GUI) (Fig. 1). Here, in the *Input Files* panel, the software needs to be supplied with the path to the Agent and Reaction files. It also requires an Output Directory under *Model Options*, a time step for Brownian motion and a time step for reactions under *Global Values*. The time step can be changed later but generally has to be chosen on a bespoke basis for each different model simulation (see Note 2). For our example model, we will use a time step of 0.0001 s for Brownian motion and 0.05 s for reactions. Lastly, select GPU under *Model Options*. Finally, click Make.

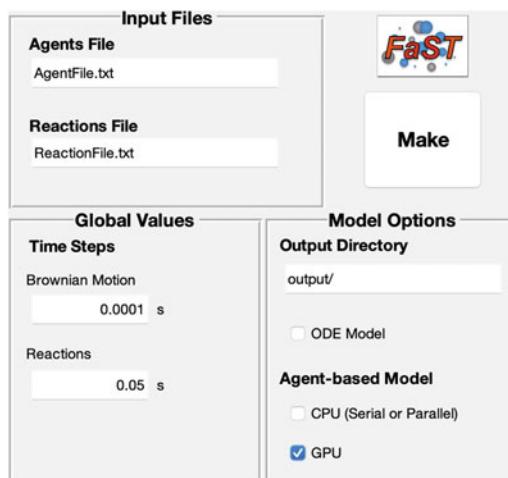


Fig. 1 The FaST graphical user interface

3.1.4 Evaluating the Code

FaST makes multiple files within the chosen output folder; these can be split into three categories, model-dependent files (`globals.h`, `AgentVariables.h`, `ReactionVariables.h`, `XMLModelFile.xml` and `functions.c`), scripts to create the initial states (`0Creator.c`, `globals.h`, `AgentVariables.h`), and scripts to retrieve simulation data into standard table format (`getdata.c`, `ReadData.h`, `WriteData.h`). Verify that these files have indeed been created in the output directory given in Subheading 3.1.3. A synopsis of what each file includes is included in Table 3.

If these files have not been created, this is likely due to an error within the input files or options. If running from outside of Matlab, it may be advisable to run the FaST executables within a command prompt window for improved error reporting.

3.2 Compilation with FLAME GPU

Compilation of FaST simulations for GPU-based architecture requires the FLAME GPU 1 software development kit (SDK). Simulations are compiled for Windows via Microsoft Visual Studios (Subheadings 3.2.2 and 3.2.3) or for Linux-based systems through `make` (Subheading 3.2.4). This section will cover how to create a new project folder for your FaST code and compile it using FLAME GPU.

3.2.1 Creating a New Project

FLAME GPU is distributed with a python tool, `new_examplify.py` located in the tools folder, to automatically create a new Visual Studios project, and makefile from the FLAME-GPU-SDK \examples\EmptyExample template. While it is possible to duplicate the empty example and modify the folder names, file names, and individual files manually, we would highly recommend using this tool. To run the tool, open a python command interface, and change to the FLAME GPU SDK directory and type:

Table 3
FaST output files and their function

File name	Used for	Description
globals.h	Initial states Simulation	This header file contains global parameters, particularly for the geometry of the simulated cell
AgentVariables.h	Initial states Simulation	This header file contains definitions for the concentration and diffusion of each single agent. Diffusion values are given as sigma values for a Gaussian probability distribution in nm as outlined in the publication of Fullstone et al.
0Creator.c	Initial states	This script creates the initial or starting states of the signalling pathway according to the concentration, geometry and localization of each protein
ReactionVariables.h	Simulation	This header file contains definitions for each individual reaction either as binding radius, unbinding radius, probability of unbinding, degradation, or synthesis rates as calculated according to the publication of Fullstone et al.
XMLModelFile.xml	Simulation	This file contains the description of the agents, memory variables, functions, and communications. It is used to by FLAME GPU to create the formal communicating X-machine model
functions.c	Simulation	This file contains the code for all the functions contained in the pathway.xml file
getdata.c	Analysis	This c file contains the code for collecting the data from the simulation and converting it to an excel file
ReadData.h	Analysis	This contains the code for reading the different memory variables from a simulation
WriteData.h	Analysis	This contains the code for writing the excel file. This can be modified to analysis different aspects of the model

```
python tools\new_example.py --base EmptyExample
Fast_example
```

This will create a new folder FLAME-GPU-SDK\examples\Fast_example which contains the following files and folders: iterations folder, src folder, Visual Studio solution file (Fast_example.sln), Visual Studio project file (Fast_example.vcxproj), Visual Studio filters file (Fast_example.vcxproj.filters), Makefile, and readme.md. The complete output source code (including the .xml file, the .c file, all .h files, and the \xml folder) generated by FaST in Subheading 1 should be placed in the FLAME-GPU-SDK\examples\Fast_example\src\model folder. This should replace the example functions.c and XMLModelFile.xml files that are already contained in this folder. Do this before proceeding with the subsequent steps.

3.2.2 Setting Up

*Microsoft Visual Studios
(Windows Only)*

Compilation of GPU-accelerated FaST simulations on Windows occurs through Microsoft Visual Studios. The project can be opened into Visual Studio by opening the newly created `Fast_example.vcxproj` found within `FLAME-GPU-SDK\examples\Fast_example`. Depending on the version of Visual Studios, CUDA, and operating system, you may be requested to update the Visual Studio build tools and Windows target. We have had mixed results with using different build tools and Windows target than that specified by FLAME GPU projects, and therefore we advise to stick to the tried and tested versions specified by FLAME GPU (for more details, *see Note 3*).

3.2.3 Compilation with

*Microsoft Visual Studios
(Windows Only)*

Now we will build the simulation code using Visual Studios. Initially, we will build this in debug mode without a visualization (*Debug_Console* mode) in the case of unexpected problems. This can be changed easily within the Project properties; other options are build for release (*Release_Console*) or for visualization (*Debug_Visualisation* or *Release_Visualisation*). To build the simulation, select *Build Solution* from the Build menu or press `ctrl+B`. The compilation was successful if the line

```
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

appears in the Output window. If it failed, check what error messages were returned, and try to discern the issue. The most likely issue at this stage is incompatible versions of CUDA, inability to find the correct CUDA version as specified in the Project menu → Build Customizations or incompatible build tools (*see Note 3*). You may receive some warnings of unused variables; this is expected behaviour when not all types of reactions are utilized as in the current example. By default, the simulation executable should be created in the `FLAME-GPU-SDK\bin\x64\Debug_Console` folder.

3.2.4 Compilation with

Make (Unix Only)

For Unix, compilation of FaST simulations is achieved using *make*. A makefile is generated by the `new-example.py` script ran in Sub-heading [3.2.1](#) and found within `FLAME-GPU-SDK\examples\Fast_example`. This makefile contains settings for compilation of the FaST simulation and links to a common FLAME GPU makefile, defined in `FLAME-GPU-SDK\tools\common.mk`, for final compilation. At this stage, we will not be using the visualization features of FLAME GPU, and therefore we have to edit the makefile accordingly. In the makefile, set the `HAS_VISUALISATION` flag (line 29) to 0.

Next, we will compile the simulation using the Terminal:

```
cd \PATH-TO\FLAME-GPU-SDK\examples\Fast_example
make
```

By default, the simulation executable should be created in the FLAME-GPU-SDK\bin\linux-x64 folder.

3.3 Running a Simulation

Now we have successfully built the simulation code, it is time to run a FaST simulation. This generally has three steps: specification of the initial states of the simulation, running the simulation, and analyzing the results. For this part, we will use a command line interface, the instructions here are written for the Developer Command Prompt that is distributed with Visual Studios and its native C\C++ compiler, but this can also be done using terminal, other command prompts, and C compilers.

3.3.1 Creating the Initial States

In order to run a FaST simulation, a declaration of the initial states of the simulation, specified in a modified XML format, is required. This, in short, specifies which molecules are present at the beginning of the simulation and their memory variables, such as xyz coordinates and their identity. In our example, we will use 0Creator, a tool created by FaST, for creating such an initial states file. The 0Creator also requires the AgentVariables.h and globals.h files for compilation. AgentVariables.h contains information on the number of each molecule present at time 0, and their diffusion coefficient. globals.h defines the geometry; this is pre-set to a cubic geometry with a planar membrane separating intracellular and extracellular compartments but can be changed easily from within globals.h. To compile and run 0Creator, open the command prompt and type the following:

```
cd C:\PATH-TO\FLAME-GPU-SDK\examples\Fast_example
\src\model
cl 0Creator.c
0Creator.exe
```

This should create a 0.xml file within the FLAME-GPU-SDK \examples\Fast_example\src\model\xml folder. Inspect the contents of 0.xml; it should contain information for each LIGAND and RECEPTOR agent created by the 0Creator.

3.3.2 Running a Simulation

Next, we will run the simulation from the command prompt. Fast simulation executables require a path to the initial states (0.xml), created in the previous section, as well as the number of iterations to run. For our example, we will run 6000 iterations, corresponding to 5 min of real time (iterations \times 0.05 s time step declared in Subheading 3.1.3). By default, FLAME GPU executables are located in the bin folder. Therefore, enter the following in the command line:

```
cd C:\PATH-TO\FLAMEGPU_SDK\bin\x64\Debug_Console
Fast_example.exe
...\\..\examples\Fast_example\src\model\xml\0.
xml 6000
```

In addition, it is also possible to change the frequency of how often the output is saved as an .xml file using the XML_output_frequency argument (default = 1). For GPU-based simulations generally, the writing of simulation data can be a particular bottleneck to fast simulation, so consider reducing the output frequency where possible.

3.3.3 Analyzing the Data

FLAME GPU outputs model simulations in XML format, including each individual particle and their memory variables. Included within FaST is an inbuilt tool, called getdata, to convert these files into a spreadsheet table format with the absolute number of each reactant at each time point. The getdata program needs to be first compiled. Compilation of getdata.c also depends on the header files, ReadData.h and WriteData.h, which read the XML format of data and write the spreadsheet format, respectively. For custom analysis of simulations, the WriteData.h file can be modified to change outputs or perform calculations. To compile getdata, type the following into the command line:

```
cd C:\PATH-TO\FLAME-GPU-SDK\examples\Fast_example
\src\model
cl getdata.c
```

To run the data script, getdata needs as arguments the number of iterations, the frequency of iterations, and the path to the iterations folder, for our example, type the following:

```
getdata.exe 6000 1 xml\
```

The data will be saved as a file called data.xls within the model folder. A simple graph of the output should resemble those seen in Fig. 2a. Please note, due to stochasticity in the simulations, minor differences are expected.

3.3.4 Adding a Visualization

One of the additional benefits of utilizing GPU-based simulation is the capability to dynamically visualize the simulation. FLAME GPU offers basic integrated visualization support while also offering CUDA OpenGL interoperability to allow creation of customized visualizations. Alternatively, FLAME GPU console mode output .xml files are also compatibility with the FLAME visualiser (https://github.com/FLAME-HPC/flame_visualiser), which can be used for post-hoc visualization of FaST simulations (example in Fig. 2b).

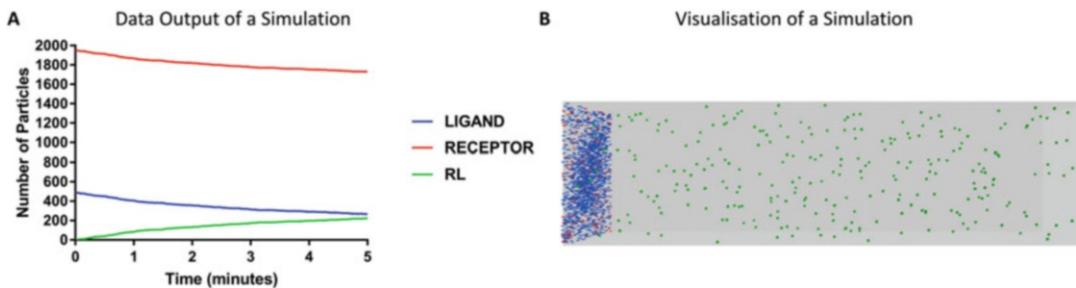


Fig. 2 Example output of a FaST simulation. **(a)** Graph of the example model produced from getdata. **(b)** Visualization of the example model produced with the FLAME Visualiser

To use the integrated visualization support, the simulation must be compiled including visualization settings contained within the header file FLAME-GPU-SDK\examples\Fast_example\src\visualisation\visualisation.h. Modifying the settings, if necessary, is well documented within the FaST GPU documentation. To compile the FaST_example in visualization mode through Visual Studio, the settings in the build menu need to be changed (Build → Configuration Manager). Select the debug_visualisation option. If compiling on unix-based systems, the HAS_VISUALISATION flag (line 29) within the makefile must be changed to 1. Build\make the simulation as before. The executables will be held in the bin\x64\Debug_Visualisation folder. To run the simulations with visualization, only the path to the 0.xml is required as an argument:

```
Fast_example.exe C:\PATH-TO\FLAME-GPU-
SDK\examples\Fast_example\src\model\xml\0.xml
```

The simulation will then run infinitely with the visualization until the visualization window is closed; therefore, the number of iterations does not need to be specified.

3.4 Customizing a Simulation

An integral part to the design of FaST is the ability to add custom functionality while maintaining the performance increase associated with GPU-parallelization. Therefore, in this section, we will demonstrate how to customize FaST simulations. For this, we will modify our example receptor-ligand binding model to add lipid rafts and restrict the movement of the receptor to these rafts. While there are several ways to do this, we will add the lipid rafts as an additional agent within the simulation. This in theory allows the rafts to be highly dynamic within the simulation, better reflecting the biological nature of lipid rafts, but also introduces the user to the requirements in adding a third type of agent (not a receptor or ligand) into the simulation. To create the lipid rafts, we will have to individually modify the XMLModelFile.xml, the functions.c, and the 0Creator.c, located in the FLAME-GPU-SDK\examples\Fast_example\src\model folder.

3.4.1 Adding an Entirely New Agent

Agents are initially declared within the XMLModelFile.xml file inside the <xagents> tag (line 13). To specify the lipid raft agent and assign it memory for id, $x/y/z$ coordinates, and a fixed radius, add the following text immediately after the <xagents> tag:

```
<gpu:xagent>
<name>Raft</name>
<memory>
<gpu:variable><type>int</type><name>id</name></gpu:variable>
<gpu:variable><type>double</type><name>x</name></gpu:variable>
<gpu:variable><type>double</type><name>y</name></gpu:variable>
<gpu:variable><type>double</type><name>z</name></gpu:variable>
<gpu:variable><type>double</type><name>radius</name></gpu:variable>
</memory>
```

Next, we will define a single function which broadcasts the lipid raft's current location to the receptor as a single message called Raft_Location. To add the function, immediately after the declaration of memory, add the following text:

```
<functions>
<gpu:function>
<name>Raft_Output</name>
<currentState>Raft</currentState>
<nextState>Raft</nextState>
<outputs>
<gpu:output>
<messageName>Raft_Location</messageName>
<gpu:type>single_message</gpu:type>
</gpu:output>
</outputs>
</gpu:function>
</functions>
```

FLAME GPU is built on the principle of a formal state machine. Therefore, we also need to declare the states for the lipid raft, the type of agent, and allocate enough memory for all raft agents. As there is only one function for Raft, it requires only a single state. To declare this, the type of agent and the memory requirement, add the following immediately after the function declarations:

```

<states>
<gpu:state>
<name>Raft</name>
</gpu:state>
<initialState>Raft</initialState>
</states>
<gpu:type>continuous</gpu:type>
<gpu:bufferSize>1024</gpu:bufferSize>
</gpu:xagent>

```

This completes the declaration of the lipid raft as a new agent.

Next, each receptor must know in which lipid raft it is located and where that lipid raft is located. Therefore, we need to modify the receptor agent memory and functions. To this end, we will firstly add a new variable to the receptor memory containing the ID number of the lipid raft it is located in. To do this, immediately after the line:

```

<gpu:variable><type>double</type><name>Dc</name></gpu:variable>

```

within the receptor agents memory (ensure that this follows <name>Receptor</name> in the document), add the following line:

```

<gpu:variable><type>int</type><name>Raft_id</name></gpu:variable>

```

Furthermore, we will need to tell the receptor to read the location message from the lipid rafts to find their location. For this, we will need to find the receptor function Receptor_Output (specified within XMLModelFile.xml as <name>Receptor_Output</name>). Then, immediately before the <outputs> tag, add the following code:

```

<inputs>
<gpu:input>
<messageName>Raft_Location</messageName>
</gpu:input>
</inputs>

```

This allows the receptor to access a message, called Raft_Location, containing the ID, x -, y -, z -coordinates and radius of the lipid raft. This will be used to restrict the motion of the receptor to its own lipid raft.

Next, we have to declare what is included within the Raft_Location message itself and allocate memory to it. Locate the tag <messages> within the XMLModelFile.xml. Immediately below this tag, add the following:

```
<gpu:message>
<name>Raft_Location</name>
<variables>
<gpu:variable><type>int</type><name>id</name></gpu:variable>
<gpu:variable><type>double</type><name>x</name></gpu:variable>
<gpu:variable><type>double</type><name>y</name></gpu:variable>
<gpu:variable><type>double</type><name>z</name></gpu:variable>
<gpu:variable><type>double</type><name>radius</name></gpu:variable>
</variables>
<gpu:partitioningNone></gpu:partitioningNone>
<gpu:bufferSize>1024</gpu:bufferSize>
</gpu:message>
```

Finally, we must update the process layers of the simulation to include the new lipid raft function Raft_Output. Layers define the order that each function is performed and therefore describes any functional dependency. For example, the function Receptor_Output that we just modified requires the message Raft_Location as an input. Therefore, the function that generates the Raft_Location message, Raft_Output, must proceed Receptor_Output. To do this locate the <layers> tab and enter a new layer directly below it:

```
<layer>
<gpu:layerFunction>
<name>Raft_Output</name>
</gpu:layerFunction>
</layer>
```

3.4.2 Adding and Modifying a Function

In addition to specifying the new agent, functions, and communications through modification of the XMLModelFile.xml, it is also required to add the appropriate function code within the functions.c. Functions in FLAME GPU are declared as simple integer functions. However, they also require a set number of arguments in the correct order if the function accesses agent memory, communication messages, or the random number generator API. The best way to ensure this is done correctly is to use an inbuilt tool in FLAME GPU for building the functions.c from a functions.xslt template

and the specified XMLModelFile.xml. This requires an XSLT processor included within the FLAME-GPU-SDK\tools folder. To run this, open a command prompt, navigate to the FLAME-GPU-SDK \tools folder, and type the following:

```
XSLTProcessor.exe
..\examples\Fast_example\src\model\XMLModelFile.xml
..\FLAMEGPU\templates\functions.xslt
..\examples\Fast_example\src\model\functions_template.c
```

This generates the file functions_template.c within the FLAME-GPU-SDK\examples\Fast_example\src\model folder. In functions_template.c you can find a template for adding the lipid raft function Raft_Output. This also includes an example for adding the Raft_Location message. Below is a modified version to add to the functions.c that simply adds the function and generates the Raft_Location message containing the raft memory variables. Add this text immediately below the definitions at the top of functions.c:

```
__FLAME_GPU_FUNC__ int Raft_Output(xmachine_memory_Raft* agent,
xmachine_message_Raft_Location_list* Raft_Location_messages)
{
    int id = agent->id;
    double x = agent->x;
    double y = agent->y;
    double z = agent->z;
    double radius = agent->radius;
    add_Raft_Location_message(Raft_Location_messages,
    id, x, y, z, radius);
    return 0;
}
```

As well as adding the Raft_Output function, we also need to modify the Receptor_Output function to firstly access the Raft_Location messages and adapt its Brownian motion to the confines of the lipid raft. Firstly, replace the function declaration:

```
__FLAME_GPU_FUNC__ int
Receptor_Output(xmachine_memory_Receptor* xmemory,
xmachine_message_Receptor_Location_list*
Receptor_Location_messages, RNG_rand48* rand48)
```

With the following, also findable in the functions_template.c file, to permit access to the Raft_Location messages:

```
__FLAME_GPU_FUNC__ int
Receptor_Output(xmachine_memory_Receptor* xmemory,
xmachine_message_Raft_Location_list* Raft_Location_messages,
xmachine_message_Receptor_Location_list*
Receptor_Location_messages, RNG_rand48* rand48)
```

Next, we have to add C code so that the Receptor agents access the current location of the lipid raft they are located in. To achieve this, each Receptor has to cycle through all Raft_Output messages and identify the message that matches the Raft_id now held in the Receptor memory. Immediately at the top of the Receptor_Output function (after the curly bracket {), add the following:

```
// Declare variables for raft data
double raftx, rafty, raftz, raftradius;
// Read messages and identify parental lipid raft
xmachine_message_Raft_Location* current_message =
get_first_Raft_Location_message(Raft_Location_
messages);
while (current_message)
{
// Get my raft ID and location
if (xmemory->Raft_id == current_message->id)
{
raftx = current_message->x;
rafty = current_message->y;
raftz = current_message->z;
rafradius = current_message->radius;
}
current_message =
get_next_Raft_Location_message(current_message,
Raft_Location_messages);
}
```

This saves the x -, y -, and z -coordinates and radius of the lipid raft. Thereafter, we must create a rule to prevent the Brownian motion of receptor beyond the confines of the lipid raft as defined by the radius. Therefore, immediately prior to closing the Brownian motion loop with the line

```
counter++;
```

add the following code:

```
// Prevent motion outside current raft
double distance = sqrt (((xmemory->x-raftx)*(xmemory->x-
raftx))+((xmemory->z-raftz)*(xmemory->z-raftz)));
if (distance > raftradius)
{
xmemory->x -= deltax;
xmemory->z -= deltaz;
}
```

This should finalize the changes to the model itself. At this point it is advisable to check that this compiles as expected using the steps in Subheading 3.2. If any error messages are reported, try to identify and correct the error (*see Note 4*).

3.4.3 Modifying the Initial States

In addition to adding lipid rafts to the simulation code, they must be declared in the initial states. Therefore, in this section, we will modify the initial state creator built by FaST, OCreator, to add lipid rafts and restrict all initial receptors to these lipid rafts. To do this we will implement a loop that creates 20 lipid rafts of radius 150 nm and add a set rule that prevents the rafts from intersecting. To do this, immediately before the line

```
/* Create LIGAND Ligand Agents */
```

add the following code:

```
/* Create lipid raft agents */
printf ("Creating RAFT Agents\n");
int raft_counter = 0;
double raftx[20];
double raftz[20];
double raftradius[20];
while (raft_counter < 20)
{
int intersecting = 1;
// Prevent intersection of current raft with other
rafts
while (intersecting == 1)
{
// Set raft location
rafradius[raft_counter] = 150.0;
raftx[raft_counter] = minextracellular_xboundary +
rafradius[raft_counter] +
(rand()/(double)(RAND_MAX))*((maxextracellular_x-
boundary-
```

```

minextracellular_xboundary)-(2*raftradius[raft_-
counter])),;
raftz[raft_counter] = minextracellular_zboundary +
raftradius[raft_counter] +
(rand()/(double)(RAND_MAX)*((maxextracellular_z-
boundary-
minextracellular_zboundary)-(2*raftradius[raft_-
counter])));
// Check intersection
intersecting = 0;
int counter = 0;
while ((counter < (raft_counter-1)) && intersecting
== 0)
{
double distance = sqrt (((raftx[raft_counter]-
raftx[counter])*(raftx[raft_counter]-
raftx[counter]))+((raftz[raft_counter]-
raftz[counter])*(raftz[raft_counter]-raftz[coun-
ter])));
if (distance < (raftradius[raft_counter]+raftra-
dius[counter]))
{
intersecting = 1;
}
counter++;
}
}
// Write raft variables to 0.xml file
fprintf(file, "<xagent>\n");
fprintf(file, "<name>Raft</name>\n");
fprintf(file, "<id>%d</id>\n", raft_counter);
fprintf(file, "<x>%f</x>\n", raftx[raft_counter]);
fprintf(file, "<y>%f</y>\n", minextracellular_y-
boundary);
fprintf(file, "<z>%f</z>\n", raftz[raft_counter]);
fprintf(file, "<radius>%f</radius>\n", raftradius
[raft_counter]);
fprintf(file, "</xagent>\n");
raft_counter++;
}

```

This code not only generates the raft agents but also writes the coordinates to arrays that can be used to restrict the receptor agents to individual lipid rafts. Next, to alter the coordinates of the receptors, locate the section headed by

```
/* Create RECEPTOR Receptor Agents */
```

Thereafter, in the while loop, we need to replace the declaration of the x - and z -coordinates to reflect the lipid raft localization. Delete the following lines:

```
double x = minintracellular_xboundary +
(rand()/(double)(RAND_MAX)*(maxintracellular_x-
boundary-
minintracellular_xboundary));
double y = minextracellular_yboundary;
double z = minintracellular_zboundary +
(rand()/(double)(RAND_MAX)*(maxintracellular_z-
boundary-
minintracellular_zboundary));
```

and replace it with the following:

```
int raft_number = (int) (rand()/(double)(RAND_MAX)
*20.0);
double r = 151.0;
double x, y, z;
y = minextracellular_yboundary;

// Set receptor coordinates
while (r > raftradius[raft_number])
{
x = raftx[raft_number] - raftradius[raft_number] +
(2 *
raftradius[raft_number] * rand()/(double)(RAND-
MAX));
z = raftz[raft_number] - raftradius[raft_number] +
(2 *
raftradius[raft_number] * rand()/(double)(RAND-
MAX));
// Check coordinates are within raft radius
r = sqrt(((x-raftx[raft_number])*(x-raftx[raft_-
number]))+((z-
raftz[raft_number])*(z-raftz[raft_number])));
}
```

Finally, we need to write the memory variable Raft_id to the 0.xml file. To do this, locate the line

```
fprintf(file, "<Dc>%f</Dc>\n", Dc_state_10);
```

and immediately after, add the following line:

```
fprintf(file, "<Raft_id>%d</Raft_id>\n", raft_number);
```

Now the 0Creator.c can be compiled, ran, analyzed, and visualized as before in Subheadings 3.2 and 3.3. If problems are encountered in compilation or simulation of the customized code, please see **Note 4** for troubleshooting advice.

4 Notes

1. Picking a reasonable diffusion coefficient

Picking a reasonable diffusion coefficient for intracellular and membrane proteins is not trivial in the absence of primary measurement (e.g., by fluorescence recovery after photo-bleaching (FRAP) or fluorescence correlation spectroscopy (FCS)) due to molecular crowding and nonlinearity between protein size and diffusion coefficients. A number of papers have tried to study the size diffusion coefficient relationship in different species, including in *E. coli* and human cells, and while subject to anomalous outliers, these can be used as a reasonable estimate in the first place [10–14].

2. Picking a reasonable time step

FaST uses discrete time steps for Brownian motion and reactions, with reactions being driven by the relative proximity of two molecules after diffusion. The choice of time step is a critical decision in maintaining the balance between efficient runtimes while retaining accuracy. Generally, the algorithms in FaST will be more accurate with a shorter time step but will also be slower. The choice of time step should be made dependent on the modelled system, considering the diffusion length, geometry, reaction rates, and concentration of reactants. Generally, the maximum diffusion length a molecule should take in a single iteration should be an order of magnitude below the characteristic length of the geometry. In FaST simulations, an option is included to maintain different time steps for Brownian motion and reactions. Brownian motion has a generally much lower computational burden than reactions, and so this enables the user to accurately model particle motion while retaining computationally efficient reaction time steps. For reactions, the binding radius and unbinding radius is calculated relative to the time step. Similar to the diffusion length, the binding and unbinding radii should be significantly smaller than the characteristic length of the geometry. Moreover, the assumptions used in FaST require that the probability of multiple potential reactants occurring within the binding radius of any giving molecule is very low. In the event of this occurring, the molecule will choose to bind to only one of the reactants. This is an acceptable correction when such events are rare;

however, when these events are relatively common, then FaST will begin to create slower kinetics compared to the original reaction rate used as an input. The likelihood of this happening is directly related to the binding radius and concentration of reactants. The user is advised to trial shorter time steps that fit the above considerations and increase it until unacceptable decreases in accuracy occur.

3. *FLAME GPU and Visual Studio Build Dependencies*

Coming into FLAME GPU as a relative novice (as I did myself), one of the first initial challenges is to get the varying versions of Visual Studios, CUDA toolkits, Windows SDKs, and VS C++ build tools to work seamlessly together to build the simulations. The examples included with the current FLAME GPU distribution, at the time of writing (FLAME GPU 1.5.x), were setup for compilation with CUDA 9.2, Windows SDK 8.1, and the C++ build tools MSVC v140. All simulations in this paper were compiled within Microsoft Visual Studio Community 2019 using CUDA 10.2, Windows SDK 8.1, and C++ build tools MSVC v140. We have had mixed results with using different versions of the SDK and build tools, with it working well the majority of times but with occasional issues in Release deployment. If you choose different versions, the dependencies must be changed to allow successful building and compilation. This can be done within Visual Studios itself through Properties → Configuration Properties → General (for SDK target and VS C++ build tools) and Build Dependencies → Build Customizations (CUDA version). Alternatively, these properties can be edited in the Visual Studios project file (.vcxproj). If you plan to use FaST and FLAME GPU a lot, I'd personally recommend making the required alterations to the EmptyExample as it can be used as a template for new models without the need to reconfigure the build settings each time.

4. *Customization Compilation Issues*

When compiling the custom code, errors might be encountered. This most likely stems from the code being copied into the wrong place. Particularly, within the XML file, some functions and memory variables are similarly or identically named between the Ligand and Receptor agent types. If you cannot identify precisely where a segment of code was meant to be placed, we have included a markup version of the edited files within the supporting depository (<https://doi.org/10.18419/darus-2175>). These files more clearly identify where the new code should be added.

References

1. Chuang H-Y, Hofree M, Ideker T (2010) A decade of systems biology. *Annu Rev Cell Dev Biol* 26:721–744
2. Ideker T, Galitski T, Hood L (2001) A new approach to decoding life: systems biology. *Annu Rev Genomics Hum Genet* 2:343–372
3. Bray SSA, Bray D (2004) Stochastic simulation of chemical reactions with spatial resolution and single molecule detail. *Phys Biol* 1:137–151
4. Andrews SS (2017) Smoldyn: particle-based simulation with rule-based modeling, improved molecular interaction and a library interface. *Bioinformatics* 33:710–717
5. Slepoy SJP, Slepoy A (2005) Microbial cell modeling via reacting diffusive particles. *J Phys Conf Ser* 16:305–309
6. Kerr R, Bartol T, Kaminsky B et al (2008) Fast Monte Carlo simulation methods for biological reaction-diffusion systems in solution and on surfaces. *SIAM J Sci Comput* 30:3126–3149
7. Pogson M, Holcombe M, Smallwood R, Qwarnstrom E (2008) Introducing spatial information into predictive NF-kappa B modelling – an agent-based approach. *PLoS One* 3:e2367
8. Klann MT, Lapin A, Reuss M (2011) Agent-based simulation of reactions in the crowded and structured intracellular environment: Influence of mobility and location of the reactants. *BMC Syst Biol* 5(1):1–14
9. Fullstone G, Guttà C, Beyer A, Rehm M (2020) The FLAME-accelerated signalling tool (FaST) for facile parallelisation of flexible agent-based models of cell signalling. *NPJ Syst Biol Appl* 6:10
10. Nenninger A, Mastroianni G, Mullineaux CW (2010) Size dependence of protein diffusion in the cytoplasm of Escherichia coli. *J Bacteriol* 192:4535–4540
11. Schavemaker PE, Boersma AJ, Poolman B (2018) How important is protein diffusion in prokaryotes? *Front Mol Biosci* 5:93
12. Kalwarczyk T, Kwapiszewska K, Szczepanski K et al (2017) Apparent anomalous diffusion in the cytoplasm of human cells: the effect of probes in polydispersity. *J Phys Chem B* 121: 9831–9837
13. Ramadurai S, Holt A, Krasnikov V et al (2009) Lateral diffusion of membrane proteins. *J Am Chem Soc* 131:12650–12656
14. Weiß K, Neef A, Van Q et al (2013) Quantifying the diffusion of membrane proteins and peptides in black lipid membranes with 2-focus fluorescence correlation spectroscopy. *Biophys J* 105:455–462

Part II

Advances in Integrative Analysis of Signalling Networks



Chapter 10

Modeling Cellular Signaling Variability Based on Single-Cell Data: The TGF β -SMAD Signaling Pathway

Uddipan Sarma, Lorenz Ripka, Uchenna Alex Anyaegbunam, and Stefan Legewie

Abstract

Nongenetic heterogeneity is key to cellular decisions, as even genetically identical cells respond in very different ways to the same external stimulus, e.g., during cell differentiation or therapeutic treatment of disease. Strong heterogeneity is typically already observed at the level of signaling pathways that are the first sensors of external inputs and transmit information to the nucleus where decisions are made. Since heterogeneity arises from random fluctuations of cellular components, mathematical models are required to fully describe the phenomenon and to understand the dynamics of heterogeneous cell populations. Here, we review the experimental and theoretical literature on cellular signaling heterogeneity, with special focus on the TGF β /SMAD signaling pathway.

Key words Heterogeneity, Mathematical modeling, Signal transduction, Cell-to-cell variability, Ordinary differential equations, Stochastic modeling, Single cells, TGF β signaling, SMAD transcription factors, Numerical simulation

1 Introduction

Heterogeneity is an implicit part of life that is observed on nearly all biological scales [1]. Perhaps the most widespread effect of heterogeneity is the creation of a wide spectrum of life-forms during evolution [2]. Primarily, heterogeneity aids robustness to a biological system in a fluctuating environment, such that, a broader niche in phenotypic traits is collectively achieved by a population of a species, which may then add to better chances of their survival [3]. However, the presence of heterogeneity is observed at even finer levels—individuals within one species exhibit unique properties, and further, genetically identical cells within the same (multi-cellular) organism utilizes heterogeneity to achieve distinct cell fate

Uddipan Sarma, Lorenz Ripka and Uchenna Alex Anyaegbunam contributed equally to this work.

decisions [1]. At the level of single cells, heterogeneity in expression of individual gene or protein (also called cell-to-cell variability) may lead to cell-specific responses to external cues and distinct physiological trajectories [1, 3].

In this chapter, we provide an overview over cellular heterogeneity in the context of intracellular signaling. We will specifically discuss how signaling in a heterogeneous cell population can be modeled in silico and will point out assumptions underlying these models. The chapter is divided in three parts: in the first section, we describe biological systems where signaling heterogeneity plays a role in cellular decision-making. Furthermore, we review evidence that signaling heterogeneity is mostly deterministic in nature and discuss the molecular sources of signaling heterogeneity. In the second part, we describe mathematical modeling frameworks that can be used to model heterogeneous cell populations and fluctuations in cellular signaling pathways. We mainly focus on deterministic modeling approaches using ordinary differential equations, in which heterogeneity is introduced by parameter sampling and discuss approaches for the quantitative fitting of single-cell models to experimental data. In the third part, we focus on the TGF β /SMAD signaling pathway that plays a key role in tissue homoeostasis and cell migration but also in diseases such as cancer. We review the literature on single-cell analysis of this pathway and demonstrate that key features of heterogeneous TGF β /SMAD signaling can be understood by mechanistic modeling. We then discuss our recent modeling work, in which we quantitatively described cellular subpopulations of TGF β /SMAD signaling and heterogenous signaling at the single-cell level. As an outlook for future research, we summarize how fluctuations in signaling pathways affect noisy downstream gene expression and decision-making.

1.1 Heterogeneity in Cellular Responses to External Cues

From early microscopic observations in cell culture, it became clear that not all cells respond identically to the same external stimulus. Intriguingly, it seems that not only genetic differences between cells contribute to heterogeneity but that nongenetic origins arising from stochasticity in cellular networks also play an important role. In recent years, evidence has accumulated demonstrating that genetically identical cells show differences in differentiation programs [4], drug resistance [5, 6], and viral pathogenesis [7]. In the following paragraphs, we provide a brief summary of biological systems where heterogeneity plays a functional role in cellular behavior.

Stress Responses in Unicellular Organisms Heterogeneity is an important part of cellular decision-making, as evidenced by stochastic cellular differentiation events, where parts of a cell population randomly enter a new fate. Well-known examples include bacterial stress responses (e.g., [8]). For instance, during bacterial

competence, external stress induces a regulatory program in *Bacillus subtilis* that allows cells to take up DNA from the environment, thereby priming them for adaptation to stress conditions. In line with a stochastic event, it was shown that the decision to become competent is dictated by random fluctuations in the master transcriptional regulator ComK [8]. In some cases, stochastic fluctuations in stress networks occur constitutively, i.e., even in the absence of external stress. This phenomenon, known as bet hedging, ensures that subsets of cell populations are prepared to rapidly respond to stressors, thereby ensuring survival of the population in case of external changes. In *Saccharomyces cerevisiae*, isogenic clonal populations display a range of growth rates and slow growth predicts resistance to heat killing. At the molecular level, Tsl1, a trehalose-synthesis regulator, is a key component of the observed resistance, and cell-to-cell variability in Tsl1 expression correlates with growth rate and predicts cellular survival in response to stress [9].

Cellular Differentiation Likewise, in eukaryotes, cell fate decisions during tissue development appear to occur stochastically in genetically identical cell populations, and this is thought to allow for a diversification of tissues. At the molecular level, evidence is accumulating that random fluctuations in the levels of signaling pathways and master transcription factors govern cell fates (Fig. 1a). For instance, in *Drosophila melanogaster* cell-to-cell variability in the expression of the transcription factor, spineless creates the retinal mosaic for color vision and is thus important for the spatiotemporal organization of the eye [10]. Moreover, subpopulations of clonally derived hematopoietic progenitor cells with low or high expression of a stem cell marker (Sca-1) were observed to be in dramatically different transcriptional states and gave rise to different blood cell lineages in multipotent murine hematopoietic cell line [4]. In developing mice, cell-to-cell variability in the expression of certain genes (e.g., Fgf4) was found to determine the inner cell mass (ICM) lineage segregation of the blastocyst [11]. Further evidence for nongenetic heterogeneity and its impact on animal development comes from genetic mutations with incomplete penetrance. These mutations cause physiological defects only in a subset of genetically identical animals. For instance, during *C. elegans* development, the expression level of elt-2, a self-activating transcription factor, is critical for intestinal cell-fate specification. Strong embryo-to-embryo heterogeneity in elt-2 expression and thus failure of intestinal development in a subset of embryos is observed if the upstream regulator skn-1 is inactivated by a mutation. Thus, the skn-1 mutant shows incomplete penetrance, likely because the lowered input signal shifts heterogeneous elt-2 expression to a range, where its fluctuations have profound effects on intestinal development [12].

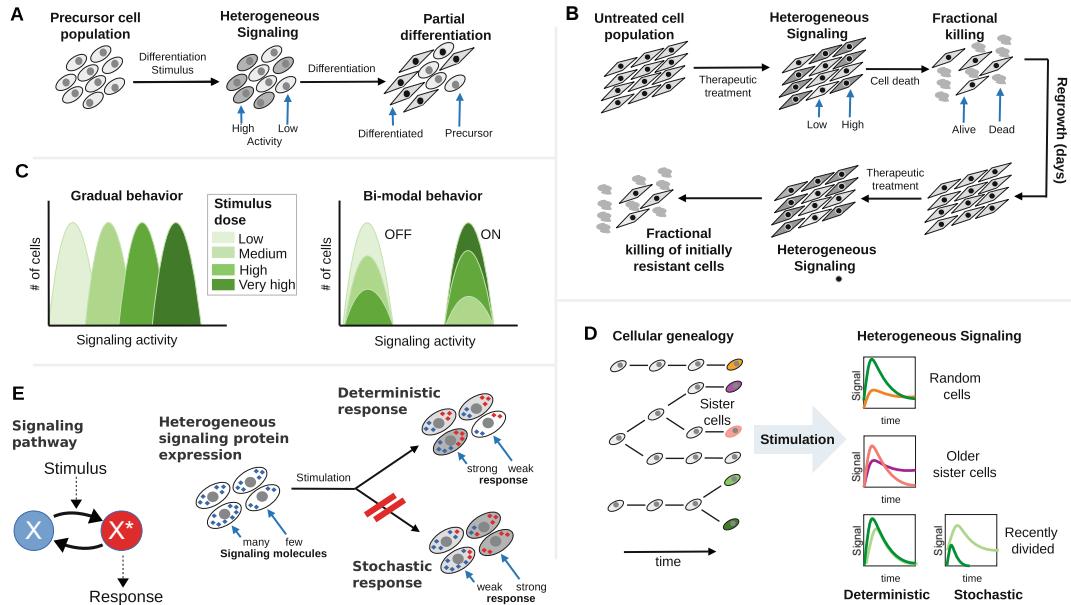


Fig. 1 Functional and physiological consequences of nongenetic heterogeneity **(a)** Heterogeneous signaling causes fractional cell differentiation. Only cells exhibiting high signaling activity (dark grey) in response to a differentiation stimulus undergo differentiation, e.g., during MAPK-induced oocyte maturation or PC12 differentiation (see text). **(b)** Signaling variability in response to therapeutic treatment. Apoptotic signaling pathways (e.g., caspases) are activated heterogeneously when cells are treated with a cytotoxic drug which results in fractional killing and (transient) resistance in the nonresponding cells. A population regrown from the therapy-resistant cells may again exhibit the same fractional killing, indicating that signaling heterogeneity is a nongenetic phenomenon. **(c)** Heterogeneous signaling may exhibit gradual or bimodal behavior. Gradual signaling pathways exhibit a unimodal activity distribution across single cells which are shifted to higher mean levels upon increasing stimulation. Signaling histograms at different doses typically overlap, which may give rise to inaccurate cellular information transfer. Signaling systems with bimodal behavior exhibit two clearly separable (“ON” and “OFF”) activity levels. Increasing stimulation does not affect the mean signaling activity of the ON and OFF subpopulation but shifts the fraction of cells in each class. **(d)** Sister cells experiments indicate deterministic behavior of signaling. For a signaling pathway with stochastic dynamics, even recently divided sister cells would show distinct (stochastic) signaling responses. In signaling, recently divided sister cells are typically more correlated in their signal response than random cells, likely because sisters show common protein expression patterns or cell cycle stages. Over a timescale of hours to days, sister cell similarity is lost (“older sister cells”), indicating a nongenetic mechanism of signaling heterogeneity. **(e)** Variations in signaling protein expression cause deterministic heterogeneity of signaling. A high total expression level of a positive regulator ($X_{tot} = X + X^*$) causes a strong response in a signaling pathway with deterministic behavior. In a signaling pathway with stochastic dynamics, the signaling response is less or not at all related to the protein content. In models of cellular heterogeneity, protein expression is often assumed to be stable at the timescale of signaling

Tumor Progression and Drug Resistance Cancer therapy aims for a complete eradication of tumor cells. However, in reality, complete killing is rarely achieved, as signaling pathways involved in the execution of cell death fail to be activated in all cells and certain subpopulations are therefore resistant to the therapeutic treatment

(Fig. 1b). Thus, cytotoxic drugs often result in fractional killing, especially when the drug concentration inside a tumor is limiting. In cell culture, fractional killing has been reported in response to a variety of treatments, including chemotherapy and apoptosis-inducing receptor ligands such as TRAIL [5, 6, 13–16]. At the molecular level, fractional killing involves cell-to-cell variation in cellular regulatory molecules, e.g., in the tumor suppressor p53 in response to chemotherapy [6]. Thus, nongenetic variability may confer resistance to therapeutic intervention and could play a role in tumor evolution [17].

These examples demonstrate that cellular heterogeneity can have a strong impact of cell fate decisions in biological systems. It is therefore crucial to quantitatively measure cellular heterogeneity using experimental methods such as live-cell imaging, flow cytometry, or single-cell RNA sequencing (reviewed in refs. 18–23). Furthermore, predictive modeling approaches describing the variability of molecular networks at the single-cell level will be essential to rationally manipulate cellular differentiation processes and to design effective combinatorial therapeutic intervention strategies.

1.2 Heterogeneity in the Activity of Cellular Signaling Pathways

The above examples of stochastic decision-making mainly focused on cellular fluctuations in nuclear transcription factors. Signaling pathways controlling these master transcription factors similarly show strong nongenetic cell-to-cell variation that is linked to cell fate. This was initially shown for mitogen-activated protein kinase (MAPK) signaling and later extended to other signaling systems.

In a pioneering study, the group of James Ferrell analyzed *Xenopus* oocyte maturation in response to the maturation-inducing hormone progesterone [24]. This maturation response is mediated by the MAPK signaling pathway, and due to the large size of oocytes, the authors could perform single-cell Western blot experiments to determine cell-to-cell variation in the activity of this pathway. They showed that MAPK signaling was activated in an all-or-none manner within individual cells, i.e., every oocyte either had low or high (but not intermediate) MAPK activity level, and MAPK activity therefore showed a bimodal distribution (Fig. 1c, right). This switch-like (ON or OFF) MAPK activation with strong heterogeneity between cells caused maturation in only a fraction of cells, and ON/OFF-switching was shown to arise from a positive feedback at the level of the signaling pathway [24]. Likewise, switch-like and heterogeneous MAPK activation was reported to be involved in other cell fate decisions such as T-cell activation [25], PC12 cell differentiation [26], and in the UV stress response mediated by the closely related JNK MAPK signaling pathway [27]. Thus, MAPK signaling frequently mediates switch-like and highly heterogeneous cell fate decisions. However, the pathway can also be gradually activated, with a unimodal but heterogeneous

distribution of activity states in the population (Fig. 1c, left), e.g., in EGF-stimulated fibroblasts. There, the pathway transmits quantitative information to the nucleus, and switch-like cell fate decisions are established at the level of downstream gene-regulatory networks [28, 29]. In conclusion, heterogeneity in MAPK signaling is a widespread phenomenon, but the qualitative features (uni- vs. bimodal distribution) of the heterogeneous population can vary, implying plasticity in network behavior.

Other signaling systems seem to be less flexible in their signaling output. For instance, the apoptosis signaling system, involved in sensing cytotoxic stress and death receptor signals, seems to invariably induce heterogeneous all-or-none responses at the level of the executing caspase enzymes (e.g., [30]). This may ensure that programmed cell death is executed completely and irreversibly, but only in a fraction of cells in a tissue, thereby preventing complete loss of all cells in a tissue. Yet other signaling pathways such as Akt [31, 32] and TGF β /SMAD [33, 34] signaling show a heterogeneous but gradual (unimodal) response in the majority of cellular systems studied. Thus, these signaling pathways transmit quantitative information about extracellular stimulus concentration from the cell membrane to the nucleus, where gene-regulatory networks mediate cell fate decisions.

One important question in signal transduction is how accurate information transmission is possible despite strong heterogeneity in the activity of gradual signaling pathways. Based on a combination of quantitative experiments, mathematical modeling, and concepts from information theory, it became clear that quantitative information is frequently encoded in the temporal dynamics (i.e., the shape) of the signal [35–38]. For instance, the NFkB signaling pathway shows oscillatory dynamics, and the nature of activated target genes in the nucleus depends on the frequency and amplitude of the signaling pathway oscillation [39]. Another concept for reliable signal transmission despite signaling heterogeneity is relative signal transmission, where the (noisy) absolute signaling activity is less important for cellular behavior when compared to the (more robust) stimulus-induced fold change over basal [29, 40–43]. Combined live-cell imaging and modeling studies support this concept for Erk, Wnt, and TGF β signaling pathways, and it has also been described how downstream gene-regulatory networks in the nucleus could respond to fold changes rather than absolute signaling levels [40–42].

Taken together, both gradual and bimodal signaling pathways show strong cell-to-cell variability in their activity. Single-cell experiments characterize quantitative information transmission and switch-like decision-making in a heterogeneous cell population and therefore provide a basis for quantitative modeling.

1.3 Signaling Fluctuations Are Often Nongenetic and Temporally Stable

To model signaling heterogeneity, assumptions need to be made about the properties and origins of the fluctuations. Evidence from the literature suggests that signaling heterogeneity is nongenetic and that the cell-specific features of signaling can be assumed to be temporally stable during a typical cellular stimulation experiment.

Cell cultures are often derived from tumor cells and are thus potentially genetically unstable. This raises the question of whether heterogeneity in signaling pathways is really nongenetic in nature. Strong evidence for a nongenetic contribution to heterogeneity in signaling events comes from restimulation experiments with ligands and drugs triggering cell death (Fig. 1b). In several of these studies, an initial treatment killed the majority of the cell population, and the cells were kept in culture for several days before they were subjected to another treatment with the same stimulus. Even though the initial survivors were resistant to the first treatment, a large part of them became sensitive to the second treatment (Fig. 1b). Thus, resistance was not inherited to all offspring of resistant cells, i.e., genetically determined, but was gradually lost during several days of culture, arguing for an epigenetic mechanism of heterogeneity [5, 13, 44, 45].

A second line of evidence for a nongenetic signaling heterogeneity came from sister cell experiments, in which cells and their division events were tracked before stimulation to record cellular progeny (Fig. 1d). A common observation in these lineage tracing experiments is that freshly divided sister cells show very similar signaling responses and that the similarity of sisters got lost over time after the common division event. For instance, for TRAIL-induced apoptosis, it was shown that freshly divided sister cells have a high correlation in the death time, i.e., the time it takes for the TRAIL stimulus to induce cell death [46]. Thus, the experiment suggests that sister cells are initially in a very similar (signaling) state, but interestingly they lose the signaling similarity with a characteristic half-life of 11 h (after the common division event). Similar observations were made for other regulatory networks, including the cell cycle, the spindle assembly checkpoint, MAPK, as well TGF β signaling [33, 47–50]. Sister cell experiments have several important implications for cellular signaling heterogeneity and mathematical modeling thereof:

1. First, they further support nongenetic sources of heterogeneity, since the loss of sister cells similarity on a timescale of hours is much faster than any genetic drift due to DNA mutations.
2. Second, the initially high sister cell similarity suggests that heterogeneity does not arise from stochastic dynamics in signaling reactions: If signaling molecules are present in very low amounts, signaling reactions (e.g., phosphorylation) could be probabilistic events that give rise to strong heterogeneity and high dissimilarity in the signaling events of freshly divided

sisters (or even high dissimilarity if the same cell would be stimulated repeatedly). Certain signaling pathways indeed show such stochastic dynamics (*see* Subheading 2.1.3 and [51]), but the sister cell experiments suggest that this is typically not the case, which agrees with the fact that signaling proteins are often expressed at high molecule numbers [52].

3. Third, the timescale of sister cell similarity (multiple hours) suggests that signaling heterogeneity can be assumed to be stable at the timescale of a typical stimulation experiment (minutes to hours). The conclusion of temporal stability is also supported by a recent restimulation analysis with insulin-like growth factor, in which the rank of single cells with respect to Akt signaling activity was stable over several hours [32].

These conclusions greatly simplify the mathematical modeling of cellular signaling pathways, as stochastic modeling of signaling dynamics can be neglected in most cases. Instead, a deterministic ordinary differential equation approach can be used, in which certain kinetic parameters are assumed to be different between cells, but stable over time.

1.4 Signaling Heterogeneity Arises from Fluctuations in Signaling Protein Levels

To introduce heterogeneity into a mathematical model, we need to know the molecular sources of temporally stable signaling pathway fluctuations. Obvious sources for signaling fluctuations are cell-to-cell differences in cell cycle stage [19] or cell density [53]. In growing adherent mammalian cells, cell divisions combined with cell motility can create variations in local cell densities, cell-cell contacts, relative location, and the amount of free space per cell. Combined, these parameters constitute the population context of an individual cell [54, 55]. However, single-cell signaling studies exist where these sources have been excluded experimentally, but the heterogeneity in the signaling output persists [33, 48, 51].

In recent years, it has become apparent that random fluctuations in the total cellular concentrations of signaling proteins may underlie temporally stable cell-to-cell variability in signaling pathway activity (Fig. 1e). In some cases, signaling heterogeneity could be traced back to fluctuations in the expression of specific signaling molecules, e.g., in MAPK [25], PI3K/Akt [56], and JAK/STAT [57] signaling. For instance, a pioneering work on T-cell activation showed that MAPK activation depends on cell-to-cell variability in the expression of the CD8 co-receptor and the antagonizing phosphatase SHP-1. Interestingly, even though each protein contributes to variability, co-regulation of CD8 and SHP-1 expression levels at the single-cell level limits diversity and promotes robustness of signaling [25]. In other systems, signaling heterogeneity cannot be explained by fluctuations in a single protein, but the control seems distributed over many protein levels. For example, in their work on apoptosis signaling, Spencer et al. showed that no

single protein level could accurately predict cell death timing in response to TRAIL treatment, unless a specific pathway regulator (BID) was overexpressed and its cellular level then became a good predictor of cell death timing [46].

Even in genetically identical cell populations, (signaling) proteins show strong cell-to-cell variability in their levels, since epigenetic control and gene expression are stochastic events at the single-cell level (*see also Subheading 2.1.3*): each gene is present only at two copies (alleles) per cell and gene regulation at such low molecule numbers is probabilistic [58–61]. As a consequence, each (signaling) protein level follows a log-normal distribution, that is, the fold change of a protein is normally distributed around its population mean [46, 58, 62]. Although the mean and standard deviation of the distribution are protein-specific, a typical human protein shows a threefold difference in expression across the cells of a population (where the fold change is measured between the 10th and 90th percentile of the distribution) [47, 58]. This strong variation in each and every signaling protein leads to strong fluctuations in signaling. Time-resolved measurements further suggest that fluctuations in signaling protein expression and pathway activity are coupled: when protein expression fluctuations in mammalian cells are followed over time, the timescale of stochastic changes in protein expression is similar to the timescale of signaling pathway desynchronization between sister cells (multiple hours to days) [46, 58]. This is consistent with a model, in which sister cells initially share the same proteome content and signaling activity, and then over time and simultaneously lose both types of similarity due to stochastic gene expression fluctuations [46, 47, 58].

Taken together, these observations indicate that mammalian signaling proteins show strong fluctuations in their levels. In the following, we will discuss deterministic modeling approaches of cellular signaling pathways in which cell-to-cell variability in signaling protein concentrations is taken into account.

2 Methods

2.1 Mathematical Modeling of Cellular Heterogeneity

Signaling pathways often respond in very similar manner to stimulation when the same cell is stimulated repeatedly, or when sister cells are in a similar state and harbor a similar proteome content. These observations suggest that cells respond deterministically to stimulation and that deterministic mathematical modeling approaches can be used to simulate cellular heterogeneity *in silico*. Such deterministic models are often based on ordinary differential equations (ODEs) which represent reaction networks within the cell, typically using mass action-based reaction kinetics (reviewed in ref. 63). ODE models assume that the biochemical molecules in the cell are present in sufficiently large amounts (and well-stirred), so

that stochastic fluctuations at the single-molecule level can be neglected and the biochemical species can be described using continuous variables, representing average molar signaling protein concentrations within the cell. If an ODE system is simulated twice with the same set of kinetic reaction parameters and initial conditions (i.e., protein concentrations), it will yield exactly the same solution, representing the deterministic nature of the approach. This determinism is in contrast to stochastic simulation algorithms (reviewed in ref. 64) which explicitly describe single-molecule fluctuations and therefore give rise to distinct simulation results for each realization.

In this section, we discuss how intracellular signal transduction in a heterogeneous cell population can be modeled in silico. We mainly focus on deterministic ODE-based modeling and point out how these models can provide insights into several aspects of cellular heterogeneity, including noisy decision-making and the robustness of signaling networks. We then discuss how these models can be calibrated based on single-cell data to obtain a quantitative match between experiment and theory. Finally, we briefly summarize stochastic modeling approaches that are relevant for the modeling for some signaling networks operating at low molecule numbers and particularly for gene-regulatory networks involved in cell fate decisions downstream of signaling pathways.

2.1.1 Applications and Limitations of Population-Average Models

Experiments aimed at understanding intracellular processes were traditionally performed in bulk, combining material from thousands to millions of cells. For instance, signaling pathway dynamics were studied using Western blot experiments, in which phosphorylated signaling intermediates are detected using phospho-specific antibodies [24, 65]. Due to averaging over a large number of cells, these experiments do not provide information about single-cell heterogeneity but only represent the behavior of one hypothetical average cell. In systems biology, early quantitative models were built based on the available population-average data and therefore describe one representative cell (Fig. 2a).

Even though not meant to represent cellular heterogeneity, population-average models can provide important insights into several signaling phenomena at the single-cell level including cellular decision-making and robustness of networks against fluctuations in their components. For instance, Ferrell et al. (1998) showed that the decision of Xenopus oocyte maturation in response to progesterone involves an all-or-none biological response at the level of MAPK signaling (*see* Subheading 1.2; [24]). To better understand this single-cell phenomenon, the authors constructed a population-average model of the signaling network and concluded that switch-like (bistable) behavior in the MAPK cascade arises from a positive feedback loop that amplifies the signal once MAPK signaling exceeds a certain threshold. Likewise, other population-average

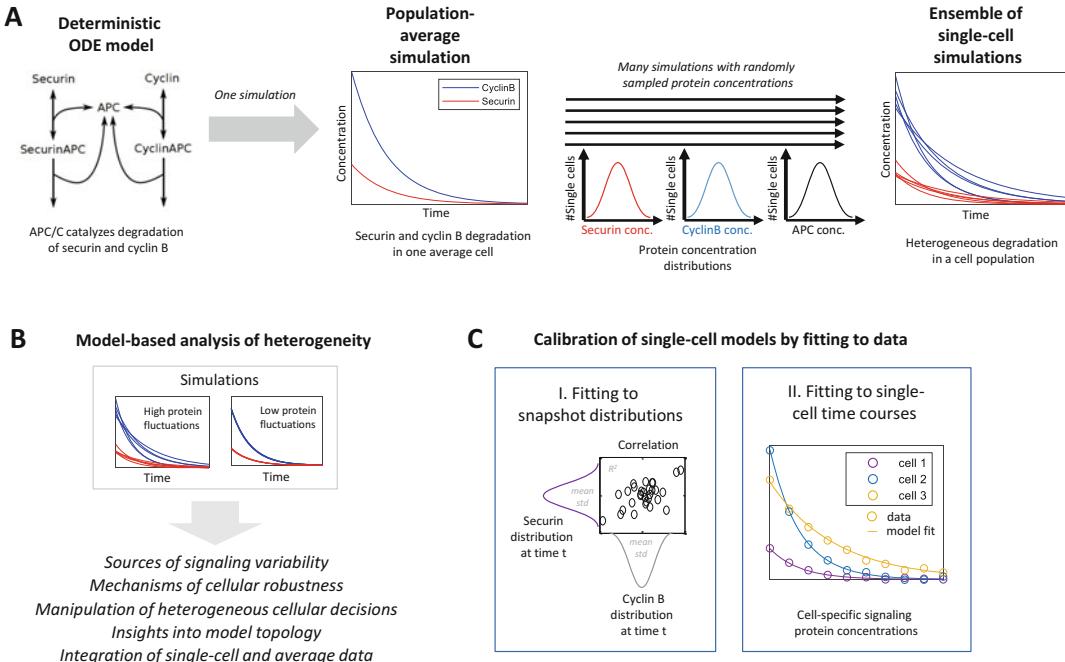


Fig. 2 Deterministic modeling of cellular heterogeneity **(a)** A kinetic model of securin and cyclin B degradation by the anaphase-promoting complex (APC) during mitosis (left) is described by a deterministic ODE model (see also ref. 66) and numerically integrated to simulate the protein dynamics in one average cell (middle). Heterogeneity is introduced into the system by performing repeated simulations while sampling protein concentrations (and in some cases kinetic parameters) from log-normal distributions (right). **(b)** Model-based analysis of cellular heterogeneity. For model analysis, simulations are performed for varying parameters and/or degree of protein concentration fluctuations, or different biological mechanisms are considered in the model. Thereby, the model provides insights into molecular sources of heterogeneity, mechanisms of biological robustness, and allows for the design of new experiments. **(c)** Quantitative fitting of ensemble models to heterogeneous single-cell data. In the literature, models of cellular heterogeneity were calibrated by fitting cross-sectional snapshot data at a particular time point or by directly fitting the kinetic model to single-cell trajectories. In both cases, cell-specific parameters are estimated to yield an optimal match between model and experiment

modeling studies provided insights into mechanisms of switch-like decision-making and therefore have implications for bimodal behavior at the single-cell level [26, 67–72]. Population-average models further provided insights into biological robustness against fluctuations in signaling protein concentrations [25, 66, 73–75]: For example, using single-cell experiments, Kamenz et al. (2015) observed that the timing of mitotic events is highly robust and is buffered against variations in the concentrations of mitotic regulatory proteins [66]. A population-average model could explain the observed robustness and predicted conditions where mitotic timing is compromised. Thereby, the population-average model identified critical transitions in the network, and experimental validation showed that these transitions led to network failure in a subset of

cells due to cell-to-cell variability in the molecular components. In general, for gaining insights into robustness, a so-called sensitivity analysis can be performed, in which the initial conditions and kinetic parameters are systematically perturbed (usually one at a time) to understand their role in the behavior of the network. Given that cellular signaling fluctuations often arise from cell-to-cell variability in signaling protein expression (Subheading 1.4), sensitivity analyses focusing on the impact of signaling protein concentrations at the population level can provide valuable insights into biological variability and robustness in single cells [25]. However, though potentially useful, population-average models do not directly represent signaling distributions across single cells (as those shown in Fig. 1c) and therefore do not allow for a quantitative comparison of the model simulations to a single-cell experiment. Furthermore, a population-average model constructed based on population-average data may lead to misleading conclusions for cell fate decision networks with ON/OFF-behavior [24, 62]: In binary decision-making, single cells show all-or-none (digital) signaling but do so heterogeneously for a given stimulus concentration. Hence, the activity distribution is bimodal (Fig. 1c), and the mean response of the population gradually increases with increasing stimulation. Therefore, at the population-average level, digital responses may appear gradual (analogue) when averaged.

Taken together, population-average models fail to quantitatively describe single-cell distributions and can only be as insightful as the experimental data they are based on. Models that are designed solely based on population-average data implicitly assume that the population-average data is a good approximation for the true underlying single-cell behavior, and therefore may lead to wrong conclusions. To overcome these limitations, models of cellular heterogeneity were developed to quantitatively describe the cell-to-cell variability in signal transduction.

2.1.2 Deterministic Models of Signaling Heterogeneity: Implementation and Scope

How does a deterministic model of cellular signaling heterogeneity look like? In most cases, heterogeneity is simulated using an ensemble of single-cell models (Fig. 2a). Here, individual cells are described by repeating the simulation using the same deterministic ODE model, each simulation run corresponding to one cell, and cell-to-cell variability is introduced by perturbing the model in each realization.

Based on the arguments presented in Subheading 1.4, the perturbation leading to cell-to-cell variability is mainly introduced by assuming cell-specific signaling protein concentrations. Additionally, kinetic parameters can be assumed to be cell-specific if the corresponding reaction rates are in turn controlled by (fluctuating) proteins (this may, e.g., be true for enzymatic reactions). Given that protein expression levels are log-normally distributed (see

Subheading 1.4), single cells are modeled by repeated simulations in which all protein concentrations in the model are sampled from independent log-normal distributions (Fig. 2a). In this approach, sometimes termed Monte-Carlo sampling or nonlinear mixed effect modeling, the protein fluctuations are typically restricted to a biologically reasonable range. Specifically, the coefficient of variation of the lognormal distribution ($CV = \text{std}/\text{mean}$) is chosen between 0.1 and 0.4 [46, 58, 62]. Since proteins from the same pathway may be co-regulated at the single-cell level [58], sometimes correlated fluctuations in signaling protein fluctuations are assumed [33]. Notably, when sampling only the initial total protein concentrations, while leaving the model otherwise unperturbed, one assumes that nongenetic sources of signaling heterogeneity are temporally stable during pathway stimulation. Hence, the modeling framework captures the key features of cellular signaling heterogeneity, including deterministic behavior, temporal stability, and protein concentrations as a noise source (Subheading 1).

Compared to a population-average model, such single-cell ensemble modeling approaches reproduce the complete heterogeneous cell population and allow for a quantitative comparison of the model to experimental single-cell data. Specifically, while the population-average model by definition only represents the mean, ensemble models capture higher momentum statistics of the heterogeneous model species such as the standard deviation, their correlations or time course features such as the autocorrelation function (e.g., [46]). Deterministic ensemble modeling approaches have been used in several studies on signaling, often in combination with experimental analyses at the single-cell level [14–16, 35, 41, 46, 49, 62, 71, 73, 76–87]. These models provided experimentally testable predictions and led to a better understanding of heterogeneous signal transduction. In particular, the following phenomena were analyzed (Fig. 2c):

1. *Sources of signaling fluctuations:* Ensemble models were used to characterize how fluctuations in individual signaling protein expression levels affect the signaling outcome. Thereby, the most critical signaling protein expression fluctuations could be identified as main sources of cell-to-cell variability in signal transduction [16, 33, 46, 49, 57, 87–89]. This led to a better understanding of molecular mechanisms causing heterogeneous cellular decision-making.
2. *Design principles of biological robustness:* Robust and reliable signal transduction must occur despite strong noise. By adding or removing certain reactions in the models, insights were gained into design principles that mitigate signaling variability [41, 62, 73, 80], thereby promoting robustness, e.g., during embryonic development [73]. Several robustness-promoting network motifs could be identified including negative feedback

[73, 80], fold-change detection [41], and correlated expression fluctuations in positive and negative regulators of signaling pathways [80, 86].

3. *Characterization and manipulation of heterogeneous decision making:* For cellular differentiation and during stress responses, cell-to-cell variability may be beneficial, as not all cells of a heterogeneous population enter a new fate and die in response to stress, respectively (see Subheading 1.1). Modeling of signaling pathways involved in cellular differentiation and cell death allowed for a quantitative analyses of decision-making at the level of signaling and thus for the emergence of bimodal signaling distributions. The models yielded predictions for the reprogramming of cell fates for novel experimental conditions [83, 84] and allowed for the optimization of therapeutic treatment responses in cell culture [14–16, 82]
4. *Insights into alternative biological mechanisms and network topologies:* Since certain network motifs affect the characteristics of biological fluctuations (see above), attempts were made to infer (reverse engineer) the wiring of signaling networks based on single-cell data. The idea is that signaling fluctuations contain a fingerprint for the underlying molecular interactions and that the model topology that best describes the signaling fluctuations is the most probable one. Several studies used a defined model topology and compared a set of relatively minor modifications in the model against single-cell data [85, 90]. In a less biased top-down approach, Sachs et al. inferred the topology of signaling networks from single-cell data without prior knowledge using a Bayesian framework [91].
5. *Integration of single-cell and population-average data:* Ensemble models of cell populations allow for simulations at both the single-cell and population-average levels and can thus be used to integrate both types of data [33, 57, 85]. Thereby, the models on the one hand exploit highly informative single-cell data which often can be done only at low throughput and for few molecular species (especially for time-resolved live-cell imaging). On the other hand, they take into account population-average information that can more easily collected for multiple experimental conditions and molecular species. Accordingly, the integration of population-average and single-cell data led to a better discrimination of competing model hypotheses when fitting an ensemble model to experimental data [85].

2.1.3 Stochastic Modeling of Signaling and Gene Expression Heterogeneity

Signal transduction cascades typically control cellular decisions by activating gene expression responses in the nucleus. Expression of target genes (e.g., cell cycle regulators or cell adhesion molecules) then controls the morphological features of a cell such as cell division and migration. In addition, target genes often act as negative feedback regulators that downregulate the signal once gene expression has been activated [52]. Thus, signaling and gene expression responses are intimately connected, and both may need to be taken into account in realistic models of cellular decision-making. In this context, it should be pointed out that deterministic models may no longer be suitable for modeling of cellular heterogeneity if gene expression is taken into account, e.g., for modeling transcriptional feedback or nuclear propagation of the signal.

The reason is that gene regulation is an intrinsically stochastic process with strong temporal fluctuations (reviewed in ref. 59), although deterministic sources of heterogeneity (i.e., the cellular state) also seem to play a role [54, 60]. Stochastic behavior arises from the fact that transcriptional regulators are typically expressed at very low levels and that a cell contains only two copies of each gene. As a result, random (Brownian) fluctuations at the level of individual reactions are not averaged out and significantly impact on the activity of a gene, especially at the level of mRNA production. Therefore, stochastic approaches are typically used for modeling heterogeneity of gene expression [59, 64]. In early work, Arkin and colleagues used the Gillespie algorithm to simulate biochemical reactions leading to gene expression and predicted stochastic cell-to-cell variation in protein numbers for biologically realistic parameter ranges [92, 93]. The prediction of stochastic mRNA and protein expression was later confirmed experimentally in bacteria and mammalian cells (reviewed in ref. 59). In higher organisms, noise seems to be larger in magnitude compared to bacteria, since chromatin states seem to give rise to switching of genes between ON and OFF states. In time courses of single-cell gene expression, this is observable as transcriptional bursts, i.e., episodes of high gene expression that are separated by phases with low activity [60, 61, 94]. The simplest stochastic model which realistically describes transcriptional bursts is the so-called random telegraph model, where a gene promoter is assumed to reversibly switch between a transcriptional active and an inactive state (reviewed in ref. 59). Notably, depending on the gene under consideration, more promoter states may need to be taken into account to describe the data [94, 95]. To jointly model signal transduction and gene expression, these stochastic promoter models were coupled to deterministic models of signaling, and this yielded insights into the dynamics of target gene expression [96, 97] and into the long-term regulation of signaling heterogeneity by stochastic signaling protein expression fluctuations [82].

In certain cases, intrinsic stochastic dynamics may arise within the signaling network, especially if the pathway operates at low molecule numbers [51, 97–99]. The level of initial signal sensing may be especially prone to stochastic dynamics, since cell surface receptors are often only expressed at a few hundred or thousand molecules per cell [52, 100]. Then, processes like receptor endocytosis which simultaneously remove hundreds of molecules at once from the cell surface, may give rise to digital behavior and consequently strong stochastic fluctuations of signaling activity [99, 101]. Accordingly, stochastic models of signaling networks have been proposed to describe heterogeneous decision-making [51, 96, 98, 100], and several of these studies focused on fluctuations at the receptor level [96, 98–100].

2.1.4 Quantitative Modeling of Cellular Heterogeneity

In many cases, ensemble modeling approaches are semiquantitative in the sense that the kinetic reaction parameters and the protein fluctuations (i.e., the standard deviation of their distribution) are tuned manually. While such semiquantitative modeling is valuable in many cases, the long-term goal is a quantitative match between model and experiment. This can be achieved by directly fitting the single-cell models to single-cell data by minimizing the difference between the simulated and measured single-cell distributions.

Quantitative single-cell model fitting of signaling and gene expression has now been applied in a number of publications and is a lively area of research [57, 60, 85, 86, 88–90, 94, 102–111]. In several of these studies, the fitted models involve deterministic sources of heterogeneity [85, 86, 107], stochastic fluctuations [94], or a combination of both [60, 88, 105]. In the deterministic case, only signaling protein concentrations may be cell-specific parameters [85] or all model parameters may show cell-to-cell variability [89]. For a comprehensive overview over the assumptions and the computational methods, we refer to the recent review by Hasenauer and Loos [112].

The methods for model calibration can be classified based on the type of experimental data they use, single-cell snapshot or time course data (Fig. 2c and [112]). For snapshot data, only distributions at single time points are considered, and therefore potential correlations between observations at consecutive time points are neglected. Despite this limitation, the approach benefits from the fact that snapshot data can typically be generated on a higher throughput, i.e., for more cells and molecular species, when compared to time-resolved measurements. For instance, high-throughput snapshot data can be generated using flow cytometry, mass cytometry, or single-cell RNA sequencing, and the larger wealth of information should be beneficial for the training of reliable mathematical models. Accordingly, several approaches were proposed for the model calibration based on snapshot data

[88–90, 102, 104, 111]. In deterministic models, different assumptions were made about the type of fluctuations in parameter values, ranging from the unimodal log-normal distribution to multimodal distributions, or even no specific assumption was made about the nature of fluctuations (nonparametric distribution; reviewed in ref. 112). For instance, Hasenauer et al. employed multimodal mixtures of normal parameter distributions to infer subpopulations (with distinct mean parameters) by fitting a model of NGF signaling to snapshot data [102].

A limitation of the snapshot approach is that essential information about temporal behavior in single cells gained from live-cell imaging (e.g., an oscillatory pattern) may be lost. Therefore, snapshot information may be less well suited for the identification of molecular sources of heterogeneity when compared to time-resolved data (discussed in ref. 110). As a consequence, several studies suggested to directly fit a mechanistic model to single-cell time course data [85, 86, 103, 105–110]. In a naive approach, each individual cell could be fitted separately by minimizing the residuals between model and data, and then the single-cell fitting results are combined to yield cell population distributions of interest, e.g., for signaling protein expression levels. In this so-called standard two-stage approach, each cell is thus analyzed as an independent subproblem (stage 1), and then the cell population is assembled (stage 2). However, stage 1 suffers from the problem that the model parameters in systems biology models can almost never be correctly estimated (identified), especially based on live-cell imaging data which typically only covers one molecular species for each cell. Thus, the fitting uncertainties are high, and the heterogeneity between cells is overestimated [106, 107], which limits the predictive power of the two-stage approach unless very small models are considered.

To circumvent this problem, information about the cell population distribution needs to be taken into account during fitting of single cells [85, 106, 107, 110]. Specifically, the fitted likelihood function combines information from all cells, and these additional constraints improve the identifiability of single-cell parameters which leads to smaller uncertainties in model predictions. For instance, as a constraint in deterministic models, it can be ensured that protein concentration fluctuations follow a log-normal distribution [85, 106]. Moreover, the model can be simultaneously fitted to single-cell and population-average data, and it has been shown that the combination of both types leads to a better discrimination of model variants compared to the use of either alone [85].

It should be noted that the current quantitative models of cellular signaling and gene expression heterogeneity are typically limited to a few species and reactions. Therefore, qualitative ensemble modeling approaches (Subheading 2.1.2) are still very valuable

for large-scale networks and typically led to more profound “biological” insights when compared to quantitative approaches. Further improvements are needed in the computational methods for quantitative model fitting to reduce computational cost and to integrate various types of data including cross-sectional snapshots, high-resolution live-cell imaging, and population-average data. This will improve identifiability of model parameters, the certainty of model predictions and will be helpful to discriminate competing model variants also in larger networks.

2.2 Heterogeneity in TGF β Signaling: Modeling and Impact on Cellular Behavior

2.2.1 TGF β Signaling in Health and Disease

In the final part of the chapter, we discuss the characteristics and modeling of TGF β /SMAD signaling at the single-cell level. We initially start with an overview over the pathway and its role in controlling cell fates. Then, we summarize its dynamic features at the single-cell level and outline how population-average as well as single-cell modeling approaches provided insights into the pathway dynamics. Finally, we review recent work, in which the link between fluctuations in SMAD proteins and target gene expression was explored.

TGF β belongs to a family of soluble extracellular ligands that activate intracellular signaling by binding to cell surface receptors. As depicted in Fig. 3a, signaling is initiated by a cascade of events that involves TGF β binding to the TGF β R2 receptor, and this receptor-ligand complex in turn binds to the TGF β R1 receptors (also known as ALK5) to build an activated receptor complex [113]. The active TGF β receptor functions as an intracellular kinase that phosphorylates cytoplasmic SMAD2/3 proteins which upon phosphorylation form heterotrimers with SMAD4 (e.g., (SMAD2)₂(SMAD4)). SMAD heterotrimers translocate to the nucleus and there act as transcription factors, i.e., they bind to and activate gene promoters to regulate the target gene expression [114]. The signaling pathway activity is terminated by nuclear dephosphorylation of SMAD proteins, dissociation of the complexes, and finally the nuclear exit of SMAD proteins.

TGF β induces several cellular responses including cell cycle arrest, apoptosis, and cell migration [115]. TGF β -induced cell migration typically involves the so-called epithelial-to-mesenchymal transition (EMT), a phenotypic remodeling of cells in which the cytoskeletal reorganization and loss of cell-cell junctions allows epithelial cells to evade from their original location by acquiring a motile, migratory, mesenchymal phenotype [116]. Given these widespread roles in cellular remodeling, it is not surprising that TGF β and closely related ligands (e.g., GDF11 or BMPs) play a critical role in embryogenesis and tissue homeostasis but also in diseases such as cancer or fibrosis [117]. For instance, in higher vertebrate development, gastrulation and neural crest formation depend on EMT induced by TGF β superfamily members

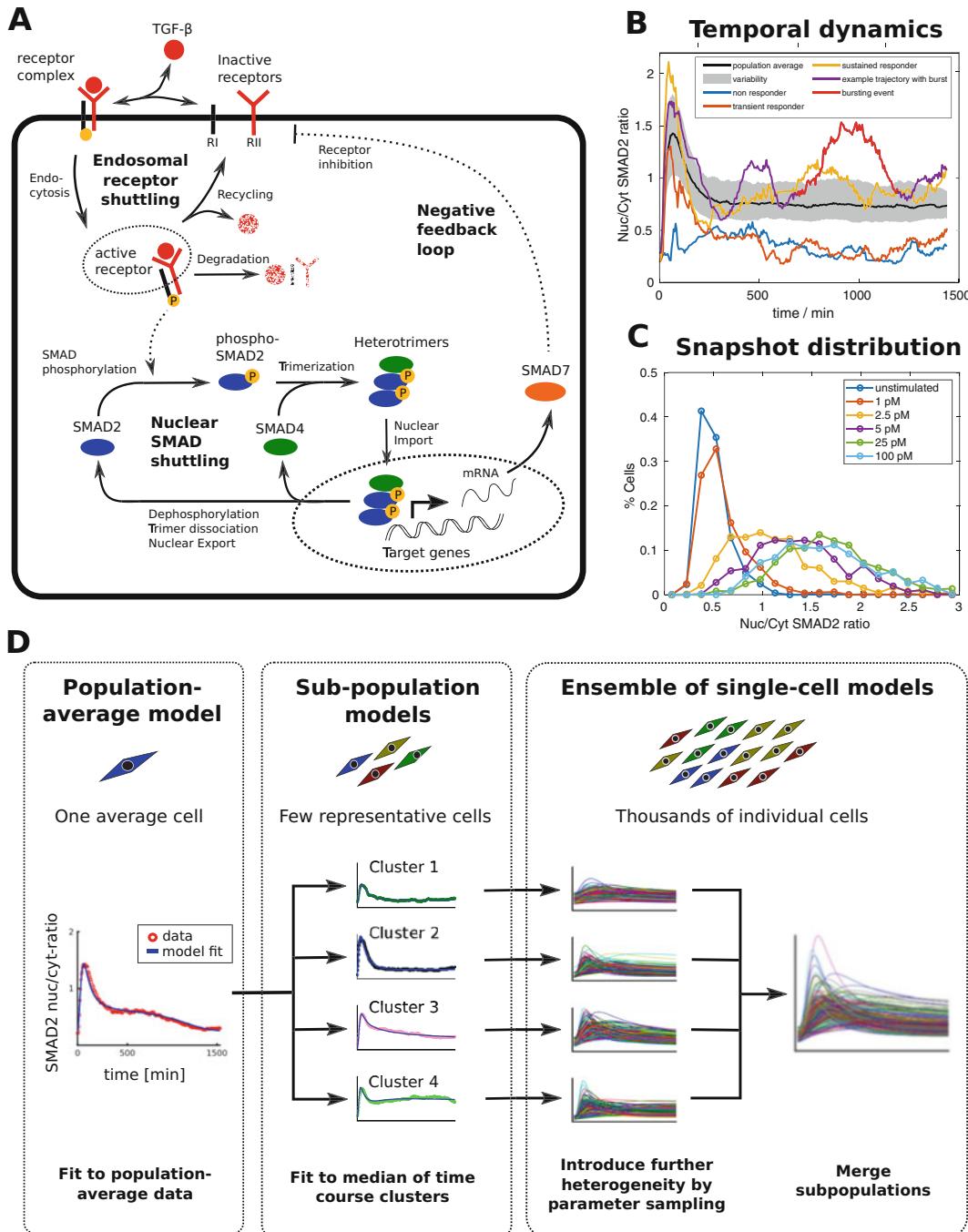


Fig. 3 Quantitative analysis and modeling of TGF β /SMAD signaling heterogeneity (a) Schematic representation of TGF β signaling including endosomal receptor shuttling, nuclear SMAD shuttling, and negative feedback: The binding of the ligand to a RII receptor leads to the recruitment of an RI receptor building an activated receptor complex. This complex can, when internalized, mediate the phosphorylation of SMAD2 proteins. The receptor complex either gets degraded or free receptors are recycled back to the cells surface. In the cytosol, phosphorylated SMAD2 proteins form trimers with SMAD4 which translocate to the nucleus, thereby increasing the experimentally measurable nuclear-to-cytoplasmic (Nuc/Cyt) SMAD2 ratio. In the nucleus, the SMAD heterotrimer acts as a transcription factor to induce downstream target genes of TGF β including

[116]. Likewise, TGF β -induced apoptosis and cell cycle arrest maintain tissue homeostasis and prevent overgrowth in developing and adult tissues, e.g., in the liver [115]. If the cytostatic effect of TGF β or its ability to induce apoptosis is lost, this leads to tumor progression. However, TGF β signaling not only acts as a tumor suppressor but plays a dual role in cancer progression, as in late-stage tumors aberrant TGF β -induced EMT signaling promotes the formation of metastasis [118]. Thus, during cancer development, a specificity switch occurs, in which TGF β signaling no longer promotes cytostatic responses but mostly induces cell migration.

At the molecular level, this specificity switch involves a change in the set of target genes regulated by TGF β /SMAD signaling: In late-stage tumors, TGF β no longer downregulates growth-promoting oncogenes like Myc and fails to upregulate cell cycle inhibitors (e.g., p15, p21). Instead, TGF β induces gene expression changes that are crucial for mediating early steps of reprogramming from epithelial to mesenchymal identity including the downregulation of classical epithelial and upregulation of mesenchymal markers [118]. Based on experimental evidence, several hypotheses have been proposed to explain how the same signaling pathway can induce qualitatively distinct gene expression responses depending on the cellular context: (1) context-specific expression of transcription cofactors involved in SMAD-dependent gene expression [119], (2) alterations in the concentrations of SMAD2 and SMAD3 each of which controls specific sets of target genes [120], or (3) encoding of specific gene expression programs by the temporal dynamics of the SMAD pathway [121, 122]. Specifically, it has been suggested that a transient SMAD signal may be sufficient for EMT and cell migration, while sustained signaling additionally triggers cell cycle arrest [121]. Thus, quantitative insights into the

Fig. 3 (continued) SMAD7 which acts as a negative feedback inhibiting the activity of RII receptors. **(b)** Temporal dynamics of SMAD nuclear translocation at the single-cell level. Population average (black) and standard deviation (gray shades) of the single-cell Nuc/Cyt SMAD2 ratio after stimulation with 100 pM of TGF β . Exemplary trajectories of transient, sustained, and nonresponding cells are shown in different colors (see legend). Another cell (purple) shows a bursting event (red). (Data from Strasen et al. [33]). **(c)** Gradual behavior of SMAD signaling at the single-cell level. Snapshot histograms of Nuc/Cyt SMAD2 ratio 70 min (time of peak) after stimulation with varying doses of TGF β (see legend). All distributions are unimodal and shift toward higher mean values with increasing stimulation. (Data from Strasen et al. [33]). **(d)** Three-tiered modeling approach for modeling TGF β /SMAD signaling heterogeneity: In step (1), a population-average model is fitted to experimental data. In step (2), the model is refined to a description of subpopulations which are identified from the measured single-cell trajectories using a clustering approach. For each subpopulation, the model is fitted to the median time courses of a cluster, assuming subpopulation-specific signaling protein expression. Step (3) yields simulations of individual cells, since signaling protein levels are sampled from log-normal distributions in each subpopulation model. In step (4) single-cell trajectories from step (3) are combined according to yield a description of the complete cell population. (Figure modified from Strasen et al. [33])

pathway dynamics by time-resolved live-cell imaging [18] and mathematical modeling are important to better understand cellular responses to TGF β stimulation.

2.2.2 Lessons Learned from Single-Cell Experiments of TGF β Signaling

At the single-cell level, individual cells respond very differently to TGF β treatment. Due to this heterogeneity, time-resolved analyses of SMAD signaling at the single-cell level are valuable tools to understand the link between signaling dynamics, gene expression, and cellular outcome. In fact, single-cell studies further supported that the amplitude and/or duration of the SMAD signal partially determines whether a cell will react at all to stimulation and/or whether it will respond with migration or cell cycle arrest [33, 36].

Established experimental readouts of TGF β /SMAD signaling at the single-cell level include measurements of receptor levels and their internalization [123], nuclear translocation of SMAD proteins [33, 34, 36, 41, 124], SMAD trimerization [125], and SMAD-induced gene expression [34, 73]. At the signaling level, SMAD nuclear translocation assays are most widely used. They rely on the mild overexpression of SMAD2 or SMAD4 fluorescent fusion proteins, and the nuclear translocation of the fluorophore is then used as a proxy for pathway activation [33, 34, 36, 41, 124, 126]. By automated microscopy, images are taken on a temporal resolution of a few minutes. Subsequently image analysis is performed to track cells and to segment them into nuclear and cytoplasmic compartments. The amount of SMAD-associated fluorophore in nucleus or the nucleus-to-cytoplasmic fluorescence ratio is then used as a measure of signaling pathway activity. Depending on the study, this technique allowed for signaling analysis in 250–1500 single cells per condition over a time frame of 45 min to 24 h.

These large-scale single-cell datasets indicated that heterogeneous SMAD signaling at the single-cell level exhibits a few key features that are recurrently observed across experimental groups and cellular systems:

1. *SMAD signaling is gradual at the single-cell level* (Fig. 3c): In Subheading 1, it was discussed that certain signaling pathways like the MAPK and apoptosis cascades show bimodal behavior, i.e., complete or no activation in individual cells. Available single-cell studies on TGF β signaling suggest that this pathway rather acts like a gradual continuum, i.e., snapshot histograms of SMAD2 nuclear translocation show a unimodal distribution with strong cell-to-cell variability (Fig. 3c). With increasing TGF β doses, this mean value of this continuous distribution gradually shifts to higher signaling levels [33, 34, 36, 41, 126].
2. *Single cells show qualitative differences in signal shape* (Fig. 3b): Single-cell analyses show strong differences in the absolute level of SMAD2 or SMAD4 nuclear translocation between

cells at a given time point [33, 34, 36, 41, 126]. In time-resolved analyses, individual cells may also be distinct in the shape of the signal, e.g., in the kinetics or the degree of adaptation to a lower plateau after the initial peak amplitude. Such heterogeneity in the shape of the signal was observed either at the level of SMAD2 or SMAD4 nuclear translocation [33, 34, 36] or at the level of target gene expression [34]. In contrast, in other cellular systems, the shape of SMAD2 nuclear translocation was fairly similar between cells [41]. Clustering techniques using dynamic time warping as a similarity measure between cells were used to sort the trajectories of individual cells into classes with qualitatively different dynamics [33, 34]. Using this clustering approach, we found that even for a given TGF β dose, some cells do not respond to the stimulus (non-responders), others show a transient response, whereas the remainder show sustained pathway activation (Fig. 3b). At very low TGF β stimulation, most cells belong to the nonresponding cluster, whereas at intermediate and high TGF β doses, the transient and sustained clusters predominate, respectively. Population-average measurements are a mixture of these qualitatively distinct responses and therefore only partially cover the complexity of the pathway at the single-cell level. Interestingly, the signalling clusters are better predictors for TGF β -induced cell migration and division when compared to the applied extracellular ligand dose [33]. This further suggests that the cellular decisions are linked to SMAD signaling dynamics at the single-cell level.

3. *Single cells show burst-like shuttling of SMAD proteins* (Fig. 3b): Upon stimulation, the population-average response of the SMAD signaling pathway typically shows an initial peak amplitude ~60 mins after stimulation. Afterward, the population-average signal slowly declines over a timescale of several hours but may remain constantly elevated, e.g., upon strong TGF β stimulation, but this depends on the cellular context. For such sustained behavior, the single-cell response is distinct from the population average and shows repeated bursts of nuclear translocation (Fig. 3b). Specifically, the nuclear SMAD2 or SMAD4 levels decline strongly after the initial peak, before again reaching once or multiple times levels comparable to the level of the initial peak [33, 34, 99]. This did not appear to a technical artifact of imaging, as a generic nuclear marker (H2B) did not show bursting behavior [33]. Furthermore, SMAD4 bursts were reported in developing Xenopus embryos in which TGF β family ligands play an important role and the behavior could be reproduced in isolated animal cap explants [126]. Interestingly, these pulsatile SMAD translocation

dynamics are irregular in their timing intervals and amplitudes, suggesting that they may, in part, arise from stochastic dynamics of the SMAD signaling pathway.

In the following, we will discuss how mathematical models can provide insights into these dynamical features at the single-cell level. We will first review population-average models of TGF β /SMAD signaling and will then turn to modeling approaches at the single-cell level.

2.2.3 Population-Average Models of TGF β /SMAD Signaling

Early kinetic models of TGF β /SMAD signaling mainly focused on the description of Western blot measurements of SMAD2/3 phosphorylation and complex formation [127–135]. Since these experimental methods provide average quantifications of thousands to millions of cells, the resulting models describe the behavior of a representative average cell and fail to capture heterogeneity in the population. Population-average models of SMAD signaling are typically based on deterministic ordinary differential equations (ODEs), and the individual reaction steps are formulated based on mass-action kinetics. Models proposed in literature have been reviewed elsewhere [136, 137] and differ in the level of detail they consider and in the reaction mechanisms they focus on. Still, most of the models share a set of key mechanisms including receptor-ligand binding, receptor shuttling to the endosome, receptor-mediated SMAD phosphorylation SMAD (de)phosphorylation, trimerization and nuclear translocation, as well as transcriptional negative feedback via target genes that, for instance, inhibit receptor signaling (Fig. 3a). The kinetic parameters are typically not known and were estimated by fitting the models to experimental data [33, 131, 138], or the parameter space was explored by random sampling or sensitivity analysis [128, 130, 139, 140].

Interestingly, population-average models alongside with quantitative experiments reproduced and provided insights into several features of heterogeneous TGF β /SMAD signaling at the single-cell level including the gradual behavior of the pathway, transient vs. sustained signaling and pulsatile pathway dynamics. Thus, they provide hints to mechanisms of heterogeneity and serve as a basis for deterministic modeling of variability at the single-cell level.

1. *Gradual dose-response behavior:* Population-average modeling studies and dose-response measurements revealed gradually increasing SMAD signaling in response to increasing doses of TGF β . Specifically, intracellular SMAD signaling exhibits a shallow dose-response curve with a Hill coefficient (n_H) of close to or less than 1 [33, 73, 132, 141]. Modeling studies further revealed that switch-like dose-response behavior ($n_H = 4.5$) late after stimulation (24 h) is *not* an inherent

feature of the SMAD signaling pathway but arises from degradation of extracellular TGF β in the cell culture dish [132, 142]. Hence, the SMAD signaling pathway models respond gradually to perturbations (in both TGF β concentration and intracellular protein concentrations) and are therefore consistent with a unimodal SMAD nuclear translocation distribution at the single-cell level.

2. *Features controlling signal amplitude and duration:* Sensitivity analysis of population-average models revealed the relative importance of individual reaction steps in controlling the signal amplitude and duration [129–132]. It seems that the signal duration (i.e., the shape of the signal) is mainly set by the kinetics of receptor-ligand binding and receptor shuttling. Accordingly, SMAD nuclear translocation cycle typically shows similar dynamics as the receptor level and mainly acts as a remote sensor that directly reflects receptor changes, though with a slight time delay of ~10 min [129, 143]. Molecular mechanisms that control the signaling dynamics at the receptor (and thus the SMAD) level include (i) receptor down-regulation from the cell surface by internalization of receptor-ligand complexes into endosomal compartments [133, 144]; (ii) cell-mediated degradation of extracellular TGF β , again by internalization of receptor-ligand complexes and subsequent intracellular degradation of the ligand [127, 132]. This mechanism of signal termination becomes an important factor in controlling the length of the signal if the number of extracellular TGF β molecules per cell is limiting (low TGF β concentration and/or small extracellular medium volume); (iii) negative feedback of SMAD target genes to the receptor level, e.g., by SMAD-induced expression of inhibitory SMAD7 and BAMBI proteins. These proteins bind to TGF β receptor complexes, thereby inhibiting their kinase activity and targeting them for degradation [33, 130, 135]. Taken together, the signal duration is controlled by multiple mechanisms at the receptor level. Using sensitivity analysis of population-average models, a similar multilevel regulation by many reaction steps in the pathway was shown for the absolute scale (i.e., the amplitude) of the SMAD signal [129, 131]. At the single-cell level, such mechanisms jointly control heterogeneous signaling dynamics, and this can be investigated by parameter sampling in a deterministic model (*see below*).
3. *Pulsatile SMAD shuttling:* Population-average modeling and quantitative experimental analyses suggested that SMAD signaling induced by TGF β or BMP could show (damped) oscillatory behavior, in which a single stimulus induces two or more repeated pulses of SMAD nuclear translocation [130, 144]. Using global parameter sampling, Wegner et al.

proved that oscillations require the presence of transcriptional negative feedback—if this feedback is switched off, no physiologically plausible parameter configuration can produce oscillations [130]. Accordingly, knockdown of transcriptional feedback regulators SMAD6 and SMAD7 abolished BMP-induced SMAD oscillations [144]. It is possible that such oscillatory negative feedback contributes to repeated bursting of SMAD2 or SMAD4 nuclear translocation observed in single cells, though additional stochastic mechanisms need to be taken into account to describe the irregularity of bursts [33, 99, 126].

2.2.4 Towards Quantitative Modeling of SMAD Signaling Heterogeneity

On the basis of the established population-average models, we recently derived a deterministic modeling framework to quantitatively describe cell-to-cell variability in the TGF β /SMAD signaling pathway [33]. Our study was based on imaging data in which the nuclear translocation of SMAD2/4-GFP fusion proteins was monitored in thousands of living MCF10A cells over 24 h.

As a basis for modeling, we initially analyzed characteristic features of SMAD signaling heterogeneity. We performed sister cell experiments and found that sister cells are more similar than random pairs of cells but desynchronize after several hours. This indicated that signaling fluctuations are nongenetic but temporally stable. We then analyzed potential sources of heterogeneity in SMAD signaling and considered that SMAD signaling may be influenced by cell cycle stage and/or cell density [125, 145]. Using live-cell imaging, we followed cell division events and quantified the cell density before and during TGF β stimulation and found that these two factors had negligible impact on heterogeneous signaling in our culture conditions. Taken together, this indicated that SMAD signaling heterogeneity can be modeled using a deterministic approach based on ODEs (Subheading 2.1), with the assumption of stochastic (but temporally stable) fluctuations in signaling protein expression levels.

We started with a detailed kinetic pathway model that describes known mechanisms of SMAD signaling and comprises a total of 23 molecular species and 45 kinetic reaction parameters. Like most other models of TGF β signaling, our model contained the main features described in Subheading 2.2.3, including an endosomal receptor shuttling module, a SMAD translocation module, and a transcriptional feedback module (Fig. 3a). With this model, we sought to describe a large experimental dataset, in which several levels of signaling (TGF β receptor expression, nuclear translocation of SMAD2 and SMAD4, as well as SMAD7 mRNA expression) were measured for multiple experimental conditions at the single-cell and population-average levels. In total, this dataset comprised >1,000,000 data points, mainly from densely sampled live-cell

imaging experiments at multiple experimental conditions. Owing to the high complexity of model and data, we did not aim for a quantitative fitting of the model to the single-cell data but instead devised a three-tiered, modeling strategy to derive a quantitative description of heterogeneous signaling (Fig. 3d). Initially, we describe the population-average dynamics. Then, we refine the description of the pathway to the level cellular subpopulations showing qualitatively distinct signaling dynamics. Finally, we develop an ensemble of single-cell models to describe the complete heterogeneous cell population. Specifically, the three modeling steps were as follows:

1. *Population-average modeling*: To derive a quantitative description of the SMAD signaling dynamics, we initially fitted the model to population-average data (Fig. 3d, left). For fitting, we used the population-median nuclear translocation time courses of fluorescently labeled SMAD2 and SMAD4 for varying TGF β doses and for restimulation experiments, in which cells were repeatedly challenged with the ligand. Furthermore, the fitting took into account pathway measurements that were only possible at the population-average level (TGF β receptor protein expression, SMAD7 mRNA expression). After calibration, we validated the predictive power of our model for previously untested molecular species (time course of extracellular TGF β degradation) and experimental conditions (restimulation experiments, inhibition of transcriptional feedback loops by small-molecule inhibitor DRB).
2. *Description of cellular subpopulations*: Having a predictive population-average model at hand, we sought to quantitatively describe variability in signaling while limiting the computational cost. Therefore, we refrained from fitting our model to the complete single-cell population (Subheading 2.1.4) but only fitted six subpopulations which show qualitatively distinct dynamics of signaling (e.g., transient vs. sustained; see Fig. 3d, middle). These subpopulations were identified by k-means clustering of single-cell SMAD2 nuclear translocation time courses according to their similarity in shape and amplitude. We separately fitted the subpopulation-median time course of each cluster and only allowed variation in the expression of signaling proteins (e.g., TGF β receptors, SMADs) within the range of typical cell-to-cell variation (+/– twofold). In contrast, the kinetic parameters were fixed to their population-average value, i.e., their variability was neglected. With these assumptions, we could quantitatively describe all subpopulations and had therefore developed our model from a population-average description to a description of six representative cells with characteristic dynamical features.

3. Ensemble modeling of complete cell population: To directly compare our simulations to single-cell experiments, we converted the subpopulation models to an ensemble of artificial cells representing the heterogeneity of the entire cell population (Fig. 3d, right). Artificial single cells belonging to each subpopulation were generated by repeated simulation with signaling protein concentrations varying around the best-fit values of the corresponding subpopulation model. The full cell population was assembled in silico by combining artificial cells according to the experimentally observed proportion of corresponding subpopulations. The degree of variation was assumed to be the same for all sampled signaling proteins. The common protein coefficient of variation (std/mean) was chosen by matching the simulated and experimentally observed snapshot distributions at particular time points using summary statistics.

Taken together, we obtained an in silico cell population with realistic properties close to the experimental data. Importantly, we could show that our three-tiered modeling approach, in which we considered the subpopulation structure (**step 2**), yielded a better agreement with single-cell snapshot distributions (**step 3**) when compared to direct sampling of protein concentrations in the population-average model. The model reproduced key features of single-cell TGF β signaling including gradual (unimodal) behavior and strong heterogeneity in the time course shape (Subheading 2.2.2). By calculating Euclidean distances, we quantitatively compared the simulated single-cell trajectories to the six experimentally observed time course clusters. Since the model took into account subpopulation information, we obtained a very similar dose-dependent decomposition into nonresponding, transient, and sustained signaling classes as for the experimental data. Thus, the model correctly takes into account temporal correlations in signaling pathway activity and can be used to predict drifts in the shape and proportion of the original subpopulations for any experimental condition. In fact, we confirmed such predictions to a knockout of the negative feedback regulator SMAD7. As predicted by the model, we found that the effect of SMAD7 on the signaling dynamics was restricted to certain cellular subpopulations and was observed for specific doses of TGF β only. Hence, the model allowed us to quantitatively understand the cell-specific impact of experimental perturbations and allowed mechanistic insights into cellular heterogeneity.

One limitation of the current model is that the best-fit parameter values in population-average and subpopulation fitting are not unique (non-identifiability problem). Nevertheless, robust predictions could be made, since very similar simulation results were obtained when comparing multiple fits obtained during a multi-

start optimization (repeated model fitting from different starting parameters). The subpopulation fitting (**step 2**) currently corresponds to a the standard two-stage approach discussed in Subheading 2.1.3, since the protein concentrations in each subpopulation were estimated separately without additional constraints about the protein distribution in the cell population. After assembly of the complete cell population (**step 3**), we confirmed that signaling protein levels in the model show a realistic log-normal distribution. However, such a distribution is not automatically granted in the current approach. Therefore, it would be beneficial to either improve the identifiability of parameters by model reduction or to take into account additional constraints during subpopulation fitting.

Taken together, our study suggests that heterogeneity of TGF β /SMAD signaling at the single-cell level can be quantitatively described using a deterministic modeling approach. Key features of the pathway at the single-cell level were reproduced, including gradual behavior and cell-specific characteristics in signaling shape. Since the model is deterministic in nature, it currently does not describe the apparently stochastic, burst-like shuttling of SMAD proteins into the nucleus (Subheading 2.2.2). To describe this phenomenon, stochastic effects, most likely in endosomal receptor shuttling, need to be taken into account, and quantitative fitting approaches can be used to match burst features in the stochastic model to the experimental data [99]. Such stochastic bursting deserves further investigation, as other signaling pathways such as the MAPK cascade similarly show repeated pulses of activation which have profound impact on cell fate [43, 146].

2.2.5 Future Directions: Link Between Signaling and Gene Expression Heterogeneity

Another aspect that deserves further attention in the future is how fluctuations in signaling proteins translate into fluctuations in downstream target gene expression. SMAD transcription factors mediate cellular responses by binding to target gene promoters, thereby inducing large-scale gene expression programs involved in cell migration, EMT, and cell cycle arrest [115, 147]. Given this link between SMAD binding and gene expression, it seems that cell-specific gene expression and morphological changes may be predictable based on the amplitude and/or dynamics of SMAD signaling [147]. In fact, by clustering single-cell SMAD time courses, we found that cell migration and cell division kinetics can—in part—be explained based on the dynamics of SMAD nuclear translocation [33].

Based on these observations, it is natural to extend current mathematical models to SMAD-induced gene expression and—in the long run—to TGF β -induced cell fate decisions. At the population-average level, SMAD signaling dynamics seem indeed to be related to gene expression, as models simultaneously

describing time courses at both levels are well-established: Most of these studies modeled the dynamics of certain negative feedback regulators or pathway targets using SMAD-dependent transcription and linear degradation of the target gene. By simultaneously fitting such synthesis and decay models using SMAD kinetics as an input, target gene mRNA dynamics can be well-explained in multiple cases [33, 73, 130, 139, 142, 148, 149]. A recent combined modeling and experimental study extended this idea to a larger set of target genes based quantitative and time-resolved measurements SMAD trimeric complexes [138]. Using a model fitting framework, the authors inferred the specificity of target gene induction by distinct SMAD complexes and successfully predicted gene expression outcomes for novel experimental conditions.

Despite such good accordance of SMAD signaling and gene expression responses in cell populations, it remains challenging to directly link both levels in single cells. Experimentally, such analyses require SMAD signaling dynamics and gene expression to be simultaneously measured in the same cell by live-cell imaging of fluorescent SMAD reporters and subsequent smFISH of mRNA expression in fixed cells [34, 41]. Those two studies conducted so far agree that on a single-cell level, neither SMAD2 nor SMAD4 absolute levels in the nucleus accurately predict the cell-specific expression of target mRNAs. However, Frick et al. reported that the TGF β -induced fold change of the nuclear SMAD levels relative to basal predicts stimulus-induced target gene expression responses. For the genes snail and CTGF, they found Spearman rank correlation coefficients of ~0.5 between those fold changes and the mRNA abundance as measured by smFISH. These findings could not be confirmed by [34], who analyzed the expression of CTGF with high temporal resolution using live luminescence imaging. They found no correlation between the fold changes in nuclear SMAD and CTGF expression. Thus, even though the proposed fold-change detection in SMAD target gene expression may be an elegant way to reliably respond to stimulation despite high variability in absolute nuclear SMAD levels, it remains to be confirmed whether this is general phenomenon applicable to many genes and cellular systems.

Therefore, the mechanistic link between SMAD signaling fluctuations, gene expression, and heterogeneous cellular responses remains to be established. In any case, a deterministic 1:1 correspondence of signaling and gene expression appears unlikely, since gene expression modeling requires stochastic modeling of promoter switching, as [150] showed for SMAD target genes (*see also Subheading 2.1.3*). Even though each individual gene might respond stochastically and with little correlation to the SMAD signal, it still remains possible that SMAD signaling fluctuations directly affect cellular outcomes through their cumulative effect on

many target genes controlling a common biological process. Genome-wide single-cell RNA sequencing approaches will shed light on such coordinated gene expression programs at the level of individual cells.

Recent work has presented methods for the targeted manipulation of SMAD signaling dynamics at the single-cell level [36, 124], and similar tools were developed for other signaling pathways [39, 43, 83]. The combination of such highly controllable tools with gene expression measurements will provide direct insights into the impact of SMAD signaling dynamics on gene expression outcomes and will advance our understanding of heterogeneous decision-making by the TGF β pathway.

References

1. Altschuler SJ, Wu LF (2010) Cellular heterogeneity: do differences make a difference? *Cell* 141:559–563
2. Komin N, Skupin A (2017) How to address cellular heterogeneity by distribution biology. *Curr Opin Syst Biol* 3:154–160
3. Ackermann M (2015) A functional perspective on phenotypic heterogeneity in microorganisms. *Nat Rev Microbiol* 13:497–508
4. Chang HH, Hemberg M, Barahona M et al (2008) Transcriptome-wide noise controls lineage choice in mammalian progenitor cells. *Nature* 453:544–547. <https://doi.org/10.1038/nature06965>
5. Sharma SV, Lee DY, Li B et al (2010) A chromatin-mediated reversible drug-tolerant state in cancer cell subpopulations. *Cell* 141: 69–80. <https://doi.org/10.1016/j.cell.2010.02.027>
6. Paek AL, Liu JC, Loewer A et al (2016) Cell-to-cell variation in p53 dynamics leads to fractional killing. *Cell* 165:631–642. <https://doi.org/10.1016/j.cell.2016.03.025>
7. Weinberger LS, Burnett JC, Toettcher JE et al (2005) Stochastic gene expression in a lentiviral positive-feedback loop: HIV-1 tat fluctuations drive phenotypic diversity. *Cell* 122: 169–182. <https://doi.org/10.1016/j.cell.2005.06.006>
8. Suel GM, Kulkarni RP, Dworkin J et al (2007) Tunability and noise dependence in differentiation dynamics. *Science* (80-) 315:1716–1719. <https://doi.org/10.1126/science.1137455>
9. Levy SF, Ziv N, Siegal ML (2012) Bet hedging in yeast by heterogeneous, age-correlated expression of a stress protectant. *PLoS Biol* 10:e1001325. <https://doi.org/10.1371/journal.pbio.1001325>
10. Wernet MF, Mazzoni EO, Çelik A et al (2006) Stochastic spineless expression creates the retinal mosaic for colour vision. *Nature* 440:174–180. <https://doi.org/10.1038/nature04615>
11. Ohnishi Y, Huber W, Tsumura A et al (2014) Cell-to-cell expression variability followed by signal reinforcement progressively segregates early mouse lineages. *Nat Cell Biol* 16:27–37. <https://doi.org/10.1038/ncb2881>
12. Raj A, Rifkin SA, Andersen E, Van Oudenaarden A (2010) Variability in gene expression underlies incomplete penetrance. *Nature* 463: 913–918. <https://doi.org/10.1038/nature08781>
13. Flusberg DA, Roux J, Spencer SL, Sorger PK (2013) Cells surviving fractional killing by TRAIL exhibit transient but sustainable resistance and inflammatory phenotypes. *Mol Biol Cell* 24:2186–2200. <https://doi.org/10.1091/mbc.E12-10-0737>
14. Lincoln FA, Imig D, Boccellato C et al (2018) Sensitization of glioblastoma cells to TRAIL-induced apoptosis by IAP- and Bcl-2 antagonism. *Cell Death Dis* 9:1–14. <https://doi.org/10.1038/s41419-018-1160-2>
15. Crawford N, Salvucci M, Hellwig CT et al (2018) Simulating and predicting cellular and in vivo responses of colon cancer to combined treatment with chemotherapy and IAP antagonist Birinapant/TL32711. *Cell Death Differ* 25:1952–1966. <https://doi.org/10.1038/s41418-018-0082-y>
16. Roux J, Hafner M, Bandara S et al (2015) Fractional killing arises from cell-to-cell

- variability in overcoming a caspase activity threshold. *Mol Syst Biol* 11:803. <https://doi.org/10.1525/msb.20145584>
17. Brock A, Chang H, Huang S (2009) Non-genetic heterogeneity – a mutation-independent driving force for the somatic evolution of tumours. *Nat Rev Genet* 10: 336–342. <https://doi.org/10.1038/nrg2556>
 18. Spiller DG, Wood CD, Rand DA, White MRH (2010) Measurement of single-cell dynamics. *Nature* 465:736–745. <https://doi.org/10.1038/nature09232>
 19. Loewer A, Lahav G (2011) We are all individuals: causes and consequences of non-genetic heterogeneity in mammalian cells. *Curr Opin Genet Dev* 21:753–758
 20. Gaudet S, Miller-Jensen K (2016) Redefining signaling pathways with an expanding single-cell toolbox. *Trends Biotechnol* 34:458–469
 21. Albeck JG, Pargett M, Davies AE (2018) Experimental and engineering approaches to intracellular communication. *Essays Biochem* 62:515–524
 22. Gough A, Stern AM, Maier J et al (2017) Biologically relevant heterogeneity: metrics and practical insights. *SLAS Discov* 22:213–237
 23. Jeknić S, Kudo T, Covert MW (2019) Techniques for studying decoding of single cell dynamics. *Front Immunol* 10:755
 24. Ferrell JE, Machleder EM (1998) The biochemical basis of an all-or-none cell fate switch in *xenopus* oocytes. *Science* (80-) 280:895–898. <https://doi.org/10.1126/science.280.5365.895>
 25. Feinerman O, Veiga J, Dorfman JR et al (2008) Variability and robustness in T cell activation from regulated heterogeneity in protein levels. *Science* (80-) 321:1081–1084. <https://doi.org/10.1126/science.1158013>
 26. Santos SDM, Verveer PJ, Bastiaens PIH (2007) Growth factor-induced MAPK network topology shapes Erk response determining PC-12 cell fate. *Nat Cell Biol* 9:324–330. <https://doi.org/10.1038/ncb1543>
 27. Miura H, Kondo Y, Matsuda M, Aoki K (2018) Cell-to-cell heterogeneity in p38-mediated cross-inhibition of JNK causes stochastic cell death. *Cell Rep* 24:2658–2668. <https://doi.org/10.1016/j.celrep.2018.08.020>
 28. Mackeigan JP, Murphy LO, Dimitri CA, Blenis J (2005) Graded mitogen-activated protein kinase activity precedes switch-like c-Fos induction in mammalian cells. *Mol Cell Biol* 25:4676–4682. <https://doi.org/10.1128/MCB.25.11.4676-4682.2005>
 29. Cohen-Saidon C, Cohen AA, Sigal A et al (2009) Dynamics and variability of ERK2 response to EGF in individual living cells. *Mol Cell* 36:885–893. <https://doi.org/10.1016/j.molcel.2009.11.025>
 30. Rehm M, Dübbmann H, Jänicke RU et al (2002) Single-cell fluorescence resonance energy transfer analysis demonstrates that caspase activation during apoptosis is a rapid process: role of caspase-3. *J Biol Chem* 277: 24506–24514. <https://doi.org/10.1074/jbc.M110789200>
 31. Gross SM, Rotwein P (2015) Akt signaling dynamics in individual cells. *J Cell Sci* 128: 2509–2519. <https://doi.org/10.1242/jcs.168773>
 32. Gross SM, Dane MA, Bucher E, Heiser LM (2019) Individual cells can resolve variations in stimulus intensity along the IGF-PI3K-AKT signaling axis. *Cell Syst* 9:580–588.e4. <https://doi.org/10.1016/j.cels.2019.11.005>
 33. Strasen J, Sarma U, Jentsch M et al (2018) Cell-specific responses to the cytokine TGF β are determined by variability in protein levels. *Mol Syst Biol* 14:e7733. <https://doi.org/10.1525/msb.20177733>
 34. Tidin O, Friman ET, Naef F, Suter DM (2019) Quantitative relationships between SMAD dynamics and target gene activation kinetics in single live cells. *Sci Rep* 9:1–11. <https://doi.org/10.1038/s41598-019-41870-2>
 35. Selimkhanov J, Taylor B, Yao J et al (2014) Accurate information transmission through dynamic biochemical signaling networks. *Science* (80-) 346:1370–1373. <https://doi.org/10.1126/science.1254933>
 36. Sorre B, Warmflash A, Brivanlou AH, Siggia ED (2014) Encoding of temporal signals by the TGF- β pathway and implications for embryonic patterning. *Dev Cell* 30:334–342. <https://doi.org/10.1016/j.devcel.2014.05.022>
 37. Purvis JE, Lahav G (2013) Encoding and decoding cellular information through signaling dynamics. *Cell* 152:945–956
 38. Zhang Q, Gupta S, Schipper DL et al (2017) NF- κ B dynamics discriminate between TNF doses in single cells. *Cell Syst* 5:638–645.e5. <https://doi.org/10.1016/j.cels.2017.10.011>
 39. Ashall L, Horton CA, Nelson DE et al (2009) Pulsatile stimulation determines timing and specificity of NF- κ B-dependent transcription.

- Science (80-) 324:242–246. <https://doi.org/10.1126/science.1164860>
40. Goentoro L, Shoval O, Kirschner MW, Alon U (2009) The incoherent feedforward loop can provide fold-change detection in gene regulation. *Mol Cell* 36:894–899. <https://doi.org/10.1016/j.molcel.2009.11.018>
 41. Frick CL, Yarka C, Nunn H, Goentoro L (2017) Sensing relative signal in the Tgf- β / Smad pathway. *Proc Natl Acad Sci U S A* 114: E2975–E2982. <https://doi.org/10.1073/pnas.1611428114>
 42. Lee RECC, Walker SR, Savery K et al (2014) Fold change of nuclear NF- κ B determines TNF-induced transcription in single cells. *Mol Cell* 53:867–879. <https://doi.org/10.1016/j.molcel.2014.01.026>
 43. Li P, Elowitz MB (2019) Communication codes in developmental signaling pathways. *Development* 146:dev170977. <https://doi.org/10.1242/dev.170977>
 44. Pisco AO, Huang S (2015) Non-genetic cancer cell plasticity and therapy-induced stemness in tumour relapse: What does not kill me strengthens me. *Br J Cancer* 112:1725–1732
 45. Inde Z, Dixon SJ (2018) The impact of non-genetic heterogeneity on cancer cell death. *Crit Rev Biochem Mol Biol* 53:99–114. <https://doi.org/10.1080/10409238.2017.1412395>
 46. Spencer SL, Gaudet S, Albeck JG et al (2009) Non-genetic origins of cell-to-cell variability in TRAIL-induced apoptosis. *Nature* 459: 428–432. <https://doi.org/10.1038/nature08012>
 47. Geva-Zatorsky N, Rosenfeld N, Itzkovitz S et al (2006) Oscillations and variability in the p53 system. *Mol Syst Biol* 2:2006.0033. <https://doi.org/10.1038/msb4100068>
 48. Deathridge J, Antolović V, Parsons M, Chubb JR (2019) Live imaging of erk signalling dynamics in differentiating mouse embryonic stem cells. *Development* 146:dev172940. <https://doi.org/10.1242/dev.172940>
 49. Heinrich S, Geissen EM, Kamenz J et al (2013) Determinants of robustness in spindle assembly checkpoint signalling. *Nat Cell Biol* 15:1328–1339. <https://doi.org/10.1038/ncb2864>
 50. Sandler O, Mizrahi SP, Weiss N et al (2015) Lineage correlations of single cell division time as a probe of cell-cycle dynamics. *Nature* 519:468–471. <https://doi.org/10.1038/nature14318>
 51. Rand U, Rinas M, Werk JS et al (2012) Multi-layered stochasticity and paracrine signal propagation shape the type-I interferon response. *Mol Syst Biol* 8:584. <https://doi.org/10.1038/msb.2012.17>
 52. Legewie S, Herzel H, Westerhoff HV, Blüthgen N (2008) Recurrent design patterns in the feedback regulation of the mammalian signalling network. *Mol Syst Biol* 4:190. <https://doi.org/10.1038/msb.2008.29>
 53. Snijder B, Pelkmans L (2011) Origins of regulated cell-to-cell variability. *Nat Rev Mol Cell Biol* 12:119–125. <https://doi.org/10.1038/nrm3044>
 54. Battich N, Stoeger T, Pelkmans L (2015) Control of transcript variability in single mammalian cells. *Cell* 163:1596–1610. <https://doi.org/10.1016/j.cell.2015.11.018>
 55. Snijder B, Sacher R, Rämö P et al (2009) Population context determines cell-to-cell variability in endocytosis and virus infection. *Nature* 461:520–523. <https://doi.org/10.1038/nature08282>
 56. Yuan TL, Wulf G, Burga L, Cantley LC (2011) Cell-to-cell variability in PI3K protein level regulates PI3K-AKT pathway activity in cell populations. *Curr Biol* 21:173–183. <https://doi.org/10.1016/j.cub.2010.12.047>
 57. Adlung L, Stapor P, Tönsing C, et al (2019) Cell-to-cell variability in JAK2/STAT5 pathway components and cytoplasmic volumes define survival threshold in erythroid progenitor cells. *bioRxiv* 866871. <https://doi.org/10.1101/866871>
 58. Sigal A, Milo R, Cohen A et al (2006) Variability and memory of protein levels in human cells. *Nature* 444:643–646. <https://doi.org/10.1038/nature05316>
 59. Raj A, van Oudenaarden A (2008) Nature, nurture, or chance: stochastic gene expression and its consequences. *Cell* 135:216–226
 60. Fritzsch C, Baumgärtner S, Kuban M et al (2018) Estrogen-dependent control and cell-to-cell variability of transcriptional bursting. *Mol Syst Biol* 14:e7678. <https://doi.org/10.1525/msb.20177678>
 61. Chubb JR, Trcek T, Shenoy SM, Singer RH (2006) Transcriptional pulsing of a developmental gene. *Curr Biol* 16:1018–1025. <https://doi.org/10.1016/j.cub.2006.03.092>
 62. Kovary KM, Taylor B, Zhao ML, Teruel MN (2018) Expression variation and covariation impair analog and enable binary signaling control. *Mol Syst Biol* 14:e7997. <https://doi.org/10.1525/msb.20177997>

63. Klipp E, Liebermeister W, Wierling C, Kowald A (2016) Systems biology: a textbook. Wiley-VCH
64. Wilkinson DJ (2011) Stochastic modelling for systems biology, 2nd edn. CRC Press
65. Borisov N, Aksamitiene E, Kiyatkin A et al (2009) Systems-level interactions between insulin-EGF networks amplify mitogenic signaling. *Mol Syst Biol* 5:256. <https://doi.org/10.138/msb.2009.19>
66. Kamenz J, Mihaljev T, Kubis A et al (2015) Robust ordering of anaphase events by adaptive thresholds and competing degradation pathways. *Mol Cell* 60:446–459. <https://doi.org/10.1016/j.molcel.2015.09.022>
67. Calzone L, Tournier L, Fourquet S et al (2010) Mathematical modelling of cell-fate decision in response to death receptor engagement. *PLoS Comput Biol* 6: e1000702. <https://doi.org/10.1371/journal.pcbi.1000702>
68. Blüthgen N, Legewie S (2008) Systems analysis of MAPK signal transduction. *Essays Biochem* 45:95–108. <https://doi.org/10.1042/bse045095>
69. Legewie S, Blüthgen N, Herzel H (2006) Mathematical modeling identifies inhibitors of apoptosis as mediators of positive feedback and bistability. *PLoS Comput Biol* 2:1061–1073. <https://doi.org/10.1371/journal.pcbi.0020120>
70. Legewie S, Schoeberl B, Blüthgen N, Herzel H (2007) Competing docking interactions can bring about bistability in the MAPK cascade. *Biophys J* 93:2279–2288. <https://doi.org/10.1529/biophysj.107.109132>
71. Podtschaske M, Benary U, Zwinger S et al (2007) Digital NFATc2 activation per cell transforms graded T cell receptor activation into an all-or-none IL-2 expression. *PLoS One* 2:e935. <https://doi.org/10.1371/journal.pone.0000935>
72. Legewie S, Sers C, Herzel H (2009) Kinetic mechanisms for overexpression insensitivity and oncogene cooperation. *FEBS Lett* 583: 93–96. <https://doi.org/10.1016/j.febslet.2008.11.027>
73. Paulsen M, Legewie S, Eils R et al (2011) Negative feedback in the bone morphogenetic protein 4 (BMP4) synexpression group governs its dynamic signaling range and canalizes development. *Proc Natl Acad Sci U S A* 108:10202–10207. <https://doi.org/10.1073/pnas.1100179108>
74. Fritzsche-Guenther R, Witzel F, Sieber A et al (2011) Strong negative feedback from Erk to Raf confers robustness to MAPK signalling. *Mol Syst Biol* 7:489. <https://doi.org/10.138/msb.2011.27>
75. Blüthgen N, Legewie S (2013) Robustness of signal transduction pathways. *Cell Mol Life Sci* 70:2259–2269
76. Birtwistle MR, Rauch J, Kiyatkin A et al (2012) Emergence of bimodal cell population responses from the interplay between analog single-cell signaling and protein expression noise. *BMC Syst Biol* 6:109. <https://doi.org/10.1186/1752-0509-6-109>
77. Dobrzański M, Nguyen LK, Birtwistle MR et al (2014) Nonlinear signalling networks and cell-to-cell variability transform external signals into broadly distributed or bimodal responses. *J R Soc Interface* 11:20140383. <https://doi.org/10.1098/rsif.2014.0383>
78. Buchbinder JH, Pischel D, Sundmacher K et al (2018) Quantitative single cell analysis uncovers the life/death decision in CD95 network. *PLoS Comput Biol* 14:e1006368. <https://doi.org/10.1371/journal.pcbi.1006368>
79. Iwamoto K, Shindo Y, Takahashi K (2016) Modeling cellular noise underlying heterogeneous cell responses in the epidermal growth factor signaling pathway. *PLoS Comput Biol* 12:e1005222. <https://doi.org/10.1371/journal.pcbi.1005222>
80. Jeschke M, Baumgärtner S, Legewie S (2013) Determinants of cell-to-cell variability in protein kinase signaling. *PLoS Comput Biol* 9: e1003357. <https://doi.org/10.1371/journal.pcbi.1003357>
81. Gaudet S, Spencer SL, Chen WW, Sorger PK (2012) Exploring the contextual sensitivity of factors that determine cell-to-cell variability in receptor-mediated apoptosis. *PLoS Comput Biol* 8:e1002482. <https://doi.org/10.1371/journal.pcbi.1002482>
82. Bertaux F, Stoma S, Drasdo D, Batt G (2014) Modeling dynamics of cell-to-cell variability in TRAIL-induced apoptosis explains fractional killing and predicts reversible resistance. *PLoS Comput Biol* 10:e1003893. <https://doi.org/10.1371/journal.pcbi.1003893>
83. Ryu H, Chung M, Dobrzański M et al (2015) Frequency modulation of ERK activation dynamics rewires cell fate. *Mol Syst Biol* 11: 838. <https://doi.org/10.15252/msb.20156458>

84. Ahrends R, Ota A, Kovary KM et al (2014) Controlling low rates of cell differentiation through noise and ultrahigh feedback. *Science* (80-) 344:1384–1389. <https://doi.org/10.1126/science.1252079>
85. Kallenberger SM, Beaudouin J, Claus J et al (2014) Intra- and interdimeric caspase-8 self-cleavage controls strength and timing of CD95-induced apoptosis. *Sci Signal* 7:ra23. <https://doi.org/10.1126/scisignal.2004738>
86. Kallenberger SM, Unger AL, Legewie S et al (2017) Correlated receptor transport processes buffer single-cell heterogeneity. *PLoS Comput Biol* 13:e1005779. <https://doi.org/10.1371/journal.pcbi.1005779>
87. Meyer R, D'Alessandro LA, Kar S et al (2012) Heterogeneous kinetics of AKT signaling in individual cells are accounted for by variable protein concentration. *Front Physiol* 3:451. <https://doi.org/10.3389/fphys.2012.00451>
88. Filippi S, Barnes CP, Kirk PDWW et al (2016) Robustness of MEK-ERK dynamics and origins of cell-to-cell variability in MAPK signaling. *Cell Rep* 15:2524–2535. <https://doi.org/10.1016/j.celrep.2016.05.024>
89. Dixit PD, Lyashenko E, Niepel M, Vitkup D (2020) Maximum entropy framework for predictive inference of cell population heterogeneity and responses in signaling networks. *Cell Syst* 10:204–212.e8. <https://doi.org/10.1016/j.cels.2019.11.010>
90. Loos C, Moeller K, Fröhlich F et al (2018) A hierarchical, data-driven approach to modeling single-cell populations predicts latent causes of cell-to-cell variability. *Cell Syst* 6: 593–603.e13. <https://doi.org/10.1016/j.cels.2018.04.008>
91. Sachs K, Perez O, Pe'er D et al (2005) Causal protein-signaling networks derived from multiparameter single-cell data. *Science* (80-) 308:523–529. <https://doi.org/10.1126/science.1105809>
92. McAdams HH, Arkin A (1997) Stochastic mechanisms in gene expression. *Proc Natl Acad Sci U S A* 94:814–819. <https://doi.org/10.1073/pnas.94.3.814>
93. Gillespie DT (1977) Exact stochastic simulation of coupled chemical reactions. *J Phys Chem* 81:2340–2361
94. Zoller B, Nicolas D, Molina N, Naef F (2015) Structure of silent transcription intervals and noise characteristics of mammalian genes. *Mol Syst Biol* 11:823. <https://doi.org/10.1525/msb.20156257>
95. Li C, Cesbron F, Oehler M et al (2018) Frequency modulation of transcriptional bursting enables sensitive and rapid gene regulation. *Cell Syst* 6:409–423.e11. <https://doi.org/10.1016/j.cels.2018.01.012>
96. Lipniacki T, Puszynski K, Paszek P et al (2007) Single TNF α trimers mediating NF- κ B activation: stochastic robustness of NF- κ B signaling. *BMC Bioinformatics* 8: 376. <https://doi.org/10.1186/1471-2105-8-376>
97. Tay S, Hughey JJ, Lee TK et al (2010) Single-cell NF- κ B dynamics reveal digital activation and analogue information processing. *Nature* 466:267–271. <https://doi.org/10.1038/nature09145>
98. Matveeva A, Fichtner M, McAllister K et al (2019) Heterogeneous responses to low level death receptor activation are explained by random molecular assembly of the Caspase-8-activation platform. *PLoS Comput Biol* 15: e1007374. <https://doi.org/10.1371/journal.pcbi.1007374>
99. Kolbe N, Hexemer L, Bammert L-M, Loewer A, Lukáčová-Medviďová M, Legewie S (2022) Data-based stochastic modeling reveals sources of activity bursts in single-cell TGF- β signaling. *PLoS Comput Biol*. 18(6):e1010266. <https://doi.org/10.1371/journal.pcbi.1010266>
100. Lauffenburger DA, Linderman JJ (1993) Receptors: models for binding, trafficking, and signaling. Oxford University Press
101. Villaseñor R, Nonaka H, Del Conte-Zerial P et al (2015) Regulation of EGFR signal transduction by analogue-to-digital conversion in endosomes. *Elife* 4:e06156. <https://doi.org/10.7554/elife.06156>
102. Hasenauer J, Hasenauer C, Hucho T, Theis FJ (2014) ODE constrained mixture modeling: a method for unraveling subpopulation structures and dynamics. *PLoS Comput Biol* 10:e1003686. <https://doi.org/10.1371/journal.pcbi.1003686>
103. Llamosi A, Gonzalez-Vargas AM, Versari C et al (2016) What population reveals about individual cell identity: single-cell parameter estimation of models of gene expression in yeast. *PLoS Comput Biol* 12:e1004706. <https://doi.org/10.1371/journal.pcbi.1004706>
104. Zechner C, Ruess J, Krenn P et al (2012) Moment-based inference predicts bimodality in transient gene expression. *Proc Natl Acad Sci U S A* 109:8340–8345. <https://doi.org/10.1073/pnas.1200161109>
105. Zechner C, Unger M, Pelet S et al (2014) Scalable inference of heterogeneous reaction kinetics from pooled single-cell recordings.

- Nat Methods 11:197–202. <https://doi.org/10.1038/nmeth.2794>
106. Almquist J, Bendrioua L, Adiels CB et al (2015) A nonlinear mixed effects approach for modeling the cell-to-cell variability of Mig1 dynamics in yeast. PLoS One 10: e0124050. <https://doi.org/10.1371/journal.pone.0124050>
107. Karlsson M, Janzén DLII, Durrieu L et al (2015) Nonlinear mixed-effects modelling for single cell estimation: when, why, and how to use it. BMC Syst Biol 9:52. <https://doi.org/10.1186/s12918-015-0203-x>
108. Yao J, Pilko A, Wollman R (2016) Distinct cellular states determine calcium signaling response. Mol Syst Biol 12:894. <https://doi.org/10.15252/msb.20167137>
109. Kalita MK, Sargsyan K, Tian B et al (2011) Sources of cell-to-cell variability in canonical nuclear factor- κ B (NF- κ B) signaling pathway inferred from single cell dynamic images. J Biol Chem 286:37741–37757. <https://doi.org/10.1074/jbc.M111.280925>
110. Dharmarajan L, Kaltenbach HM, Rudolf F, Stelling J (2019) A simple and flexible computational framework for inferring sources of heterogeneity from single-cell dynamics. Cell Syst 8:15–26.e11. <https://doi.org/10.1016/j.cels.2018.12.007>
111. Hasenauer J, Waldherr S, Doszczak M et al (2011) Identification of models of heterogeneous cell populations from population snapshot data. BMC Bioinformatics 12:1–15. <https://doi.org/10.1186/1471-2105-12-125>
112. Loos C, Hasenauer J (2019) Mathematical modeling of variability in intracellular signaling. Curr Opin Syst Biol 16:17–24
113. Feng X-H, Deryck R (2005) Specificity and versatility in tgf-beta signaling through Smads. Annu Rev Cell Dev Biol 21:659–693. <https://doi.org/10.1146/annurev.cellbio.21.022404.142018>
114. Massagué J, Seoane J, Wotton D (2005) Smad transcription factors. Genes Dev 19: 2783–2810
115. Heldin CH, Landström M, Moustakas A (2009) Mechanism of TGF- β signaling to growth arrest, apoptosis, and epithelial-mesenchymal transition. Curr Opin Cell Biol 21:166–176
116. Moustakas A, Heldin CH (2007) Signaling networks guiding epithelial-mesenchymal transitions during embryogenesis and cancer progression. Cancer Sci 98:1512–1520. <https://doi.org/10.1111/j.1349-7006.2007.00550.x>
117. Meng XM, Nikolic-Paterson DJ, Lan HY (2016) TGF- β : the master regulator of fibrosis. Nat Rev Nephrol 12:325–338
118. Ikushima H, Miyazono K (2010) TGF β signalling: a complex web in cancer progression. Nat Rev Cancer 10:415–424
119. Mullen AC, Orlando DA, Newman JJ et al (2011) Master transcription factors determine cell-type-specific responses to TGF- β signaling. Cell 147:565–576. <https://doi.org/10.1016/j.cell.2011.08.050>
120. Piek E, Ju WJ, Heyer J et al (2001) Functional characterization of transforming growth factor β signaling in Smad2- and Smad3-deficient fibroblasts. J Biol Chem 276: 19945–19953. <https://doi.org/10.1074/jbc.M102382200>
121. Nicolás FJ, Hill CS (2003) Attenuation of the TGF- β -Smad signaling pathway in pancreatic tumor cells confers resistance to TGF- β -induced growth arrest. Oncogene 22:3698–3711. <https://doi.org/10.1038/sj.onc.1206420>
122. Schmierer B, Hill CS (2007) TGF β -SMAD signal transduction: molecular specificity and functional flexibility. Nat Rev Mol Cell Biol 8: 970–982
123. He K, Yan X, Li N et al (2015) Internalization of the TGF- β type I receptor into caveolin-1 and EEA1 double-positive early endosomes. Cell Res 25:738–752. <https://doi.org/10.1038/cr.2015.60>
124. Li Y, Lee M, Kim N et al (2018) Spatiotemporal control of TGF- β signaling with light. ACS Synth Biol 7:443–451. <https://doi.org/10.1021/acssynbio.7b00225>
125. Zieba A, Pardali K, Söderberg O et al (2012) Intercellular variation in signaling through the TGF- β pathway and its relation to cell density and cell cycle phase. Mol Cell Proteomics 11:M111.013482. <https://doi.org/10.1074/mcp.M111.013482>
126. Warmflash A, Zhang Q, Sorre B et al (2012) Dynamics of TGF- β signaling reveal adaptive and pulsatile behaviors reflected in the nuclear localization of transcription factor Smad4. Proc Natl Acad Sci U S A 109:E1947–E1956. <https://doi.org/10.1073/pnas.1207607109>
127. Clarke DC, Brown ML, Erickson RA, et al (2009) Transforming growth factor depletion is the primary determinant of Smad signaling kinetics. Mol Cell Biol 29:2443–2455. <https://doi.org/10.1128/mcb.01443-08>
128. Vilar JMG, Jansen R, Sander C (2006) Signal processing in the TGF- β superfamily ligand-receptor network. PLoS Comput Biol 2:

- 0036–0045. <https://doi.org/10.1371/journal.pcbi.0020003>
129. Schmierer B, Tournier AL, Bates PA, Hill CS (2008) Mathematical modeling identifies Smad nucleocytoplasmic shuttling as a dynamic signal-interpreting system. *Proc Natl Acad Sci U S A* 105:6608–6613. <https://doi.org/10.1073/pnas.0710134105>
130. Wegner K, Bachmann A, Schad JU et al (2012) Dynamics and feedback loops in the transforming growth factor β signaling pathway. *Biophys Chem* 162:22–34. <https://doi.org/10.1016/j.bpc.2011.12.003>
131. Zi Z, Klipp E (2007) Constraint-based modeling and kinetic analysis of the smad dependent TGF- β signaling pathway. *PLoS One* 2: 1–11. <https://doi.org/10.1371/journal.pone.0000936>
132. Zi Z, Feng Z, Chapnick DA et al (2011) Quantitative analysis of transient and sustained transforming growth factor- β signaling dynamics. *Mol Syst Biol* 7:1–12. <https://doi.org/10.1038/msb.2011.22>
133. Vizán P, Miller DSJJ, Gori I et al (2013) Controlling long-term signaling: receptor dynamics determine attenuation and refractory behavior of the TGF- β pathway. *Sci Signal* 6:1–12. <https://doi.org/10.1126/scisignal.2004416>
134. Chung SW, Miles FL, Sikes RA et al (2009) Quantitative modeling and analysis of the transforming growth factor β signaling pathway. *Biophys J* 96:1733–1750. <https://doi.org/10.1016/j.bpj.2008.11.050>
135. Melke P, Jönsson H, Pardali E et al (2006) A rate equation approach to elucidate the kinetics and robustness of the TGF- β pathway. *Biophys J* 91:4368–4380. <https://doi.org/10.1529/biophysj.105.080408>
136. Zi Z, Chapnick DA, Liu X (2012) Dynamics of TGF- β /Smad signaling. *FEBS Lett* 586: 1921–1928
137. Clarke DC, Liu X (2008) Decoding the quantitative nature of TGF- β /Smad signaling. *Trends Cell Biol* 18:430–442. <https://doi.org/10.1016/j.tcb.2008.06.006>
138. Lucarelli P, Schilling M, Kreutz C et al (2018) Resolving the combinatorial complexity of Smad protein complex formation and its link to gene expression. *Cell Syst* 6:75–89.e11. <https://doi.org/10.1016/j.cels.2017.11.010>
139. Cellière G, Fengos G, Hervé M, Iber D (2011) Plasticity of TGF- β signaling. *BMC Syst Biol* 5:184. <https://doi.org/10.1186/1752-0509-5-184>
140. Nicklas D, Saiz L (2013) Characterization of negative feedback network motifs in the TGF- β signaling pathway. *PLoS One* 8: e83531. <https://doi.org/10.1371/journal.pone.0083531>
141. Shimizu K, Gordon JB (1999) A quantitative analysis of signal transduction from activin receptor to nucleus and its relevance to morphogen gradient interpretation. *Proc Natl Acad Sci U S A* 96:6791–6796. <https://doi.org/10.1073/pnas.96.12.6791>
142. Khatibi S, Zhu HJ, Wagner J et al (2017) Mathematical model of TGF- β signalling: feedback coupling is consistent with signal switching. *BMC Syst Biol* 11:1–15. <https://doi.org/10.1186/s12918-017-0421-5>
143. Inman GJ, Nicolás FJ, Hill CS (2002) Nucleocytoplasmic shuttling of Smads 2, 3, and 4 permits sensing of TGF- β receptor activity. *Mol Cell* 10:283–294. [https://doi.org/10.1016/S1097-2765\(02\)00585-3](https://doi.org/10.1016/S1097-2765(02)00585-3)
144. Miller DSJ, Schmierer B, Hill CS (2019) TGF- β family ligands exhibit distinct signalling dynamics that are driven by receptor localisation. *J Cell Sci* 132:jcs234039. <https://doi.org/10.1242/jcs.234039>
145. Krakowski AR, Laboureau J, Mauviel A et al (2005) Cytoplasmic SnoN in normal tissues and nonmalignant cells antagonizes TGF- β signaling by sequestration of the Smad proteins. *Proc Natl Acad Sci U S A* 102:12437–12442. <https://doi.org/10.1073/pnas.0504107102>
146. Albeck JG, Mills GB, Brugge JS (2013) Frequency-modulated pulses of ERK activity transmit quantitative proliferation signals. *Mol Cell* 49:249–261. <https://doi.org/10.1016/j.molcel.2012.11.002>
147. Bohn S, Hexemer L, Huang Z, Strohmaier L, Lenhardt S, Legewie S, Loewer A (State- and stimulus-specific dynamics of SMAD signalling determine fate decisions in individual cells. *Proc Natl Acad Sci*. (in press) <https://doi.org/10.1073/pnas.2210891120>
148. Casanovas G, Banerji A, d'Alessio F et al (2014) A multi-scale model of hepcidin promoter regulation reveals factors controlling systemic iron homeostasis. *PLoS Comput Biol* 10:e1003750. <https://doi.org/10.1371/journal.pcbi.1003750>

- Biol 10:e1003421. <https://doi.org/10.1371/journal.pcbi.1003421>
149. Enculescu M, Metzendorf C, Sparla R et al (2017) Modelling systemic iron regulation during dietary iron overload and acute inflammation: role of Hepcidin-independent mechanisms. PLoS Comput Biol 13: e1005322. <https://doi.org/10.1371/journal.pcbi.1005322>
150. Molina N, Suter DM, Cannavo R et al (2013) Stimulus-induced modulation of transcriptional bursting in a single mammalian gene. Proc Natl Acad Sci U S A 110:20563–20568. <https://doi.org/10.1073/pnas.1312310110>



Chapter 11

Quantitative Imaging Analysis of NF-κB for Mathematical Modeling Applications

Johannes Nicolaus Wibisana, Takehiko Inaba, Yasushi Sako,
and Mariko Okada

Abstract

Mathematical models can integrate different types of experimental datasets, reconstitute biological systems in silico, and identify previously unknown molecular mechanisms. Over the past decade, mathematical models have been developed based on quantitative observations, such as live-cell imaging and biochemical assays. However, it is difficult to directly integrate next-generation sequencing (NGS) data. Although highly dimensional, NGS data mostly only provides a “snapshot” of cellular states. Nevertheless, the development of various methods for NGS analysis has led to much more accurate predictions of transcription factor activity and has revealed various concepts regarding transcriptional regulation. Therefore, fluorescence live-cell imaging of transcription factors can help alleviate the limitations in NGS data by supplementing temporal information, linking NGS to mathematical modeling. This chapter introduces an analytical method for quantifying dynamics of nuclear factor kappaB (NF-κB) which forms aggregates in the nucleus. The method may also be applicable to other transcription factors regulated in a similar fashion.

Key words DT40, Fluorescence microscopy, Transcriptional regulation, Transcription factor, NF-κB, Image analysis

1 Introduction

Signaling networks are important for cells to adapt to changing environments and to regulate gene expression by modulating transcription factor activity. The nuclear factor kappa B (NF-κB) signaling pathway is critical for a variety of cellular developmental processes, such as cell differentiation and maturation [1]. Several mathematical models have been developed to explain the dynamics [2, 3] and gene regulation [4–6] of the NF-κB pathway, including the pathway associated with B-cell receptor (BCR) signaling [3, 7] (Fig. 1). These models and analyses have focused on the nuclear translocation dynamics of NF-κB, in which the nuclear NF-κB response shows a switch-like activation followed by oscillatory

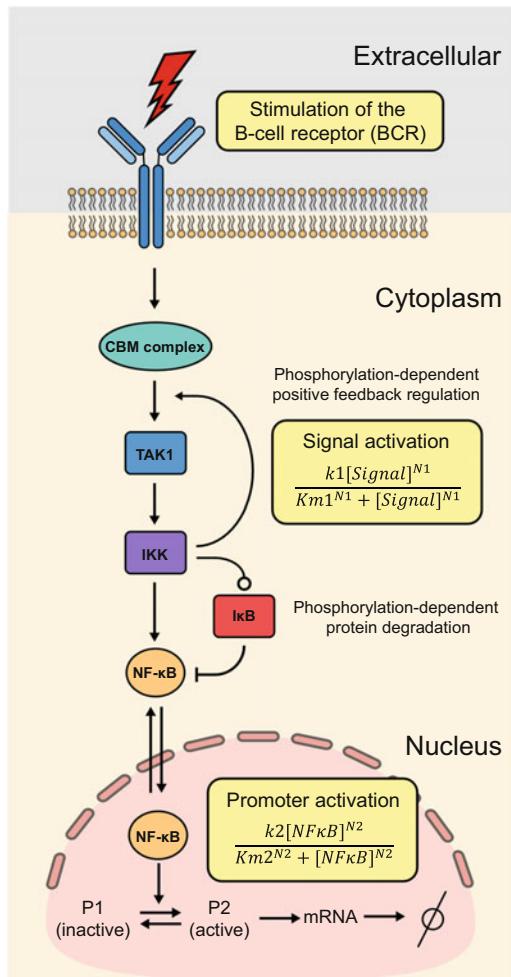


Fig. 1 Signaling and transcriptional regulation of NF-κB in B lymphocytes. Integrated reaction scheme of the NF-κB signaling network [7] and transcriptional activity based on a promoter transition model [8]. After stimulation of the IgM B-cell receptor (BCR) with an anti-IgM monoclonal antibody, the NF-κB pathway is activated. A complex containing CARMA1, BCL10, and MALT1 (CBM complex) is then activated and activates TAK1, consequently activating IKK which phosphorylates the inhibitor of kappa B (IκB). This modification promotes proteasome-mediated degradation of IκB and provides a positive feedback loop by enhancing activation of the CBM complex. This chain of events promotes the translocation of NF-κB proteins into the nucleus, where NF-κB acts as a transcription factor by driving the transition of target promoters from an inactive to active state, thereby promoting transcriptional activity

nuclear translocation in an array of different cell types and with a variety of stimuli [3, 9]. This NF-κB response has also been described as all-or-none at the single cell level [9–11]. Hence, the

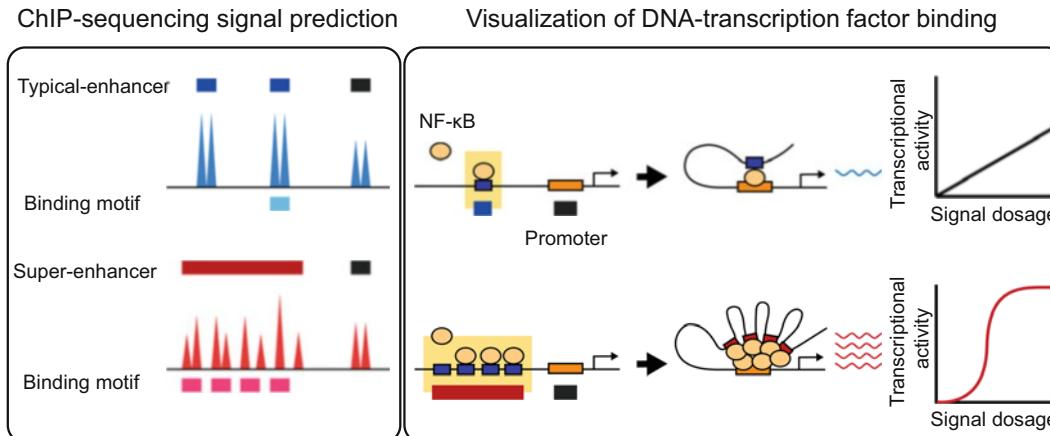


Fig. 2 Relationship between NGS data and predicted transcription factor binding to DNA. (Left) Schematic image of a hypothetical chromatin immunoprecipitation sequencing signal and the difference between typical and super-enhancers. A typical enhancer does not show cooperative binding and its transcriptional activity increases linearly with time. (Right) In the case of super-enhancers, the cooperative binding of transcription factors produces a switch-like transcriptional activity response and elevated mRNA production

unique properties of NF-κB dynamics have attracted the interest of many experimental and theoretical researchers.

Nuclear factor kappa B is also reported to be involved in the regulation of enhancers, specifically super-enhancers, which are groups of regulatory regions located in close genomic proximity that cooperatively activate gene transcription [12–14]. Super-enhancers are cell- and environment-specific and associated with switch-like gene expression, as opposed to typical enhancers that show graded gene expression [13, 15] (Fig. 2). Super-enhancers are usually identified by histone H3 lysine 27 acetylation (H3K27Ac) chromatin immunoprecipitation sequencing (ChIP-seq), transcription factor ChIP-seq, or assay for transposase-accessible chromatin sequencing (ATAC-seq) data using the ranking of super enhancer (ROSE) algorithm, which groups ChIP or ATAC peaks located in close proximity as typical- or super-enhancers by their signal intensity [14, 16]. The binding of transcription factors to DNA involved in super-enhancer regulation is thought to be highly cooperative, leading to the formation of liquid-like condensates [14], affecting DNA binding kinetics [17, 18].

The kinetics of super-enhancer formation are visualized using fluorescently labeled coactivators, such as BRD4 and MED1, which simultaneously occupy enhancer regions with transcription factors [14, 18, 19]. This process can be attributed to the formation of nuclear aggregates [18]. The super-enhancer-like properties of such aggregates can be confirmed by observing their disruption after treatment with the BRD4 inhibitor (JQ1) or 1,6-hexanediol

and also the rapid rate of recovery after photobleaching [12, 14, 18]. In addition, there are also a few other examples of direct visualization of transcription factor activity on DNA [19, 20].

Mathematical model and NGS data analysis have been indispensable in the analysis of signaling systems. The input-output relationship between the dynamics of nuclear translocation of a transcription factor and gene expression can be explained by a simple mathematical model [8]. In the case of NF- κ B, a similar mathematical model can be used to describe the switch-like transcriptional activation of super-enhancers in bulk cell populations (Figs. 1 and 2) [13]. The parameters of this model are derived from fluorescence imaging analyses and epigenetic NGS, indicating the number of transcription factors binding at enhancer regions.

Under certain conditions, NF- κ B forms a nuclear focus that share the same biophysical properties with reported super-enhancers [7, 14]. By quantifying signal strength, the number of NF- κ B molecules in each focus can be estimated and cross-evaluated using NF- κ B ChIP-seq. In this chapter, we describe a method to quantify nuclear NF- κ B foci through single-cell fluorescence imaging. This method uses freely available software and uses custom premade programming scripts and package. These results can be used to link the NGS-predicted transcription factor activity with a mathematical model of the NF- κ B signaling network (Fig. 3). NF- κ B nuclear translocation follows an all-or-none response, originally observed in the NF- κ B heterodimer component RelA-GFP overexpressing DT40 chicken lymphoma cells following BCR stimulation [7]. This response can be analyzed by

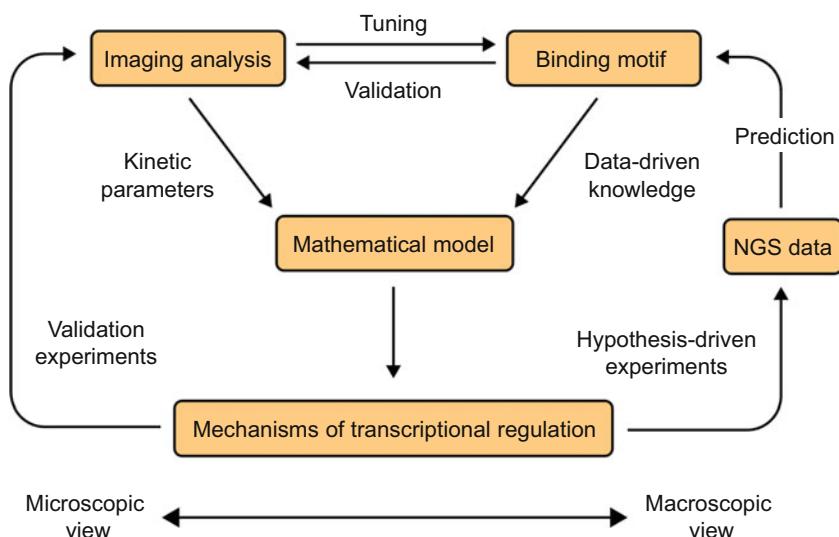


Fig. 3 Overview of the research platform for elucidating transcription activity based on imaging analysis, NGS data analysis, and mathematical modeling

quantifying RelA-GFP nuclear foci, which is related to the cooperative NF-κB assembly and formation of super-enhancers. Here, we introduce a method for the quantification and graphical visualization of nuclear foci, which is not only limited to NF-κB, but can also potentially be applicable to other transcription factor signaling pathways [21, 22].

2 Materials

2.1 Cell Culture

DT40 chicken B lymphoma cells overexpressing RelA-GFP [7] were used to analyze NF-κB foci formation. DT40 cells exhibit high homologous recombination rates, useful for pathway rewiring required for validation experiments [23]. DT40 cells grow in suspension and are stimulated with 10 µg/mL of M4, a monoclonal anti-IgM antibody that crosslinks the BCR to activate the NF-κB pathway [7]. The following materials were used to culture DT40 cells:

1. Culture medium: RPMI-1640 without phenol red, 10% fetal bovine serum, 1% chicken serum, 75 µM 2-mercaptoethanol, 1 mM sodium pyruvate, 1% Penicillin-Streptomycin solution, 1% 100× MEM nonessential amino acids solution, 2 mmol/L L-glutamine.
2. 39 °C humidified incubator containing 5% CO₂.
3. Appropriate sterile container such as a plastic culture dish or flask.

Cells were passaged every other day, where 4×10^5 cells were transferred to 3 mL of fresh medium. Cell culture was scaled as appropriate.

2.2 Microscopy and Image Acquisition Software

In our analysis, an inverted microscope IX81 (Olympus, Japan) equipped with CSU-X1 confocal scanner unit (Yokogawa, Japan), and oil-immersion objective (100×, NA 1.45) was used to obtain images. MetaMorph software (Molecular Devices, USA) was used to obtain 13 Z-slices of the stack image at 1 µm increments in the Z-direction. The image resolution was 512 × 512 pixels, in which 1 pixel = 0.16 µm.

2.3 Image and Data Analysis

1. FIJI, a distribution of the freely available “ImageJ” image processing software by the National Institutes of Health, USA (<https://imagej.net/software/fiji/>).
2. R (version $\geq 3.6.0$), an open-source programming language for statistical computing (<https://www.r-project.org/>).
3. RStudio, integrated development environment for R (<https://www.rstudio.com/>).

In this analysis, R and RStudio are used for the processing and analysis of quantified data obtained from ImageJ. ImageJ macro and R package used in this analysis can be obtained from: <https://github.com/okadalabipr/findfoci>.

3 Methods

3.1 Image Analysis Using FIJI-ImageJ

In this section, the detection of the nuclear foci from the cells is performed. First, calculate the average of the bottom eight slices of the image to identify cells. A Gaussian blur operation is performed to remove noise. Thresholding setting using the mean thresholding function is then performed along with watershed to obtain the cell mask. In this step, cells are detected and filtered for specified parameters such as circularity and size. A circle with a set diameter is drawn from the center of the cell ROI (Region of Interest), and only the foci inside this diameter are detected.

Focus detection is performed by first applying a custom sharpening kernel based on a Gaussian distribution to differentiate spatially close foci. An ROI of a set diameter is drawn on top of the foci for every layer of the image stack. For each focus, thresholding is performed according to the maximum pixel value. The output containing the coordinates of foci and cells is saved in the specified directory and used for visualization. Pipeline used in this analysis is listed in Table 1.

3.2 Procedure for Sample Image Analysis

1. Open FIJI and open the image file (File > Open) (Fig. 4a).
2. Drag and drop the macro file (focicount.ijm) into the FIJI window to open it.
3. Several parameters need to be modified based on the experimental conditions. These can be specified by editing the macro file (Fig. 4b) using the configuration shown in Table 2. For the sample images, the pixel size is 0.16 $\mu\text{m}/\text{pixel}$.
4. The parameters for cell detection can be changed by changing the line containing “Analyze Particles,” where the size is 600–infinity and circularity is 0.70–1.00 by default. For larger or smaller cell sizes, it might be beneficial to change the lower

Table 1
Pipeline for quantitative foci analysis using ImageJ and R

Step 1 (Image quantification)	Identify cells Perform foci counting on the mask created after identifying cell positions
Step 2 (Data analysis)	Assign foci to their appropriate cells based on their positions Plot scatter and line graph to visualize and summarize the foci number in individual cells

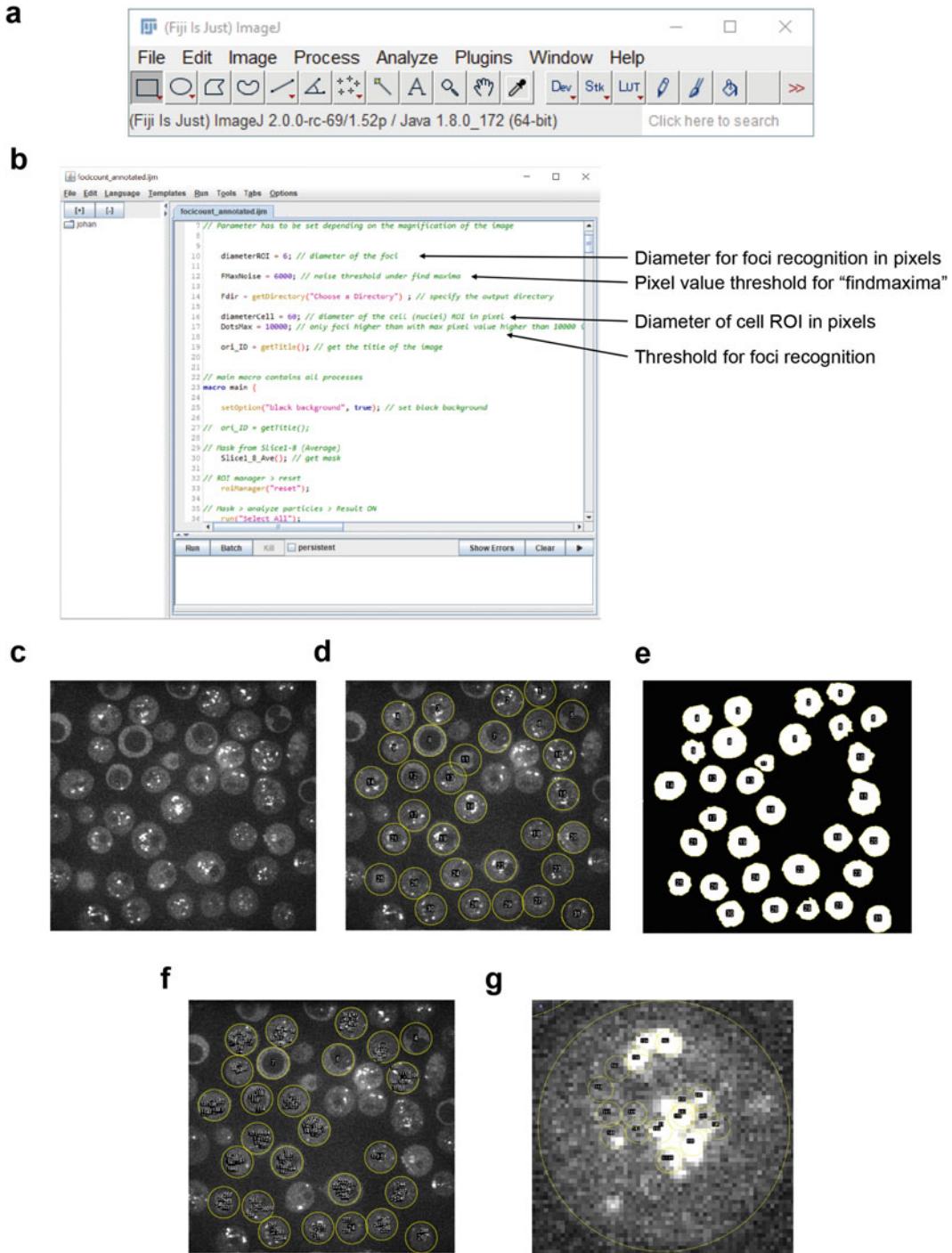


Fig. 4 FIJI windows during analysis. (a) FIJI main window. (b) Macro window. (c) Image window after loading image to FIJI. (d) Image window showing detected cells. (e) Interactive cell mask window. (f) Image window showing annotated foci. (g) Enlarged annotated image window

Table 2
Parameters used in foci detection

Parameter	Description	Default value	Note
diameterROI	Diameter of foci in pixels	6	This defaults to 0.96 μm . Foci are assumed to be approximately 0.96 μm in diameter
diameterCell	Diameter of cell in pixels	60	Foci counting will be based on this diameter, with the center of each detected cell. This defaults to 9.6 μm for the sample image
FMaxNoise	Noise threshold under find maxima	6000	This parameter should be changed depending on the experimental conditions
DotsMax	Detect foci with max pixel value higher than 10,000	10,000	This parameter should be changed depending on the experimental conditions

range from 600 to a larger or smaller number, respectively. For noncircular cells, the range for circularity can be changed to better accommodate the cell shape.

5. Set the measurement options (Analyze > Set Measurements). Check the following: “Area,” “Min & max gray value,” “Center of mass,” “Integrated density,” “Mean gray value,” “Perimeter,” “Stack position,” and “Display label”. This is necessary to generate the data needed for subsequent analyses.
6. Run the macro in the macro window (Run > Run) or by the shortcut “ctrl + R.” This will execute all codes in the ImageJ script.
7. Specify the output directory in the “imagej_output” directory. After specifying the directory, the average of the first eight slices from the bottom of the image is calculated. Following this, Gaussian blur with sigma 2 is then performed on the averaged image along with background subtraction to remove noise. Finally, an automatic local threshold is performed using the mean method with a 50-radius (these parameters can be changed according to the experimental conditions). Next, the holes are filled, and watershed is performed to separate the cells from each other.
8. The windows containing the cell mask and detected cells will be displayed (Fig. 4d, e). Manually correct the cell mask (e.g., remove dead or abnormal cells) by manipulating the ROI mask window. This creates a circular cell ROI for each cell, using cell diameters previously set in Table 1.
9. Click “OK” after confirming the ROIs. Loop cell mask to refresh the cell mask (optional). This will show the cell mask containing selected cells.

10. Click Continue to proceed with foci counting. Before counting, the macro executes the function “Convolve...” to perform spatial convolution using a specified custom kernel. This kernel sharpens images and helps distinguish spatially close foci. To enhance structures of different pixel sizes, the kernel can be modified by modifying the convolution matrix of the “Gauss_convolve” function of the ImageJ macro.
11. Following this, the macro executes the ImageJ function “find maxima” to find the potential foci with the threshold “FMax-Noise.” Then, foci inside the circular cell ROI are detected at the diameter “diameterROI” (Table 1) and centered at the maxima value detected by “find maxima.” These foci are then filtered according to the “DotsMax” (Table 1) to check if the maximum value of each focus after applying a custom kernel is greater than the specified value. The results are displayed as the ROIs in the image window (Fig. 4f, g).
12. The ROI file and csv file containing information about the foci and the cells are saved in the specified directory. The csv file contains several columns containing different foci parameters from the original image. The columns required for subsequent data analysis are the area and X and Y coordinates; the rows each represent the generated ROI (cells and foci). Along with the csv file, the processed images and foci are also exported as a montage for manual checking in the same directory.
13. This procedure should be repeated for other image samples.

3.3 Data Analysis Using R and RStudio

R and RStudio are used to visualize and analyze the quantified data obtained from ImageJ analysis. In this stage, the csv file from the output file of ImageJ analysis is read, and the foci ROI is assigned to each cell ROI according to the *x* and *y* coordinates using the column containing area. The final table containing the number of foci for each cell at each time point is generated as a csv file and a plot through user interface.

Example image data with the following naming: For example, for the file “20190816_foci_0.1ugml/20190816_foci_0.1ugml_10min_03.TIF-roi-6-FMax-6000_10000.csv,” the time unit is “min” and the dose “ugml.” Both dose and time strings should have an underscore before and after its string for automatic detection.

1. Install the package from R console.

```
devtools::install_github("okadalabipr/findfoci")
```

2. Execute the “count_foci” function while specifying the ImageJ output directory, dose units, and time units in quotation marks as shown below.

```
df <- count_foci("imagej_output", dose = "ugml",
time = "min")
```

These enable the function to detect the dose and time units according to the file name. The cell size is automatically determined by the ROI with the maximum area and the ROIs with a smaller area are regarded as foci. This function reads the csv files in the “imagej_output” directory and assigns foci to the appropriate cells. This assignment is based on the X and Y coordinates of the foci ROI and is matched with the cell ROI coordinates. Foci and cell ROI are distinguished by the ROI size. The assigned foci and cells will then be assigned to the variable “df.” Then, the resulting csv file can be saved using the function “write.csv()” onto the csv file “result.csv” containing data on the number of foci for each cell ROI.

```
write.csv(df, "result.csv")
```

3. To visualize the data in a plot, run the following command:

```
heatscatter()
```

This command runs the shiny application, which is an interactive application, for the visualization of the counted data (Fig. 5). The shiny app contains an interface that allows users to change various visualization parameters. To load the data for visualization, open the csv file using the “Browse...” button and select the csv file “result.csv.” After loading the data, a jitter plot colored by density should be created with the default settings. The available parameters are described in Table 3.

4. The output graph summarizes the data by plotting the number of foci against time in a jitter plot. The individual dots show each cell colored according to the density. The graph is summarized by either mean or median of foci number and is represented by a line graph. Right-click on the plot and select the option to save or copy the plot in png file format.

In summary, this workflow can be used to quantitatively evaluate fluorescence microscopy images of the formation of possible NF- κ B super-enhancers in DT40 cells. While DT40 grows in suspension, this analysis workflow can also be applied to quantify other transcription factors that form foci in adherent cells. The ImageJ macro can be extended to quantify other properties of the foci, such as morphology or intensity. The R package can be modified accordingly to accommodate changes in the ImageJ result output and for different visualization methods.

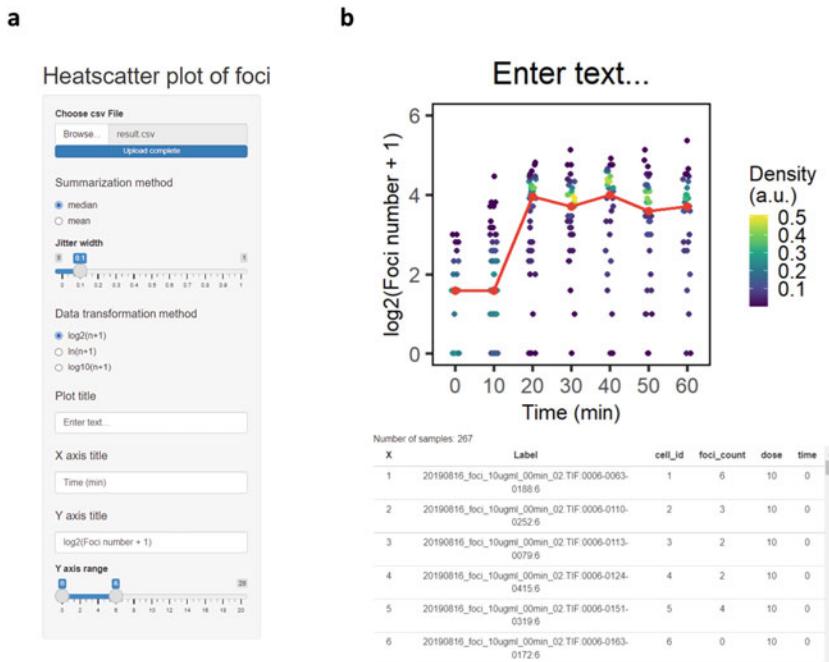


Fig. 5 Shiny application user interface for data visualization. **(a)** The sidebar on the left side shows the possible modifications that can be made to the plot. First, a csv file can be loaded using the “Browse...” button, and then a summarization function for the line graph can be chosen by using the next buttons (mean or median). The jitter width is cosmetic and changes the horizontal jitter of the data points. The data transformation method allows selection of the log transformation method that is used to shrink outliers and make the data easier to visualize. The plot title, x- and y-axis titles, can be customized by replacing the default text. Finally, the range of the y-axis can be customized by using the slider. **(b)** Visualization of the data by using a heat-scatter plot. The plot is shown at the top of the page, and the csv file used for generating it is shown at the bottom of the page, with the total number of samples indicated at the top of the table

Table 3
Parameters used in the visualization application

Parameter	Description	Default value	Notes
Summarization method	How the data are summarized to produce the line graph	Median	Median may be able to better summarize the data in the presence of outliers
Data transformation method	Log transformation of data. This is performed to shrink the data for easier visualization	$\log_2(n+1)$	Other transformation methods such as log10 and ln are available
Y-axis range	Lower and upper range of the Y-axis	0–6	If set too low, data might get cut off and produce an inaccurate summary

4 Note

While we used a spinning disk confocal microscope, we have also found that other microscopes including widefield and laser scanning confocal microscopes with comparable resolution were also able to capture the dynamics of NF- κ B foci in DT40 cells. Furthermore, it should be noted that for DT40 cells, it is important to keep the temperature stable during live imaging or the response to stimuli will be diminished.

Several mathematical models have been published that link the signaling network with the nuclear transcriptional activity of NF- κ B. For mathematical modeling of the NF- κ B signaling pathway, several ordinary differentiation equation (ODE) models reconstructed using the Python programming language are freely available at https://github.com/okadalabipr/cancer_modeling.

5 Conclusions

NGS data analysis has become a very commonly used method in modern biology. Decades ago, prediction of transcription factor activity was performed by examining the statistical enrichment of transcription motifs only from RNA-seq or microarray data. Therefore, predictions included many false positives. However, current ChIP-seq methods, combined with ATAC-seq [24] and other epigenetic sequencing methods, have become highly quantitative and can provide very reliable predictions of transcription factor and subsequent transcriptional activity. Moreover, the combination of single-cell ATAC and RNA sequencing was able to uncover the potential molecular mechanism of heterogeneous transcriptional activation of NF- κ B super-enhancer-regulated genes as opposed to switch-like transcriptional activation observed in bulk [14].

Mathematical modeling can be a very powerful tool for predicting underlying molecular mechanisms from various types of experimental datasets. However, they rely heavily on known information about the network topology. Bayesian network modeling and other statistical methods [25, 26] can be fully data-driven to infer network topology but require an enormous amount of data. The combination of NGS and mathematical modeling can complement each other, is more data-driven, and can fill in the gaps between these two methods.

Here, imaging analysis is used to obtain kinetic parameters of transcription factors, validate predictions from NGS data, and provide quantitative data to train mathematical models. Although not many examples have yet been described, a combination of live cell imaging, NGS, and mathematical modeling may prove valuable in deciphering unknown cellular functions and mechanism. To realize

this prediction and validate research findings, a comprehensive package of analytical tools must be developed in the future.

Acknowledgments

We thank Mr. Hiroki Michida and Dr. Hiroaki Imoto for discussions on bioinformatics analysis and mathematical modeling, respectively. J.N.W. was supported by the Honjo International Scholarship Foundation. M.O. was supported by JSPS KAKENHI Grant No. 15KT0084, 17H06299, 17H06302, and 18H04031, JST-Mirai program No. JPMJMI19G7, JST-CREST grant JPMJCR21N3, the Takeda Science Foundation, and the Uehara Memorial Foundation.

References

- Sasaki Y, Iwai K (2015) Roles of the NF-κB pathway in B-lymphocyte biology. In: Kurosaki T, Wienands J (eds) B cell receptor signaling. Current topics in microbiology and immunology. Springer Verlag, pp 177–209
- Cheong R, Hoffmann A, Levchenko A (2008) Understanding NF-κB signaling via mathematical modeling. *Mol Syst Biol* 4:192. <https://doi.org/10.1038/msb.2008.30>
- Inoue K, Shinohara H, Behar M et al (2016) Oscillation dynamics underlie functional switching of NF-κB for B-cell activation. *NPJ Syst Biol Appl* 2:16024. <https://doi.org/10.1038/npjbsa.2016.24>
- Nelson DE, Ihekweazu AEC, Elliott M et al (2004) Oscillations in NF-κB signaling control the dynamics of gene expression. *Science* (80-) 306:704–708. <https://doi.org/10.1126/science.1099962>
- Ashall L, Horton CA, Nelson DE et al (2009) Pulsatile stimulation determines timing and specificity of NF-κB-dependent transcription. *Science* (80-) 324:242–246. <https://doi.org/10.1126/science.1164860>
- Zambrano S, De Toma I, Piffer A et al (2016) NF-κB oscillations translate into functionally related patterns of gene expression. *Elife* 5: e09100. <https://doi.org/10.7554/elife.09100>
- Shinohara H, Behar M, Inoue K et al (2014) Positive feedback within a kinase signaling complex functions as a switch mechanism for NF-κB activation. *Science* (80-) 344:760–764. <https://doi.org/10.1126/science.1250020>
- Hao N, O’Shea EK (2012) Signal-dependent dynamics of transcription factor translocation controls gene expression. *Nat Struct Mol Biol* 19:31–39. <https://doi.org/10.1038/nsmb.2192>
- Tay S, Hughey JJ, Lee TK et al (2010) Single-cell NF-κB dynamics reveal digital activation and analogue information processing. *Nature* 466:267–271. <https://doi.org/10.1038/nature09145>
- Lee REC, Walker SR, Savery K et al (2014) Fold change of nuclear NF-κB determines TNF-induced transcription in single cells. *Mol Cell* 53:867–879. <https://doi.org/10.1016/j.molcel.2014.01.026>
- Kellogg RA, Tian C, Lipniacki T et al (2015) Digital signaling decouples activation probability and population heterogeneity. *Elife* 4: e08931. <https://doi.org/10.7554/elife.08931>
- Brown JD, Lin CY, Duan Q et al (2014) NF-κB directs dynamic super enhancer formation in inflammation and atherogenesis. *Mol Cell* 56:219–231. <https://doi.org/10.1016/j.molcel.2014.08.024>
- Michida H, Imoto H, Shinohara H et al (2020) The number of transcription factors at an enhancer determines switch-like gene expression. *Cell Rep* 31:107724. <https://doi.org/10.1016/j.celrep.2020.107724>
- Wibisana JN, Inaba T, Shinohara H et al (2022) Enhanced transcriptional heterogeneity mediated by NF-κB super-enhancers. *PLoS Genet* 18;6:e1010235. <https://doi.org/10.1371/journal.pgen.1010235>
- Lovén J, Hoke HA, Lin CY et al (2013) Selective inhibition of tumor oncogenes by disruption of super-enhancers. *Cell* 153:320–334. <https://doi.org/10.1016/j.cell.2013.03.036>

16. Whyte WA, Orlando DA, Hnisz D et al (2013) Master transcription factors and mediator establish super-enhancers at key cell identity genes. *Cell* 153:307–319. <https://doi.org/10.1016/j.cell.2013.03.035>
17. Hnisz D, Shrinivas K, Young RA et al (2017) A phase separation model for transcriptional control. *Cell* 169:13–23. <https://doi.org/10.1016/j.cell.2017.02.007>
18. Sabari BR, Dall'Agnese A, Boija A et al (2018) Coactivator condensation at super-enhancers links phase separation and gene control. *Science* (80-) 361:eaar3958. <https://doi.org/10.1126/science.aar3958>
19. Liu Z, Tjian R (2018) Visualizing transcription factor dynamics in living cells. *J Cell Biol* 217: 1181–1191
20. Ochiai H, Sugawara T, Yamamoto T (2015) Simultaneous live imaging of the transcription and nuclear position of specific genes. *Nucleic Acids Res* 43:e127. <https://doi.org/10.1093/nar/gkv624>
21. Boija A, Klein IA, Sabari BR et al (2018) Transcription factors activate genes through the phase-separation capacity of their activation domains. *Cell* 175:1842–1855.e16. <https://doi.org/10.1016/j.cell.2018.10.042>
22. Tatavosian R, Kent S, Brown K et al (2019) Nuclear condensates of the Polycomb protein chromobox 2 (CBX2) assemble through phase separation. *J Biol Chem* 294:1451–1463. <https://doi.org/10.1074/jbc.RA118.006620>
23. Vasquez KM, Marburger K, Intody Z, Wilson JH (2001) Manipulating the mammalian genome by homologous recombination. *Proc Natl Acad Sci U S A* 98:8403–8410. <https://doi.org/10.1073/pnas.111009698>
24. Buenrostro JD, Giresi PG, Zaba LC et al (2013) Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. *Nat Methods* 10:1213–1218. <https://doi.org/10.1038/nmeth.2688>
25. Sachs K (2005) Causal protein-signaling networks derived from multiparameter single-cell data. *Science* (80-) 308:523–529. <https://doi.org/10.1126/science.1105809>
26. Krishnaswamy S, Spitzer MH, Mingueneau M et al (2014) Conditional density-based analysis of T cell signaling in single-cell data. *Science* (80-) 346:1250689. <https://doi.org/10.1126/science.1250689>



Chapter 12

Resolving Crosstalk Between Signaling Pathways Using Mathematical Modeling and Time-Resolved Single Cell Data

Fabian Konrath, Alexander Loewer, and Jana Wolf

Abstract

Crosstalk between signaling pathways can modulate the cellular response to stimuli and is therefore an important part of signal transduction. For a comprehensive understanding of cellular responses, identifying points of interaction between the underlying molecular networks is essential. Here, we present an approach that allows the systematic prediction of such interactions by perturbing one pathway and quantifying the concomitant alterations in the response of a second pathway. As the observed alterations contain information about the crosstalk, we use an ordinary differential equation-based model to extract this information by linking altered dynamics to individual processes. Consequently, we can predict the interaction points between two pathways. As an example, we employed our approach to investigate the crosstalk between the NF- κ B and p53 signaling pathway. We monitored the response of p53 to genotoxic stress using time-resolved single cell data and perturbed NF- κ B signaling by inhibiting the kinase IKK2. Employing a subpopulation-based modeling approach enabled us to identify multiple interaction points that are simultaneously affected by perturbation of NF- κ B signaling. Hence, our approach can be used to analyze crosstalk between two signaling pathways in a systematic manner.

Key words Signaling crosstalk, Quantitative modeling, p53 signaling, Genotoxic stress, Single cell analysis, Live cell imaging, Subpopulation-specific modeling, IKK β , NF- κ B signaling

1 Introduction

Signaling pathways are molecular networks that enable cells to transduce internal and external signals and are thereby crucial to appropriately respond to those signals. As signal transduction mediated by a defined pathway is affected by molecular interactions with other pathways, it is important to consider the impact of signaling crosstalk on the induced cellular response [1–3]. Hence, we developed a framework that allows to systematically identify putative interaction points between two pathways [4]. We selected the p53 and NF- κ B pathway as an example to identify points of interaction between those two networks. Both transcription factors, p53 as well as NF- κ B, play pivotal roles in the cellular stress response by regulating cell fate decisions [5, 6]. The activation

status of p53 is primarily regulated by its protein abundance [7]. The ubiquitin ligase Mdm2 mediates the ubiquitination of p53 and thereby induces proteasomal degradation of p53. Upon genotoxic stress, the kinases ATM, ATR, and DNA-PK are activated and in turn phosphorylate p53 and Mdm2 causing the degradation of Mdm2 and the accumulation of p53 [8–12]. Mdm2 as well as the phosphatase Wip1 are target genes of p53 and ensure the inactivation of p53 by reestablishing the Mdm2-mediated control of the p53 level. Such negative feedback mechanisms are also found in the regulation of NF- κ B activity. In the absence of stimuli, cytoplasmic NF- κ B is bound to one of its inhibitors, I κ B α , I κ B β , or I κ B ϵ , which prevent its translocation into the nucleus [13, 14]. Extracellular ligands such as TNF α as well as intracellular DNA damage cause the phosphorylation of the kinase IKK2, a subunit of the IKK complex that induces the degradation of I κ Bs and thereby enables NF- κ B-mediated transcription of its target genes including the I κ B encoding genes [15, 16]. To investigate the crosstalk between the p53 and NF- κ B pathway in a comprehensive and mechanistic manner, we combined time-resolved single cell measurements and mathematical modeling. To predict interaction points between both networks, we monitored single cell dynamics of p53 in the presence of DNA double-strand breaks with and without perturbation of NF- κ B signaling. Due to the negative feedback mechanism that regulates the p53 level, dynamics of p53 exhibit regular accumulation pulses in the presence of DNA double-strand breaks [17, 18]. Measuring the dynamics of single cells characterizes the molecular network in great detail, while the population dynamics may be distorted due to cell-to-cell heterogeneity. Therefore, an approach is required that allows to employ single cell dynamics while also taking its heterogeneity into account. With our computational approach, we sought to reproduce the observed p53 dynamics and identify model parameters that can be linked to molecular processes affected by the perturbation of NF- κ B signaling. Therefore, we used an ordinary differential equation-based model which can give mechanistic insights in the represented molecular network. In the following, we present in a compact and stepwise manner our framework that can now be applied to further investigate the crosstalk of interacting signaling pathways.

2 Extracting Information About Crosstalk from Altered Signaling Response

The basic concept of our approach is to experimentally perturb a signaling pathway, monitor the alterations in the response of a second pathway, and then use the information from the altered response to infer computationally the interaction points between both pathways (Fig. 1a). The response of the monitored pathway is

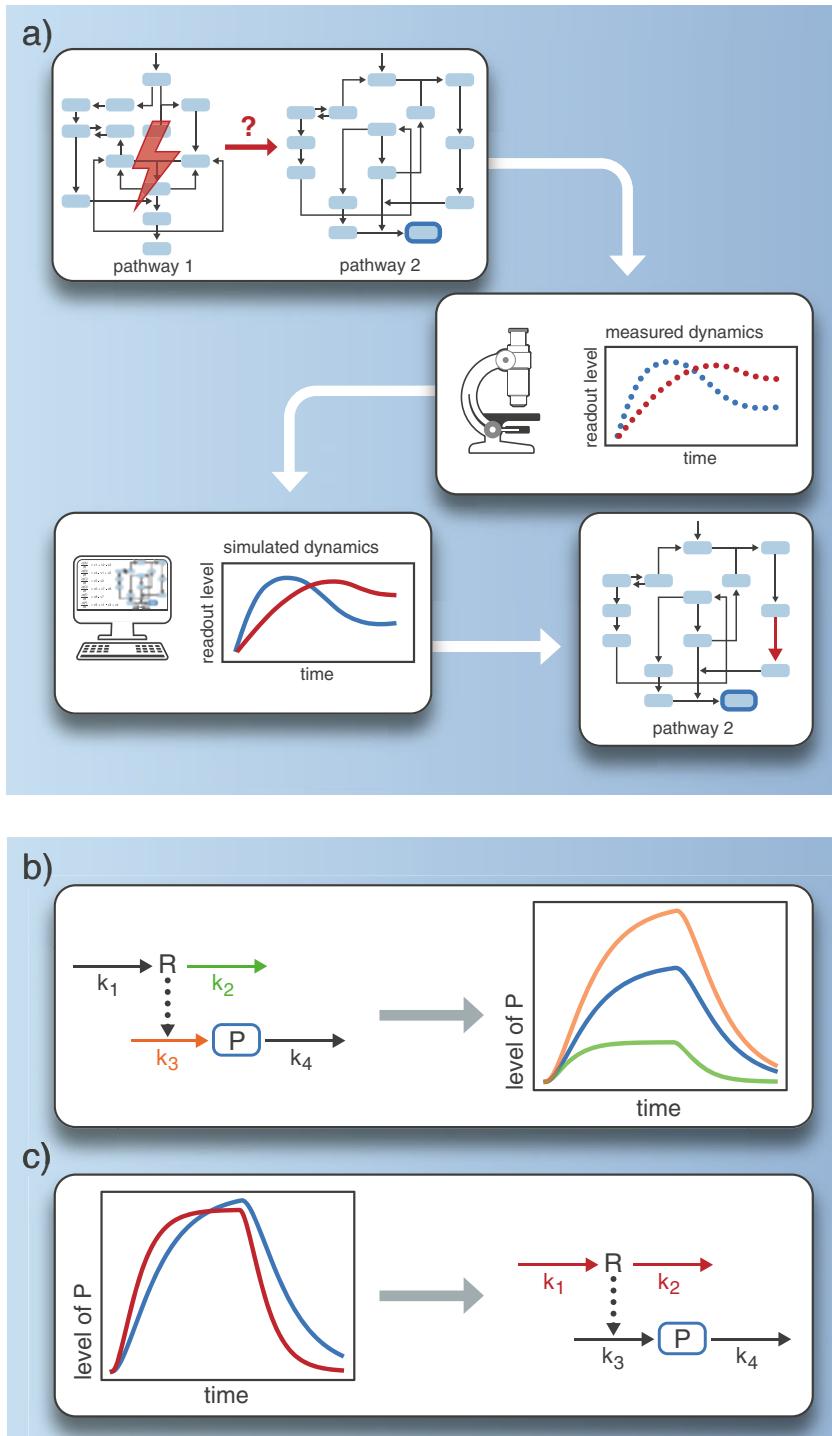


Fig. 1 Workflow of approach and link between model parameters and dynamics. **(a)** The crosstalk between two pathways is analyzed by perturbation of the interacting pathway (pathway 1) and measuring the levels of a selected network component of the second pathway (pathway 2, blue-framed box). The response of the monitored component to a given stimulus (blue dots) is altered by the perturbation of the interacting pathway (red dots). Model-based reconstruction of the altered response (blue and red lines) allows the computational

thereby represented by the dynamics of a network component of this pathway. Hence, we want to link a change in those dynamics to a change in a molecular process of the monitored pathway. An ordinary differential equation-based model is a powerful tool to resolve the link between dynamics and processes, as it can be used to mathematically describe biochemical interactions and thereby reconstruct a given molecular network. The parameters of those equations are defined as reaction rate constants that determine the rate of a specific reaction and thus represent a specific molecular process within the network. A simple and well-studied model [19–21] describing the transcription of a gene as well as the translation of the synthesized transcript into a protein is shown in Fig. 1b. Perturbing either the process representing transcription of the gene or degradation of the protein has a different impact on dynamics of the protein level. In our approach, we want to use this differential and specific effect of parameters on dynamics and derive from the altered dynamics of the monitored component those parameters that can explain the observed change (Fig. 1c). In our study [4], we experimentally perturbed NF- κ B signaling by inhibiting the kinase IKK2 and monitored alterations in the p53 response to DNA double-strand breaks. The response of p53 was quantified by tracking in single cells for 24 h the nuclear intensity of p53 in a fluorescent reporter cell line (Fig. 2a). Inhibition of IKK2 caused strong changes in the pulsatile dynamics of p53 (Fig. 2b). Additional analyses revealed that defined features of dynamics such as the timing of peaks were significantly altered [4]. A detailed description and presentation of our specific results can be found in [4, 22].

3 Model Selection and Parameterization

To link the observed changes in dynamics to a process within the monitored pathway, an appropriate ordinary differential equation-based model needs to be selected. With the aim to reproduce the response of the monitored pathway by applying parameter

Fig. 1 (continued) prediction of the process (red arrow) that is affected by the perturbation and therefore marks the interaction point between the two analyzed pathways. (b) Scheme of an ordinary differential equation-based model describing four biochemical processes: the transient transcription of a gene, translation of the synthesized mRNA (R) into a protein (P) and basal degradation of the mRNA and protein. The model parameters that are specific for a molecular process are indicated by the letter k . The corresponding simulation results show the time courses of protein P using nominal parameter values (blue line), the effect of an increase in k_2 resulting in an increased degradation rate of the mRNA (green line), or an increase in k_3 leading to an increased translation rate (orange line). (c) Simulation of altered dynamics (red line) that represents experimentally observed changes in the time course of the protein level. The changes in dynamics are used to identify those model parameters that need to be perturbed in order to reflect the observed dynamics (k_1 and k_2)

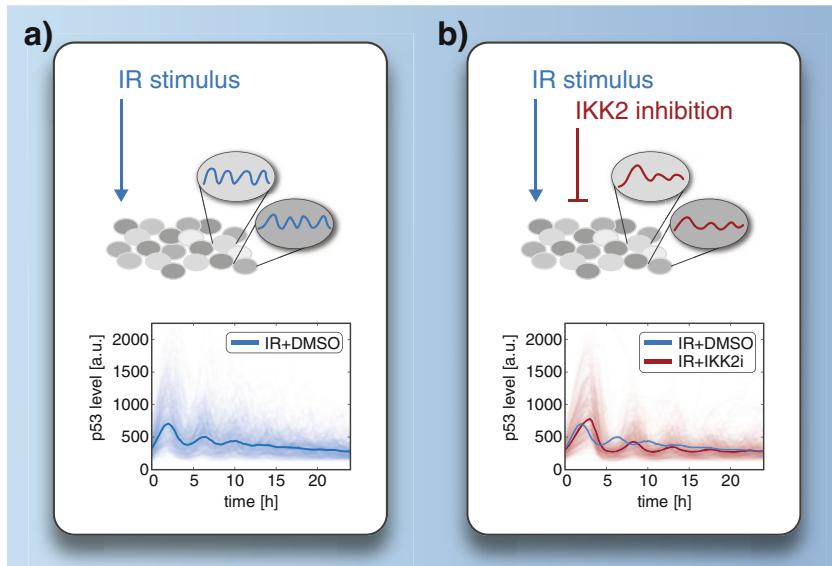


Fig. 2 Perturbing the NF- κ B signaling pathway modulates the response of p53 to DNA damage. **(a)** A population of A549 cells expressing fluorescently tagged p53 is treated with 10 Gy irradiation (IR) to generate genotoxic stress and thereby induce the pulsatile p53 response. The dynamics of p53 are tracked by quantifying the median nuclear fluorescence intensity in single cells using live-cell time-lapse microscopy. The bold blue line depicts the median of all trajectories and light blue-colored lines depict single cell trajectories. **(b)** Cells are irradiated with 10 Gy in combination with an inhibition of IKK2 activity. The quantified single cell trajectories are shown as light red lines. The bold red line denotes the median of the single cell trajectories and the bold blue line shows the median from the control experiment **(a)**. (The plots showing the quantified trajectories are reproduced from Konrath et al. [4] according to the CC BY license)

estimation, it is important to choose a model with an adequate level of detail. While an abstract model allows computational cost-efficient parameter inference and is less prone for overfitting, a more detailed model facilitates the identification of refined and meaningful interaction points between two pathways.

For the p53 signaling network, we modified a model that was developed by Batchelor et al. [23, 24]. It contains the DNA double-strand break-induced activation of p53 by kinase ATM and the p53-induced expression of its own inhibitors Wip1 and Mdm2 (Fig. 3). To be close to experimental observations and ensure precise predictions, we inferred parameter values by fitting the model to the unperturbed response of the monitored pathway (calibration data). In our study [4], we captured the calibration data, that is, the p53 response to DNA double-strand breaks, by employing time-resolved single cell imaging, which allows quantitative tracking of p53 dynamics (Fig. 2a). As the presented model is not representing the observed heterogeneity among single cell dynamics, a modeling strategy has to be chosen that accounts for the heterogeneous p53 response. There exist modeling efforts employing nonlinear mixed-effect modeling to reproduce single

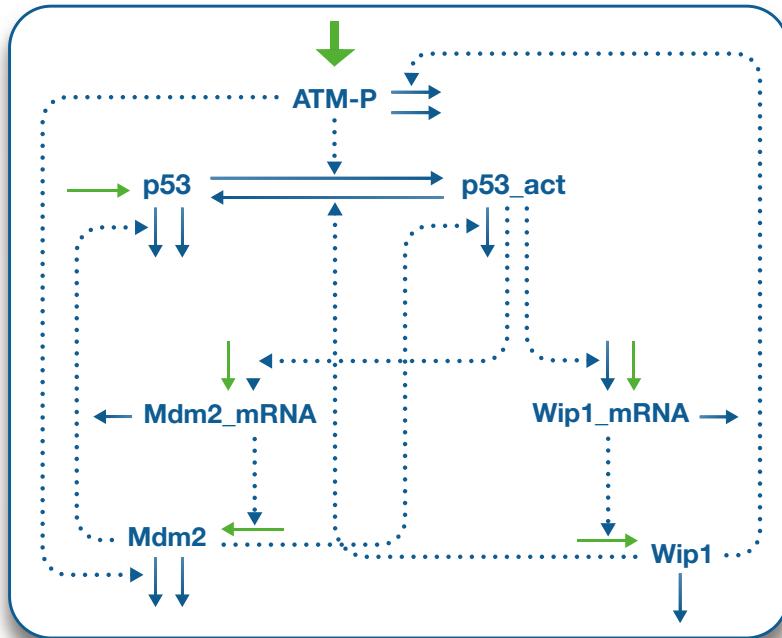


Fig. 3 Mathematical model of genotoxic p53 signaling. An ordinary differential equation-based model describes the irradiation-induced activation of ATM (ATM-P) and the concomitant activation of p53 (p53_act), transcription of the Mdm2 and Wip1 gene, as well as translation of Mdm2 and Wip1 mRNA. The dotted lines indicate regulated processes. The green arrows depict processes that are assumed to be affected by noise and are therefore defined in our modeling approach as subpopulation-specific (see Subheading 3.2 for details)

cell trajectories and thereby capture full heterogeneity among single cells [25, 26]. Depending on the pathways under investigations, these approaches are computationally expensive and require models that are capable to reproduce the noisy dynamics of an individual cell. Alternatively, we developed for our study of p53 dynamics on the single cell level a framework that is inspired by Strasen et al. [27] and that is based on reconstructing heterogeneous p53 dynamics on the subpopulation level. Specifically, single cell trajectories with similar dynamics are clustered into subpopulations, and a pool of subpopulation-specific models that retain the same network structure but allow for parameter variations within the subpopulations is developed. The subpopulation-specific models are then fitted to the averaged dynamics of subpopulations (Fig. 4). While parameter inference with our approach is computationally less demanding, it requires the clustering of trajectories and the generation of subpopulation-based models. In the following two sections, we describe the corresponding approaches that we used in our framework. Further details about the applied methods are given in [4, 22].

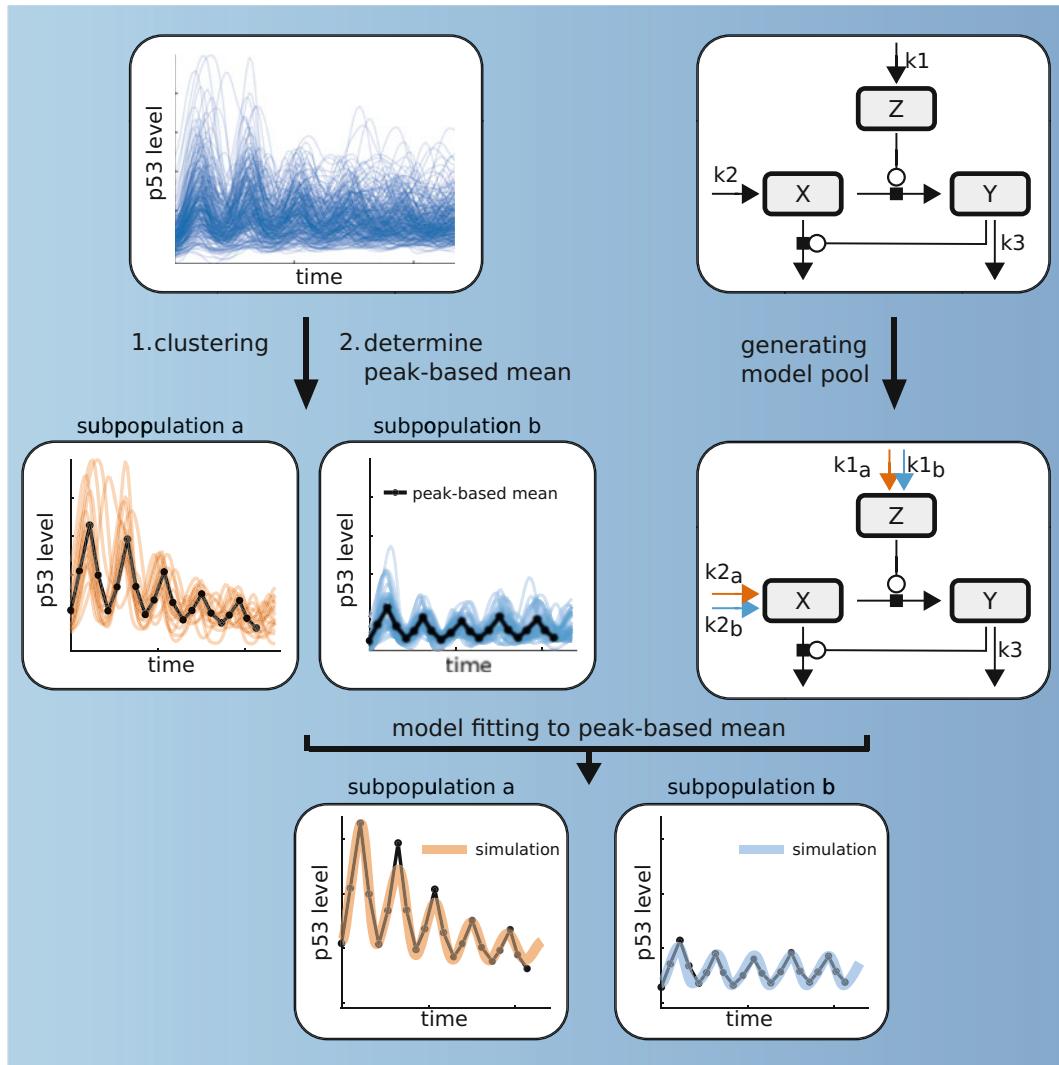


Fig. 4 Scheme of the subpopulation modeling framework. Single cell trajectories are clustered into subpopulations with similar dynamics (upper left panel). For each cluster, a peak-based mean is calculated, representing subpopulation-specific dynamics. To account for cellular heterogeneity and to reproduce the different dynamics of subpopulations, a pool of models is generated (upper right panel). In this example, the model pool comprises two models, one is specific for subpopulation *a*, the other one for subpopulation *b*. The production rates of *Z* and *X* are assumed to be susceptible to noise. Hence, parameters of these two processes are considered subpopulation-specific and therefore specific for an individual model. While k_{1a} and k_{2a} are specific for subpopulation *a* and thus assigned to one model, k_{1b} and k_{2b} are specific for subpopulation *b* and assigned to the second model. Importantly, both models share parameters such as k_3 , assuming that, for example, phosphorylation rates or degradation rates are not affected by noise. In a final step, the model pool is simultaneously fitted to the peak-based mean of each subpopulation. (The figure and legend are reproduced from Konrath et al. [4] according to the CC BY license)

3.1 Clustering of Single Cell Trajectories

The purpose of grouping single cell trajectories is to generate distinct clusters with relatively homogenous average dynamics that together reflect the heterogeneous p53 response. In our case, these clusters do not necessarily represent different cellular states but integrate all sources of cell-to-cell variability [28]. An optimal number of clusters results from grouping trajectories with small within-cluster variances while keeping the number of clusters in a range in which subpopulation-based modeling is still feasible.

As we observed in our data for a few individual trajectories strongly deviating dynamics from the overall population, we wanted to identify and neglect those trajectories in order to reduce the number of identified clusters and to prevent high within-cluster variances. The number of identified clusters is of importance, as it determines the size of the model pool and therefore the computational costs for parameter inference. High variances between trajectories within one cluster enhance the discrepancy between single cell dynamics and the mean dynamics of a cluster and thereby reduce the quality of the model in capturing cell-to-cell heterogeneity. For the identification of irregular dynamics, we quantified the number and timing of peaks of the oscillating p53 levels for each trajectory and compared them with the corresponding mean population values. Trajectories with values deviating more than two standard deviations from these population values were filtered out. To improve the identification of p53 peaks, we smoothed each trajectory by applying a Gaussian-weighted moving average. For the smoothing of pulsatile dynamics, it is important to ensure that the applied filter removes fluctuations without modulating the frequencies. This can be realized by choosing a small window size for the moving average compared to the frequency of the observed pulses.

Before the filtered trajectories can be subjected to clustering, dissimilarities between the trajectories have to be quantified. Depending on the shape of the analyzed dynamics, certain methods are better suited than others to quantify dissimilarities. Specifically, calculating the Euclidean distance is a simple and straightforward method. However, for dynamics exhibiting a recurrent pattern such as regular pulses, dynamic time warping can be applied to account for similarity of patterns among trajectories [27, 29]. This is done by nonlinear scaling of the time axis to align the two trajectories and compute their distance. An example of two single cell trajectories of p53 is shown in Fig. 5a. To apply dynamic time warping, the Euclidean distances between one data point from the first trajectory and all other data points of the second trajectory are computed, which is repeated for all remaining data points of the first trajectory. This way, the distance between data points from different time points is calculated based on the assumption of a potential shift in time between both trajectories. The resulting matrix that contains all distances between data points of the two trajectories is then used

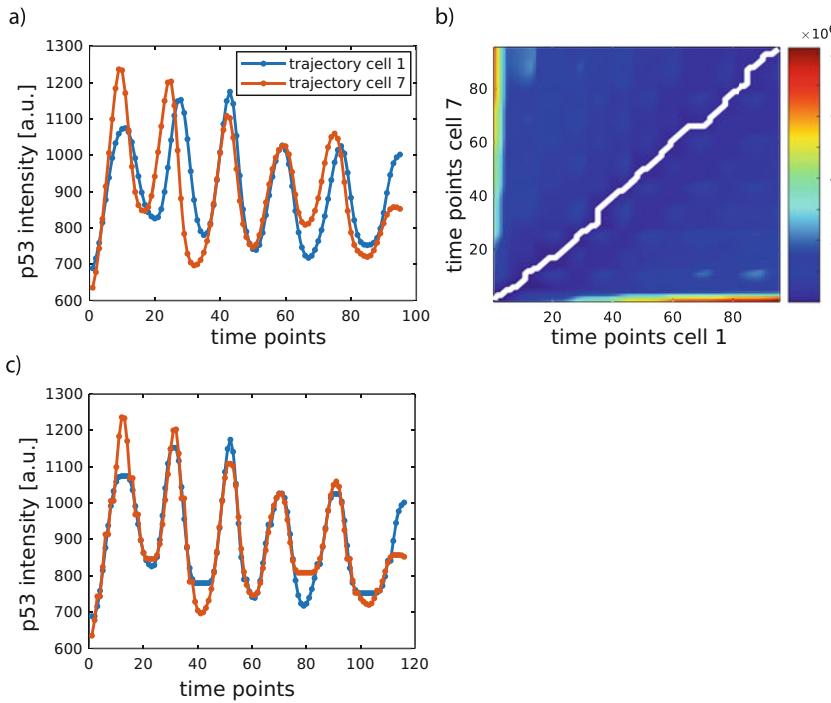


Fig. 5 Determining dissimilarities among trajectories. (a) The blue line with dots represents the trajectory of a cell with index 1, and in orange the trajectory of cell 7. (b) The color denotes the Euclidean distance between the trajectories of cell 1 and 7 at a specific time point. The white line shows the optimal warping path with the minimal distance between the two trajectories. (c) Trajectories of cell 1 and cell 7 after applying dynamic time warping. (The figure and legend are reproduced from Konrath [22] with permission from Logos Verlag Berlin GmbH)

to find the optimal warping path, that is, the minimal distance between trajectories (Fig. 5b, white line). Based on the optimal warping path, the data points of both trajectories can be aligned (Fig. 5c). The minimal distance is calculated for all possible combinations of pairs of trajectories, and the resulting dissimilarity matrix is subjected to clustering. We choose the agglomerative hierarchical clustering algorithm developed by Ward [30] as it was shown to work well for single cell dynamics [27]. The next important step is to determine the number of clusters which is not known a priori but can be estimated by different methods. We used the Calinski-Harabasz index [31] that identifies the optimal number of clusters based on the within-cluster variance, between-cluster variance, and difference between number of observations in the data set and the tested number of clusters. We computed the Calinski-Harabasz index for 1 to 40 clusters and identified the highest index in our data set for 24 clusters. We also tested the silhouette value [32] and the jump method [33]. While the silhouette value indicated an optimal cluster number of two, the jump method suggested 38 clusters. Since two clusters exhibit a high within-cluster variance and

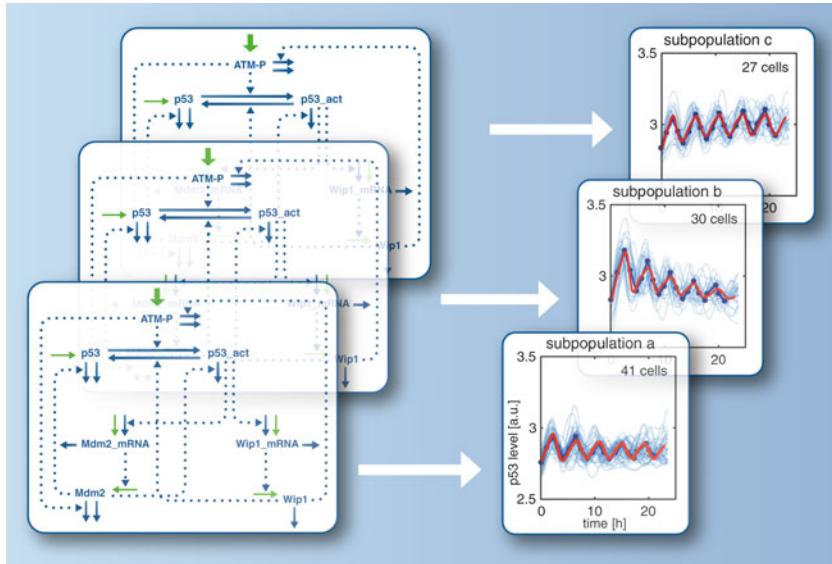


Fig. 6 Example of three models from the model pool and the corresponding fitted subpopulation-specific dynamics. The individual models differ in parameter values of the subpopulation-specific processes (green arrows). The parameters of the remaining processes (blue arrows) are identical in all models of the model pool. The simulation result of fitting the model pool simultaneously to the peak-based mean of the subpopulations (blue lines and dots) is given by the red line. For the model fit, a weighted peak-based mean is calculated based on the number of cells that were assigned to a subpopulation (top right corner of plots). (The plots showing the experimental and simulation data are reproduced from Konrath [22] with permission from Logos Verlag Berlin GmbH)

38 clusters would result in a high number of subpopulation-specific models and concomitantly high computational costs, we selected the Calinski-Harabasz index for our analysis.

Within the 24 identified clusters, we found 14 clusters with less than 10 assigned trajectories. To reduce computational costs, we neglected those 14 clusters for our subpopulation-based modeling approach. As the averaged dynamics of clusters are weighted, including those clusters with a small number of assigned trajectories would only have a minor effect on parameter inference.

In a final step of the clustering approach, an appropriate representation of averaged dynamics of clusters has to be defined. Although clusters contain trajectories with similar dynamics, computing the mean or median of trajectories for each cluster would lead to a loss of the characteristic p53 dynamics. We therefore defined a peak-based mean. To this end, we identified the peaks within each trajectory of a cluster and calculated the mean of the absolute value of peaks and the mean of the timing of peaks. Moreover, we identified the time points and absolute values between peaks of a trajectory and again calculated the mean of those values for all trajectories of a cluster. This way, pulsatile p53 accumulations are preserved in the averaged dynamics of clusters (Fig. 6, dark blue dots and line).

3.2 Subpopulation-Specific Modeling

The aim of the subpopulation-based modeling approach is to capture the heterogeneous response of single cells to a certain stimulus using a deterministic model. By clustering the single cell trajectories into subpopulations and averaging the dynamics, the variability in the response is represented by differences in the averaged dynamics. To reproduce the different dynamics, a pool of models is developed in which each model is specific for a subpopulation. The model specificity is thereby based on individual parameter values that can differ among the subpopulation-specific models, while the network structure is identical in all models. Based on the fact that the variability of protein and mRNA levels among individual cells can explain cellular heterogeneity in terms of dynamics and cellular responses [28, 34, 35], one can define processes within the chosen model as either subpopulation-specific or shared among subpopulations. Consequently, we assumed that synthesis rates of proteins and mRNAs within the p53 network are susceptible to noise, while, for instance, phosphorylation rates are not affected by noise and are therefore similar among models of the model pool. We implemented for the defined subpopulation-specific processes a fold change parameter that can take different values for each subpopulation (Fig. 6, green arrows). In contrast, the unspecific processes share the same parameter value among all models of the model pool. To infer the shared and subpopulation-specific parameter values based on the averaged subpopulation dynamics, we used the open-source MATLAB toolbox Data2Dynamics [36, 37]. All models of the model pool were fitted simultaneously to the calibration data and resulted in a good representation of subpopulation-specific dynamics (Fig. 6, red lines). For the model fit, we weighted the peak-based mean of each subpopulation based on the number of trajectories that were assigned to a subpopulation. This way, the proportion of observed dynamics is incorporated in the inferred parameter values and allows a good representation of the population by the fitted model (Fig. 6). The simulations of the best model fit show that the parameters we defined as subpopulation-specific are sufficient to reproduce the heterogeneous p53 response. However, it is not clear if all of those parameters are indeed required to reproduce the data. Hence, one can apply L1 regularization [38] and identify subpopulation-specific parameters that can be defined as unspecific and still allow a good representation of the subpopulation dynamics. Consequently, the analysis reveals essential subpopulation-specific parameters with a major impact on the heterogeneous dynamics and thereby allows to refine the parameters that were defined as subpopulation-specific. In this way, we identified 22 out of 60 subpopulation-specific parameters that are not required to be subpopulation-specific [4, 22].

With the described approaches, we established a model that reproduces the heterogeneous dynamics of p53 in the cell population and can thus be used to link the altered dynamics of p53 observed upon perturbation of the NF-κB pathway to individual parameters.

4 Identifying Points of Interaction Between Two Signaling Pathways

For model-based predictions of molecular processes that mark the interaction points between two pathways, two different strategies can be followed. Both strategies share the common aim to derive interaction points by linking perturbations of a parameter to changes in dynamics. Parameters and their corresponding molecular processes are identified as potential interaction point if a perturbation of the parameter allows to reflect the experimentally observed altered dynamics. The main difference between the two strategies is the way one (i) perturbs the parameters and (ii) quantifies the difference between simulated and experimentally observed changes in dynamics.

For the analysis of the p53/NF-κB crosstalk, we combined both strategies. First, we performed a sensitivity analysis on all model parameters to assess the impact of individual parameters and parameter combinations on p53 dynamics. To evaluate the difference between simulation and experimental observation, we compared in a qualitative manner the change in defined features of p53 dynamics. In contrast, in our second strategy, we evaluated parameters based on their capability to quantitatively reproduce the experimentally observed change by employing parameter inference. This way, the fit quality that is a quantitative measure for the difference between simulation and training data can be used to rank parameters based on their capability to reproduce the altered dynamics.

4.1 Sensitivity Analysis-Based Prediction

Sensitivity analysis is a method to quantify the impact of parameter perturbations on a given measure. Here, we selected four features of pulsatile p53 dynamics as readout for the sensitivity analysis. Specifically, we chose the time points at which peak maxima and minima are observed, the time difference between two maxima (inter-peak interval), as well as the dampening between peaks, that is, the fold change of absolute values of the first and subsequent peaks. Of note, there are various features defining the shape of certain dynamics. We decided to use the abovementioned four features for our analysis since all four features were significantly changed upon inhibition of IKK2 using three different, structurally independent, pharmacological IKK2 inhibitors [4]. These experimentally observed changes in features are then used to determine if a feature is positively or negatively changed upon the applied

perturbation. In our study, we found that IKK2 inhibition leads to an increase in all four features [4]. Consequently, we sought to identify parameter perturbations that cause an increase in all four features of the simulated p53 dynamics. To cover both, a positive as well as a negative impact of IKK2 inhibition on a process, we also considered parameter perturbations causing a consistent decrease in all four features. To systematically assess the impact of parameters on the four features, we perturbed all model parameters by +1% of their nominal value and checked if features were consistently changed. Note, for the subpopulation-specific parameters, we perturbed a fold change parameter that is shared among models and multiplied to the individual subpopulation-specific parameters. This way, subpopulation-specific parameters are simultaneously perturbed by the same extent. In our case, none of the tested parameter perturbations reproduced the experimentally observed changes in features. This led us to analyze perturbations of all possible combination of parameter pairs. Strikingly, we found 13 out of 462 possible combinations that reproduced the consistent change in features and are therefore considered as putative interaction points. Hence, this approach allowed the identification of potential targets of the crosstalk and strongly indicated the existence of more than one interaction point between the NF-κB and p53 pathway.

Since the sensitivity analysis-based approach is based on changes in features that were experimentally induced by all three tested IKK2 inhibitors, the results of the sensitivity analysis are independent of inhibitor-specific side effects. However, this approach is not suitable to systematically evaluate different perturbation strengths, especially if combinations of parameter perturbations need to be analyzed. Moreover, the quantitative change in dynamics is not taken into account. Thus, we followed a second strategy in which we fitted parameters to the altered dynamics of p53. This way, the parameter perturbation strengths are derived from the experimental data and allow to capture additive and compensatory effects for parameter combinations. In combination with incorporating quantitative changes in p53 dynamics, the second strategy can provide a more refined identification of potential interaction points.

4.2 Parameter Inference-Based Prediction

The concept of the parameter inference-based approach is to fit individual parameters or parameter combinations to the altered dynamics that are observed upon perturbation of the interacting pathway (perturbation data), while all remaining parameters are fixed to their nominal value. Note that the nominal parameter values are gained by fitting the subpopulation-specific models to the calibration data (Subheading 3.2). The parameters and parameter combinations can then be evaluated based on their capability to reproduce the observed alterations. To be able to fit the

subpopulation-specific models to the perturbation data, single cell trajectories of the perturbation data need to be processed and assigned to the subpopulations of the calibration data.

For the p53/NF- κ B crosstalk analysis, we processed the trajectories of the perturbation data set capturing the altered p53 dynamics upon IKK2 inhibition similar to the calibration data set (Subheading 3.1). To ensure that the effect of IKK2 inhibition on p53 dynamics is represented on the level of subpopulations, we assigned the single cell trajectories of the perturbation data set to the subpopulations of the calibration data set. To this end, we defined a list of experimentally observed changes upon inhibition of IKK2 in certain features of p53 dynamics. The list contains qualitative changes such as the observed delayed timing of p53 peaks as well as a quantitative change capturing the difference in the absolute values of the first p53 peak [4]. We used those observed alterations to evaluate the change in the defined features between a single cell trajectory of the perturbation data and the peak-based mean of a subpopulation. Comparing the changes in features between a trajectory and each subpopulation provides a ranking for the assignment of a given trajectory to a subpopulation. After assigning the trajectories to the subpopulations, the peak-based mean of trajectories is calculated in a subpopulation-specific manner as described before (Subheading 3.1). The pool of models can then be fitted to the altered averaged dynamics of the subpopulations. As the results of the sensitivity analysis indicated more than one interaction point, we fitted all possible pairs of parameters to the perturbation data. Based on our assumption that IKK2 inhibition affects the same process with the same magnitude in each subpopulation, we excluded individual subpopulation-specific fold change parameters from the fitting routine. Note that the fold change parameters are multiplied with a parameter that is shared among all models of the model pool (Subheading 3.2). By fitting this shared parameter, processes assumed to be susceptible to noise are still included in our fitting routine.

For parameter inference, the peak-based mean of the assigned trajectories of the perturbation data were weighted based on the number of assigned trajectories to each subpopulation as it was done before for fitting the model pool to the calibration data.

As none of the best-fitted parameter pairs allowed to reproduce the pulsatile dynamics of p53, we repeated the fitting routine with all possible combinations of parameter triplets resulting in combinations with improved fit quality. We ranked the parameter combinations based on their corresponding fit quality and considered the top ranked combinations as putative interaction points.

The advantage of the parameter inference-based approach is the additionally incorporated quantitative information that provides a more stringent evaluation of potential interaction points.

This is due to the requirement of parameter perturbations to reproduce the altered dynamics in a quantitative manner. However, the approach is computationally more costly. Moreover, in comparison to the sensitivity analysis-based approach, additional assumptions and analyses are required for the assignment of trajectories from the perturbation data to the subpopulations of the calibration data.

5 Experimental Validation of Model-Based Predictions

The two presented strategies can reveal single or multiple interaction points that allow reflecting the altered dynamics. To evaluate those predictions, orthogonal experimental data is necessary that was not used for the identification and is therefore eligible to challenge the model-based predictions. Following both strategies for our analyzed system, we found multiple interaction points that can explain the observed alterations in p53 dynamics. To test the parameter inference-based predictions, we used additional time-resolved single cell data in which we applied the IKK2 inhibitor at different time points post DNA damage induction. We tested the top 30 ranked parameter combinations by quantifying the discrepancy between simulation and mean of measured p53 levels. Note that we considered multiple parameter combinations instead of focusing on the best fitted parameter combination to include combinations with comparable fit qualities and to evaluate the performance of combinations in a wider range. To quantify the discrepancy, we calculated the weighted χ^2 value for each parameter combination and all single cell data sets. Among those 30 parameter combinations, 8 combinations could reflect the data with overall low χ^2 values and were therefore selected for further validation [4]. Notably, two out of the eight selected combinations correspond to one of the parameter combinations that was also identified by the sensitivity analysis-based approach.

With a second validation experiment, we sought to evaluate the remaining parameter combinations based on the influence of IKK2 inhibition on dynamics of other p53 network components. Specifically, we investigated the time courses of Mdm2 protein, Mdm2 mRNA, Wip1 protein, Wip1 mRNA, as well as phosphorylated checkpoint kinase 2 (Chk2), an indicator for ATM activity, using Western blot and qRT-PCR. Comparing those time courses with the corresponding simulations of the eight parameter combinations revealed four combinations with qualitatively deviating dynamics in one or more of the tested network components and were therefore sorted out. Consequently, the remaining four parameter combinations were considered as potential interaction points.

These four identified combinations basically represent two mechanisms that allow to reproduce the observed effects of IKK2 inhibition on the p53 response to DNA double-strand breaks.

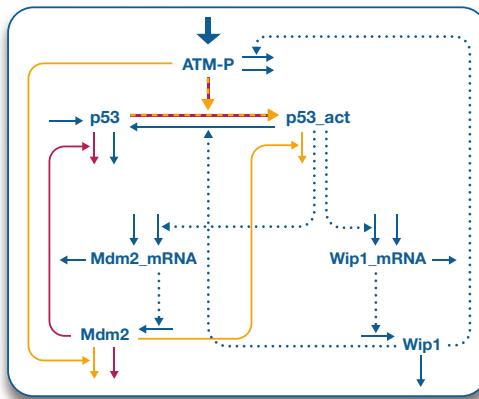


Fig. 7 Identified processes that allow to explain the observed changes in p53 dynamics upon IKK2 inhibition. The processes marked in yellow comprise the first predicted mechanism in which ATM-mediated degradation of Mdm2, activation of p53, and the degradation of activated p53 (*p53_act*) by Mdm2 is affected by the inhibition of IKK2. In the second predicted mechanism (processes marked in red), activation of p53 is affected as well in combination with an altered basal Mdm2 degradation rate and Mdm2-mediated degradation of p53

IKK2 inhibition causes in both mechanisms a reduced activation rate of p53. One mechanism indicates that IKK2 inhibition leads to an increased Mdm2-mediated degradation of inactive p53 as well as an elevated basal degradation of Mdm2. The second mechanism predicts a decreased Mdm2-mediated degradation of active p53 and a decreased ATM-mediated degradation of Mdm2 (Fig. 7).

6 Conclusion

Here, we present a framework that allows predicting molecular interactions between two signaling pathways based on alterations in dynamics of a selected pathway. As the alterations are caused by the perturbation of an interacting pathway, points of interaction between selected and interacting pathway are incorporated in the altered dynamics. To quantitatively monitor the alterations in p53 dynamics, we employed single cell time-lapse microscopy which provides characteristic p53 dynamics that would be masked in averaged dynamics of a cell population. To capture the heterogeneous p53 dynamics of single cells, we employed a subpopulation-based modeling approach in which a pool of models is fitted to subpopulation-specific p53 dynamics. The resulting calibrated model pool can then be used to link the observed alterations in dynamics to individual model parameters and the corresponding molecular processes. We propose two strategies to predict interaction points from altered dynamics, a sensitivity analysis-based

approach and a parameter inference-based approach. As our study showed, both strategies can provide valuable information on putative interaction points. Among the predicted interaction points, one interaction was predicted by both approaches, and both predicted the simultaneous perturbation of multiple interaction points between IKK2 and the p53 network. Importantly, various experimental studies revealed individual interactions at different points between IKK2 and the p53 signaling pathway and thereby strongly support our predictions. The individual experimental observations of multiple interaction points highlight the advantage of our approach to analyze crosstalk between two pathways in a comprehensive manner. Hence, our approach is eligible for complex crosstalk mechanisms and can now be applied to other pathways including but not restricted to pulsatile systems.

References

1. Won JK, Yang HW, Shin SY et al (2012) The crossregulation between ERK and PI3K signaling pathways determines the tumoricidal efficacy of MEK inhibitor. *J Mol Cell Biol* 4:153–163. <https://doi.org/10.1093/jmcb/mjs021>
2. Guo X, Wang XF (2009) Signaling cross-talk between TGF- β /BMP and other pathways. *Cell Res* 19:71–88. <https://doi.org/10.1038/cr.2008.302>
3. Oeckinghaus A, Hayden MS, Ghosh S (2011) Crosstalk in NF- κ B signaling pathways. *Nat Immunol* 12:695–708. <https://doi.org/10.1038/ni.2065>
4. Konrath F, Mittermeier A, Cristiano E et al (2020) A systematic approach to decipher crosstalk in the p53 signaling pathway using single cell dynamics. *PLoS Comput Biol* 16: e1007901. <https://doi.org/10.1371/journal.pcbi.1007901>
5. Vousden KH, Prives C (2009) Blinded by the light: the growing complexity of p53. *Cell* 137: 413–431. <https://doi.org/10.1016/j.cell.2009.04.037>
6. McCool KW, Miyamoto S (2012) DNA damage-dependent NF- κ B activation: NEMO turns nuclear signaling inside out. *Immunol Rev* 246:311–326. <https://doi.org/10.1111/j.1600-065X.2012.01101.x>
7. Hafner A, Bulyk ML, Jambhekar A, Lahav G (2019) The multiple mechanisms that regulate p53 activity and cell fate. *Nat Rev Mol Cell Biol* 20. <https://doi.org/10.1038/s41580-019-0110-x>
8. Haupt Y, Maya R, Kazaz A, Oren M (1997) Mdm2 promotes the rapid degradation of p53. *Nature* 387:296–299. <https://doi.org/10.1038/387296a0>
9. Ciccia A, Elledge SJ (2010) The DNA damage response: making it safe to play with knives. *Mol Cell* 40:179–204. <https://doi.org/10.1016/j.molcel.2010.09.019>
10. Stommel JM, Wahl GM (2004) Accelerated MDM2 auto-degradation induced by DNA-damage kinases is required for p53 activation. *EMBO J* 23:1547–1556. <https://doi.org/10.1038/sj.emboj.7600145>
11. Shieh SY, Ikeda M, Taya Y, Prives C (1997) DNA damage-induced phosphorylation of p53 alleviates inhibition by MDM2. *Cell* 91:325–334
12. Kruse J-P, Gu W (2009) Modes of p53 regulation. *Cell* 137:609–622. <https://doi.org/10.1016/j.cell.2009.04.050>
13. Ghosh S, May MJ, Kopp EB (1998) NF- κ B and Rel proteins: evolutionarily conserved mediators of immune responses. *Annu Rev Immunol* 16:225–260. <https://doi.org/10.1146/annurev.immunol.16.1.225>
14. Hinz M, Arslan SC, Scheidereit C (2012) It takes two to tango: I κ Bs, the multifunctional partners of NF- κ B. *Immunol Rev*. <https://doi.org/10.1111/j.1600-065X.2012.01102.x>
15. Oeckinghaus A, Ghosh S (2009) The NF- κ B family of transcription factors and its regulation. *Cold Spring Harb Perspect Biol* 1:a000034. <https://doi.org/10.1101/cshperspect.a000034>
16. Huang TT, Wuerzberger-Davis SM, Seufzer BJ et al (2000) NF- κ B activation by camptothecin. A linkage between nuclear DNA damage and cytoplasmic signaling events. *J*

- Biol Chem 275:9501–9509. <https://doi.org/10.1074/jbc.275.13.9501>
17. Lahav G, Rosenfeld N, Sigal A et al (2004) Dynamics of the p53-Mdm2 feedback loop in individual cells. Nat Genet 36:147–150. <https://doi.org/10.1038/ng1293>
 18. Geva-Zatorsky N, Rosenfeld N, Itzkovitz S et al (2006) Oscillations and variability in the p53 system. Mol Syst Biol 2(2006):0033. <https://doi.org/10.1038/msb4100068>
 19. Alon U (2006) An introduction to systems biology: design principles of biological circuits. CRC Press, Boca Raton
 20. Baum K, Schuchhardt J, Wolf J, Busse D (2019) Of gene expression and cell division time: a mathematical framework for advanced differential gene expression and data analysis. Cell Syst 9:569–579.e7. <https://doi.org/10.1016/j.cels.2019.07.009>
 21. Schwahnässer B, Wolf J, Selbach M, Busse D (2013) Synthesis and degradation jointly determine the responsiveness of the cellular proteome. BioEssays 35:597–601. <https://doi.org/10.1002/bies.201300017>
 22. Konrath F (2020) Mathematical modeling of NF-κB and p53 signaling in the DNA damage response. Logos Verlag Berlin GmbH, Berlin
 23. Batchelor E, Mock CS, Bhan I et al (2008) Recurrent initiation: a mechanism for triggering p53 pulses in response to DNA damage. Mol Cell 30:277–289. <https://doi.org/10.1016/j.molcel.2008.03.016>
 24. Batchelor E, Loewer A, Mock C, Lahav G (2011) Stimulus-dependent dynamics of p53 in single cells. Mol Syst Biol 7:488. <https://doi.org/10.1038/msb.2011.20>
 25. Fröhlich F, Reiser A, Fink L et al (2018) Multi-experiment nonlinear mixed effect modeling of single-cell translation kinetics after transfection. npj Syst Biol Appl 4:42. <https://doi.org/10.1038/s41540-018-0079-7>
 26. Llamosi A, Gonzalez-Vargas AM, Versari C et al (2016) What population reveals about individual cell identity: single-cell parameter estimation of models of gene expression in yeast. PLoS Comput Biol 12:e1004706. <https://doi.org/10.1371/journal.pcbi.1004706>
 27. Strasen J, Sarma U, Jentsch M et al (2018) Cell-specific responses to the cytokine TGFβ are determined by variability in protein levels. Mol Syst Biol 14:e7733. <https://doi.org/10.15252/msb.20177733>
 28. Loewer A, Lahav G (2011) We are all individuals: causes and consequences of non-genetic heterogeneity in mammalian cells. Curr Opin Genet Dev 21:753–758. <https://doi.org/10.1016/j.gde.2011.09.010>
 29. Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process. <https://doi.org/10.1109/TASSP.1978.1163055>
 30. Ward JH (1963) Hierarchical grouping to optimize an objective function. J Am Stat Assoc 58: 236–244. <https://doi.org/10.1080/01621459.1963.10500845>
 31. Calinski T, Harabasz J (1974) A dendrite method for cluster analysis. Commun Stat - Theory Methods 3:1–27. <https://doi.org/10.1080/03610927408827101>
 32. Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53–65. [https://doi.org/10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7)
 33. Sugar CA, James GM (2003) Finding the number of clusters in a dataset. J Am Stat Assoc 98: 750–763. <https://doi.org/10.1198/016214503000000666>
 34. Spencer SL, Gaudet S, Albeck JG et al (2009) Non-genetic origins of cell-to-cell variability in TRAIL-induced apoptosis. Nature 459:428–432. <https://doi.org/10.1038/nature08012>
 35. Bertaux F, Stoma S, Drasdo D, Batt G (2014) Modeling dynamics of cell-to-cell variability in TRAIL-induced apoptosis explains fractional killing and predicts reversible resistance. PLoS Comput Biol 10. <https://doi.org/10.1371/journal.pcbi.1003893>
 36. Rau A, Schilling M, Bachmann J et al (2013) Lessons learned from quantitative dynamical modeling in systems biology. PLoS One 8: e74335. <https://doi.org/10.1371/journal.pone.0074335>
 37. Rau A, Steiert B, Schelker M et al (2015) Data2Dynamics: a modeling environment tailored to parameter estimation in dynamical systems. Bioinformatics 31:1718–1726. <https://doi.org/10.1093/bioinformatics/btv405>
 38. Steiert B, Timmer J, Kreutz C (2016) L1 regularization facilitates detection of cell type-specific parameters in dynamical systems. Bioinformatics 32:i718–i726. <https://doi.org/10.1093/bioinformatics/btw461>



Chapter 13

Live-Cell Sender-Receiver Co-cultures for Quantitative Measurement of Paracrine Signaling Dynamics, Gene Expression, and Drug Response

Michael Pargett, Abhineet R. Ram, Vaibhav Murthy,
and Alexander E. Davies

Abstract

Paracrine signaling is a fundamental process regulating tissue development, repair, and pathogenesis of diseases such as cancer. Herein we describe a method for quantitatively measuring paracrine signaling dynamics, and resultant gene expression changes, in living cells using genetically encoded signaling reporters and fluorescently tagged gene loci. We discuss considerations for selecting paracrine “sender-receiver” cell pairs, appropriate reporters, the use of this system to ask diverse experimental questions and screen drugs blocking intracellular communication, data collection, and the use of computational approaches to model and interpret these experiments.

Key words Single cell, Intercellular communication, Live-cell microscopy, Cancer, ERK, MAPK, Drug screening

1 Introduction

Intercellular communication is a core process regulating normal development and pathological states such as cancer [1–5]. However, understanding the precise spatial and temporal aspects of intercellular signaling is challenging. *In vivo*, one must be able to track cell signaling responses over time and geographically within a tissue. Technical limitations make these studies challenging to perform, and the *in vivo* setting limits perturbations that can be made to quantitatively define signaling dynamics, gene expression, and cell fate. Additionally, *in vivo* signaling represents a composite effect from multiple signals, arising from multiple cell types that converge on individual “receiver” cells. Thus, an *in vitro* system that uses a reductionist approach to study intercellular communication, in a simple and controllable environment, represents an attractive model to perform “sender” to “receiver” cell interaction

studies. Herein we describe such an approach that allows detailed exploration of sender-receiver relationships and has been used successfully to study this relationship between malignant and adjacent nonmalignant cells [5]. Implementation of this approach allows for detailed spatial and temporal profiling of signaling, gene expression, and cell fate in living cells with single cell resolution and yields quantitative information that can inform computational model development. Importantly, this system is adaptable to multiple signaling pathways and cell types. Therefore, it can be used to answer questions focused on tissue development, wound healing, and other relevant biological problems coordinated by intercellular signaling.

During development and homeostasis, intracellular signaling is known to spatially and temporally orchestrate cell fate changes or behaviors resulting in the generation or maintenance of organized tissue systems [2, 3, 6–8]. One such example occurs during postnatal mammary gland development. At puberty, rising systemic estrogen levels induce mammary luminal epithelial cells to produce and secrete the epidermal growth factor receptor (EGFR) ligands, including amphiregulin (AREG), in a paracrine manner, creating a sender-receiver cell relationship between epithelial and surrounding myoepithelial and stromal cells, respectively [9–11]. Gradients of AREG are received by proximal myoepithelial cells and fibroblasts resulting in induction of gene expression programs that regulate mammary gland invasion and morphogenesis [11]. Establishment of such a sender-receiver relationship between mammary epithelial cells and the surrounding stromal cells is an essential step in functional gland development. However, this process is hijacked in some tumors of the breast. Breast cancer cells spontaneously gain the ability to secrete AREG, independent of estrogen signaling, leading to the establishment of a dysregulated sender-receiver system between individual tumor cells and between tumor and stromal cells [4, 12, 13]. Secreted AREG is bound by receiver cell EGFR, resulting in stimulation of the downstream MAPK terminal signaling effector, ERK, stimulating tumor cell proliferation, and reprogramming of the surrounding nonmalignant cell types [4, 13]. Consequently, in this specific case, the basic AREG sender-receiver cell relationship can drive normal morphogenesis, or when dysregulated, aberrant tissue function-organization relationships.

Adding complexity to the sender-receiver system, recent advances in light microscopy and reporter technologies have revealed that signaling is dynamic and varies both spatially and temporally at the single cell level. Implementation of genetically encoded reporters has allowed such detailed study using several methodologies and targeting multiple pathways. The majority of these approaches utilize Förster resonance energy transfer (FRET) or kinase translocation-based reporters. For extensive review of

these reporters, see [15, 16]; FRET reporters are composed of a donor and acceptor fluorophore (typically cyan fluorescent protein (CFP) and yellow fluorescent protein (YFP) variants, respectively) separated by a linker sequence, docking, phosphorylation site for a kinase of interest (e.g., ERK), and a phospho-amino acid binding domain. Following phosphorylation by the kinase, the phosphorylated residue and binding domain interact, reorienting the donor-acceptor pair into close proximity resulting in FRET when the CFP is excited. Kinase activity can be measured by the ratio of CFP to YFP in fluorescent microscopy or by fluorescence lifetime imaging microscopy (FLIM). Kinase translocation reporters, or KTRs, contain a tandem nuclear import and export sequence that is also a kinase substrate. Phosphorylation of the reporter by a kinase suppresses shuttling from nucleus to cytoplasm. As a result, kinase activity can be measured as the ratio of cytosolic (C) to nuclear (N) fluorescence. KTRs have been designed for a number of signaling pathways including ERK, Akt, JNK, and others [16]. For a more extensive discussion of basic translocation reporter use and analysis, we suggest reading Pargett et al. (2017), Pargett and Albeck (2018), and Kudo et al. (2018) [17–19]. One benefit of KTRs is the utilization a single fluorophore resulting in the ability to use and discriminate different reporters (e.g., ERK and Akt reporters), in a single cell, based on discrete emission wavelengths (Fig. 1). Additionally, because the tandem import/export and kinase substrate sequences can be fused to different fluorophores, one can use the same reporter to measure kinase activity and to discriminate different cell types, for example, an ERK reporter fused to a different fluorophore in each different cell type. As described in this protocol, this feature allows different cell types to carry the same functional reporter with spectrally distinct fluorophores (e.g., mVenus or mTurquoise2) to easily distinguish the signaling dynamics of sender and receiver cells with single cell resolution (Fig. 2).

With the advent of CRISPR-technologies, we are now able to tag genes of interest with fluorescent proteins at their endogenous loci. This allows one to track both signaling dynamics using genetically encoded reporters and the expression of downstream target genes, such as ERK target gene Fra-1, in living cells over time. Utilization of this approach allows determination of both signaling and resultant gene expression under varying conditions (Fig. 2) [20]. In this protocol, we describe the use of this approach to simultaneously measure the effects of sender cell growth factor secretion on receiver cell signaling and gene expression.

Making full use of reporters requires an analysis pipeline able to track single cells over time and measure the signal intensity of multiple reporters and/or tagged genes with high fidelity. With the expansion of live-cell markers, several computational methods have been developed to fill this need. Such software is able to

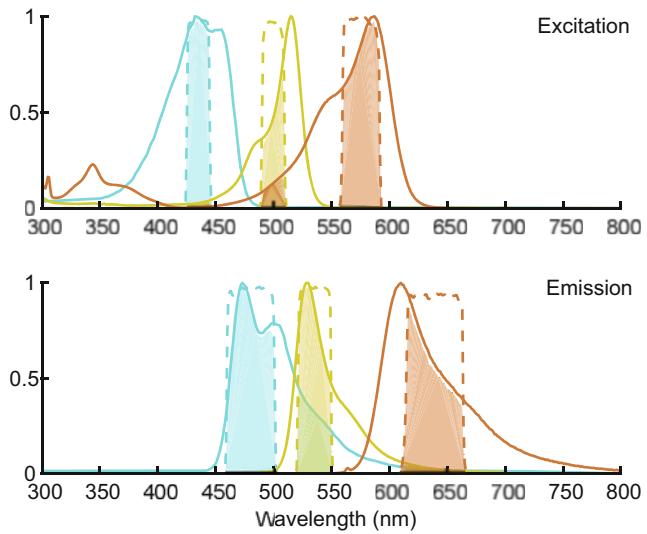


Fig. 1 Spectral comparison example for multiple fluorescent proteins. Shown as solid lines are the excitation (upper) and emission (lower) spectra for mTurquoise2 (cyan, left), YPet (yellow, center), and mCherry (red, right). Dashed lines show filter transmission bands, for example, CFP, YFP, and RFP filter sets (colored and ordered spectrally). The relative intensity of light passing from a fluorophore through a filter can be assessed by the shaded overlap areas. The net channel intensity through a filter set is proportional to the product of the shaded areas for excitation and emission of that color. Unwanted cross-talk between channels can be evaluated by the shaded areas where a fluorophore spectrum overlaps a differently colored filter. For example, the YFP emission filter overlaps the mTurquoise2 spectrum significantly (green shading, center of lower graph). However, in the excitation graph, these spectra overlap negligibly (center of upper graph). The product of these two cross-talk areas is very small compared to that for the YFP channel, resulting in well-separated channels

automatically analyze time-lapse images, identifying and tracking individual cells over time. Software implementations are increasingly available and are now largely approachable without writing custom code. However, while some of these software provide visualization and plotting tools, handling and analysis of the time series data are often performed with custom code via scripting software (e.g., MATLAB, Python, and R).

In the following protocol, we provide an approach based on experiments reported in Davies et al. (2020) that can be adapted to explore intracellular signaling biology generally, or used for other purposes, such as screening for drugs that block tumor-microenvironment signaling interactions [5]. This system is also highly adaptable to many types of sender-receiver relationships such as tumor-immune signaling, three-dimensional cultures, and tissue explants in a manner dependent upon available reporters, the appropriate microscope, environmental control equipment, and a suitable computational analysis pipeline.

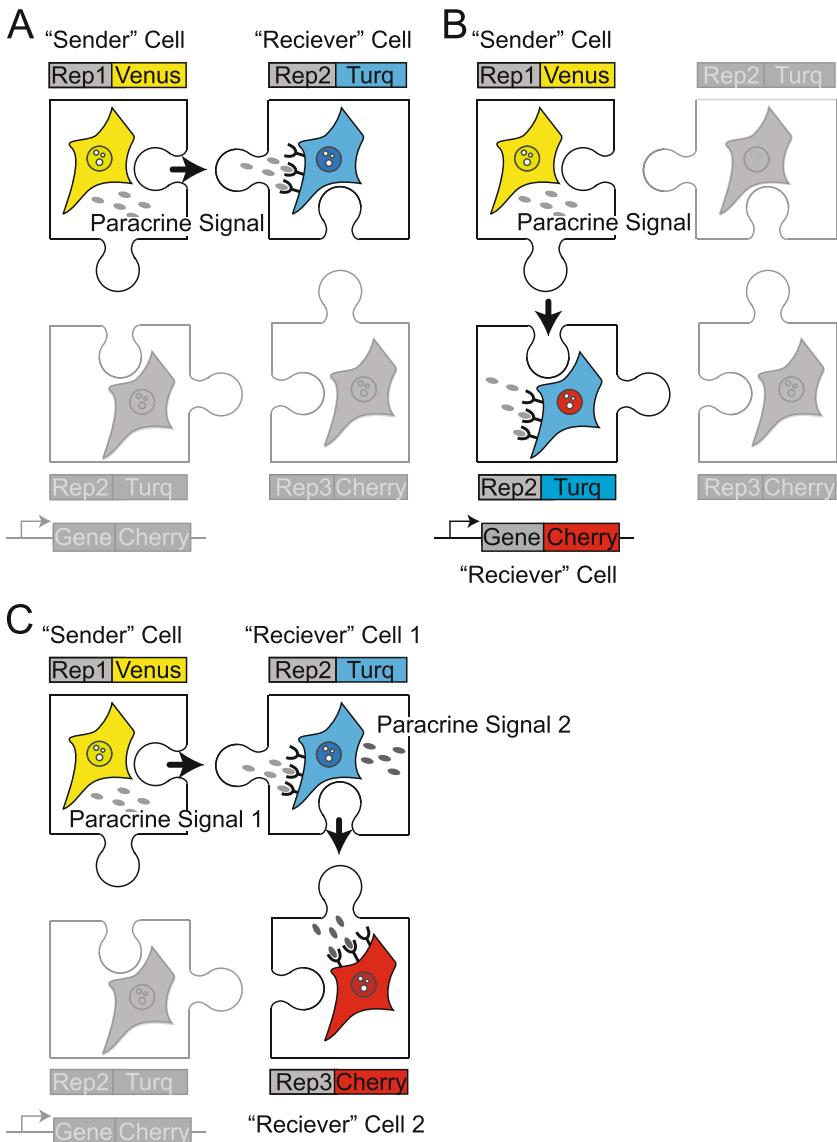


Fig. 2 Modular construction of sender-receiver cell cultures. Sender-receiver pairs, analogous puzzle pieces, can be combined in various ways and are limited only by available spectra, suitable cell pairs, and analysis capabilities. **(a)** In the simplest form, a sender-receiver pair is constructed using the same reporter (“Rep1” and “Rep2,” e.g., ERK KTR) fused to different fluorescent proteins (e.g., mVenus (yellow) or mTurquoise2 (blue) allowing visual and computational deconvolution of the data. **(b)** Using this same approach, receiver cells can be generated that co-express a reporter construct (blue) and another reporter or CRISPR-based fluorescent protein-gene fusion, represented here as a mCherry fusion (red). **(c)** A more complex assay can be created using cell lines multiple spectrally distinct sender-receiver combinations (mVenus, mTurquoise2, and mCherry fused reporter constructs; yellow, blue, and red, (“Rep1-3”), respectively), where, for example, the sender cell engages receiver cell 1 and receiver cell 1 acts as a sender for receiver cell 2

2 Materials

2.1 Cell Lines

1. 293FT cells for packaging of viral vectors (Thermo Fisher, #R70007).
2. Sender and receiver cells of interest (*see Note 1* Choice of Cell Lines). This approach applies to a wide variety of cell lines and primary cell types. For the example outlined here, we utilize the isogenic HMT-3522 malignant progression series, pairing the nonmalignant S1 cells with the malignant T4-2 cells. HMT-3522 cell lines are available commercially through Millipore Sigma (98102210, 98102212, respectively); however, the cells described here were sourced from the original lineages and only available upon request [21–23].

2.2 Cell Culture Reagents

Reagents 1–9 are required for replicating the experiments in Davies et al. (2020), if other cell types are to be utilized, adjust cell culture reagents as required.

1. Prolactin from sheep pituitary gland (Millipore Sigma L6520).
2. Insulin (Millipore Sigma I6634).
3. Sodium selenite (Millipore Sigma S5261).
4. Beta-estradiol (Millipore Sigma E8875).
5. Hydrocortisone (Millipore Sigma H0888).
6. Transferrin (Millipore Sigma T8158).
7. Recombinant human epidermal growth factor (Peprotech AF-100-15).
8. DMEM/F12 containing bicarbonate buffer (Gibco 11320082).
9. DMEM/F12 phenol-free media for imaging (Gibco 11039047).
10. DMEM high glucose containing HEPES (Gibco 12430112).
11. OptiMEM (Gibco #31985088).
12. Fetal bovine serum (Gemini Biosciences 100-106).
13. 0.25% trypsin solution (Gibco 25200056).
14. Antibiotic selection agents corresponding with reporter plasmid selection marker expression (e.g., puromycin).

2.3 CRISPR Tagging and Validation Reagents

1. Cas9/gRNA delivery plasmid (e.g., px330 Addgene #42230).
2. Homology template plasmid backbone (e.g., pENTR, pAAV, TOPO).
3. Custom gRNA PCR primers.
4. Custom homology arm PCR primers.
5. Fluorescent protein template sequence.

6. High fidelity PCR polymerase, dNTPs, and associated standard PCR reagents.
7. Thermocycler.
8. Restriction enzymes (*BbsI*).
9. T4 polynucleotide kinase.
10. Gibson Assembly Master Mix.
11. PCR or gel purification kits.
12. Competent *E. coli*.
13. Plasmid isolation kit.
14. Transfection reagents (see Subheading 2.5 for reference).
15. Standard immunofluorescence reagents, including paraformaldehyde, permeabilization buffer (e.g., % Triton X-100 diluted in PBS), blocking buffer (e.g., 2% BSA diluted in PBS containing 0.1% Triton X-100), and wash buffer (e.g., 0.1% Triton X-100 in PBS).
16. Primary antibody specific for CRISPR tagged gene(s) of interest.

Secondary antibody with a distinct emission spectrum as compared to reporters and tagged genes. We prefer Alexa Fluor (Life Technologies) tagged secondaries as they are available in a wide range of emission wavelengths.

2.4 Genetically Encoded Reporter Constructs

1. Multiple reporter constructs are available via Addgene corresponding with the published literature or by request from individual investigators. In the protocol discussed here, the original ERK-KTR reporter from Regot et al. (2014) was cloned into the pLJM1-puromycin (Addgene #19319) lentiviral vector followed by the mTurquoise2 or mVenus fluorescent protein coding sequence [16]. For a discussion on selection of reporters, see Note 2 Choice of Genetically Encoded Reporters.
2. If genetically encoded nuclear tracking markers are required, we recommend using histone H2B tagged with mCerulean, mRuby, or iRFP with the specific choice dependent upon the spectrum of other reporter(s) to be used (Addgene #90234, #90236, #90237, respectively). If a genetically encoded marker is not required, we recommend the cell permeable nuclear dye Hoechst 33342.

2.5 Viral Production Reagents

1. Sterile tissue culture dishes (6-well plate suggested).
2. Sterile 15 mL conical tubes.
3. Polybrene (Millipore Sigma TR-1003).
4. Poly-D-Lysine (Millipore Sigma P6407).

5. Fugene HD (Promega E2311) or JetPRIME (PolyPlus Transfection 114-01).
6. 5 mL syringe.
7. 0.45 uM sterile filter.
8. Viral packaging vectors – third-generation lentiviral compatible (e.g., psPAX2 Addgene #12260, and pMD2.G Addgene #12259).

2.6 Live Cell and

Fixed Imaging Materials and Reagents

1. Laminin-111 (Thermo Fisher 23017015).
2. Laminin working stock (prepared immediately before use): Laminin-111 50 µg/mL with 20 mM sodium acetate pH 4 and 1 mM CaCl₂.
3. Rat tail collagen I coating solution (Gibco A1048301).
4. Rat tail collagen I working stock: 50 µg/mL rat tail collagen dissolved in 0.02N acetic acid.
5. 10× Phosphate Buffered Saline.
6. Bovine Serum Albumin (BSA).
7. Triton X-100.
8. Multi-well glass bottom imaging plates (e.g., CellVis P96-0-N).
9. Single and multichannel pipettors.
10. Reagent reservoirs.
11. Hoechst 33342 (Thermo Fisher 62249) or genetically encoded marker (e.g., H2B mCerulean).
12. Recombinant human epidermal growth factor if using ERK-KTR, or another growth factor capable of stimulating the pathway of interest (Peprotech AF-100-15).
13. MEK inhibitor (e.g., PD0325901, Selleck Chemicals S1036) if using ERK-KTR, or another inhibitor capable of suppressing the pathway of interest.

2.7 Live Cell Microscopy Equipment

1. Stage top environmental chamber capable of maintaining 5% CO₂ and 37 °C for >12–24 h, such as OkoLabs Bold Line or Tokai Hit STX systems used in our laboratory.
2. Automated microscope capable of taking serial xy images at predetermined time points for 12–24 h. We utilize a Nikon Ti2E with automated stage and NIS-Elements software.
3. Fluorescence excitation source (e.g., Lumencor Sola II), filter sets corresponding with the chosen fluorescent reporters, and an image capture device (e.g., Photometric Prime 95B).

2.8 Image Processing and Modeling

1. Computer with at least a 2 GHz CPU and 8 GB RAM (though many software can operate on lesser systems). Depending on software, a dedicated GPU may be required, often with at least 2 GB video RAM.
2. MATLAB software and Image Processing Toolbox or comparable (see Note 3 Image Processing Software).

3 Methods

3.1 CRISPR Gene Tagging

CRISPR-Cas9 systems are widely used to generate gene knock-in cell lines and model systems [20, 24, 25]. The revolutionary method enables endogenous labeling of proteins of interest (POI) with fluorescent tags, allowing for careful investigation of expression, localization, and dynamics. This section will describe an effective protocol to generate knock-in cell lines. There are several techniques for endogenous tagging that differ in type of Cas9 protein, number of guide RNA (gRNA), length of homology arms, and delivery of each component to the target cell line [24, 26–28]. This protocol uses a human codon optimized *Streptococcus pyogenes* Cas9 nuclease in a chimeric guide RNA expression cassette (e.g., px330). The homology repair template is cloned into any desired plasmid vector (pENTR, pAAV, TOPO) [29]; however, it can optionally be delivered as a PCR product. Because CRISPR gene tagging can be technically challenging, we recommend generating edited cell lines first, then freezing down stocks prior to adding genetically encoded reporters (Subheading 3.3). See Note 4 for considerations prior to gene knock-in studies.

1. Preparation of guide RNA and homology repair template plasmid constructs. Use online tools to search for 20 nucleotide gRNA sequences followed by PAM (NGG) motifs. The gRNA will target the Cas9 protein to the cleavage site; therefore, the sequence should be close to (within ~100 base pairs) or directly over STOP codon of the gene of interest. Most online tools also provide efficiency and off target effect predictions. Consider which is most important for the proposed study and chose the top three to five gRNA sequences for cloning. The Broad Institute or Benchling websites host useful gRNA design tools.
2. If the gRNA sequence does not start with a guanine, add a G to the beginning (5' end) of the sequence to facilitate U6 promoter transcription. Then design a complementary sequence to be used for annealing and restriction cloning. Add CACC base pairs to the 5' end of the forward sequence. Then add AAAC base pairs to the 5' end of the complementary sequence. This will make the ends compatible for *Bbs*I restriction cloning. An example of annealed primers is shown below.

Forward primer 5' **CACCGNN..NN** 3'

Reverse primer 3' CNN..NNCAAA 5'

3. Order and anneal each pair of oligos together. Digest the px330 plasmid with *BbsI*, and ligate it with each annealed gRNA. A detailed px330 cloning guide can be found on Addgene.com.
4. When designing the homology repair template, ensure the following: (1) introduce silent mutations in the guide RNA recognition sequence of the homology region so that the Cas9 protein does not continue its nuclease activity after homology-directed repair (HDR). (2) Insert a flexible linker between the POI and fluorescent protein; this is important for stability of the fusion protein [30]. We recommend a 3–5× repeating sequence of Gly-Ala [24] or Gly-Gly-Ser [31] for the linker domain. (3) Ensure there is a STOP codon at the end of the fluorescent protein to terminate translation of the fusion protein. (4) Each homology arm (HA) encompasses 1.4 kb-sized base pairs on each side of the target gene stop codon. An example is shown below.

Native gene locus:

ATG_{start}...GCGGC....

TAGAC_{1.4kb} TAG_{stop} CTGGA...TGACGA_{1.4kb}

Repair template:

GCGGC....TAGAC_{1.4kb} – Linker domain – Fluorescent Protein – STOP – CTGGA...TGACGA_{1.4kb}

5. Design PCR primers for a four-piece Gibson assembly of the left homology arm, fluorescent protein, right homology arm, and the desired plasmid backbone. The linker domain DNA sequence can be inserted directly into the overhanging region of the PCR primers. PCR the homology arms from the target cell line genomic DNA, and PCR the fluorescent protein from its template plasmid. Assemble the final plasmid and amplify. This cloned plasmid will be used as the homology repair template. The template can be optionally delivered as a PCR product instead by using a primer pair flanking each homology arm.
6. Transfect the homology repair template, and the Cas9/gRNA plasmid into your target cells using optimal transfection methods for each specific cell line.
7. Monitor cells after the transfection to ensure normal growth and proliferation.
8. Positive knock-in cells can be isolated through fluorescence-activated cell sorting (FACS) or limited dilution cloning. If the cell line is tolerant of single cell plating, conduct single cell FACS into a 96-well dish or perform limited dilution cloning.

If the POI is regulated by the cell cycle, consider synchronizing the cells into the stage with highest POI expression before cell sorting. *See Note 5.*

3.2 Validation of CRISPR Gene Editing

1. Validate expression of tagged proteins. Correct integration of the fluorescent protein can be checked via western blotting for the POI. Use standard protocols suited to the cell line. This validation step may require stimulation experiments (i.e., if the POI is not constitutively expressed) using the knock-in (tagged) cell line and negative control (non-tagged) cells. *See Note 6.*
2. Validate accurate genomic insertion of tags. Following standard protocols, extract genomic DNA from knock-in and non-tagged cell lines, and PCR the edited gene locus using primers flanking the homology region of the gene (both homology arms). Analyze PCR products for size by gel electrophoresis, and validate the fidelity of the insertion by sequencing. *See Note 7.*
3. Validate intracellular localization of tagged proteins. For the highest confidence that knock-in cell lines are good representations of the parental lines, cells can be fixed and stained for immunofluorescence (using standard protocols for the cell line, *see Note 8*). Compare the fluorescence of antibodies directed at the POI with that of the fluorescent protein marker (or from antibodies against that marker).

3.3 Establishing Reporter Cell Lines

The first step is to select appropriate sender-receiver cell lines based on pathways of interest. For example, in Davies et al. (2020) we utilized the HMT-3522 malignant progression series lines composed of nonmalignant breast epithelial S1 cells and their isogenic T4-2 malignant counterparts [5]. S1 and T4-2 cells form an excellent sender-receiver pair because T4-2 have acquired the ability to secrete the epidermal growth factor ligand, amphiregulin (AREG), whereas S1 cells do not. Additionally, S1 cells express the epidermal growth factor receptor (EGFR) allowing them to receive AREG paracrine signals from T4-2 cells. *See Note 1* for important considerations of cell line selection.

Kinase translocation reporters such as ERK-KTR are an advantageous tool for studying live, single-cell signaling and can be multiplexed with other biosensors for multiple kinase activity measurements from the same cell making them suitable for sender-receiver studies (Fig. 2) [16, 32]. An important consideration is choosing the ideal fluorescent protein that serves as the readout for the reporter. Several labs have taken advantage of different color fluorescent proteins for their experiments; these include mClover, mCherry, mVenus, and mTurquoise2 [2, 5, 33]. If multiplexing with other reporters, confirm spectral compatibility by analyzing

the excitation/emission spectra of the fluorescent proteins (Fig. 1). Additionally, ensure that the microscope fluorescence filter cubes have proper excitation wavelengths and emission filters that will allow for accurate measurement of the reporter(s). Regardless of the reporter chosen, careful selection is warranted before proceeding. This is because some receptor ligand interactions can stimulate multiple downstream pathways of interest, for example, EGFR is capable of stimulating ERK, AKT, STAT, and other signaling pathways for which reporters exist. Considering the pathway, or pathways, of interest is therefore critical depending upon the goals of a particular experiment.

A clear nuclear marker must be used for proper computational nuclear and cytoplasmic segmenting. A common method facilitating accurate nuclear segmentation is to stably integrate a fluorescent protein fused to histone 2B (H2B) [16]. This fluorescent protein must be a different color than the ERK-KTR reporter. A more convenient method is to stain cells with Hoechst before the live-cell imaging experiment. These compounds can be visualized with fluorescence microscopy, have low toxicity, and can be used in live cells for several days.

1. For further discussion on selecting of sender-receiver cells, *see Note 1*.
2. For further discussion on selecting of reporters, *see Note 2*.
3. Thaw and culture 293T cells for transfection in DMEM supplemented with 10% FBS until 70–80% confluent.
4. Coat 6-well cell culture dish with 1 mL Poly-D-Lysine.
5. Incubate at 37 °C for 20 min.
6. Seed 293T cells at a density of 500,000–750,000 cells per well. Resuspend cells thoroughly so that there are no cells attached together.
7. Incubate overnight at 37 °C and 5% CO₂.
8. Transfect 293FT cells. Transfection reagents may vary; efficient chemical transfection reagents include FuGENE® (Promega) and jetPRIME® (PolyPlus) transfection reagent. Transfection protocol depends on the reagent and follows manufacturer instructions. Each reaction should contain 1 µg of the lentiviral reporter plasmid of your choice, 1 µg of the viral packaging plasmid (psPAX2), and 100 ng of the viral envelope plasmid (pMD2.G).
9. After transfection, incubate cells for ~12 h at 37 °C and 5% CO₂.
10. Change media to fresh growth media and incubate for another 24 h. If desired, you can inspect transfection rates by using a fluorescence microscope. High transfection rates will leave many cells visibly expressing the reporter construct.

11. Begin to collect virus using appropriate techniques and personal protective equipment. After the previous 24-h incubation step, the media will begin to contain viral particles. Collect this media and replenish wells with fresh media. Store collected media at 4 °C.
12. If desired, this collection step can be repeated every 24 h for 3–4 days after the initial transfection step.
13. After the desired amount of virus is collected, sterile filter the virus using a 0.45 µM filter. Create 500 µL aliquots of virus containing media, and proceed to transduction step or store aliquots at –80 °C. Aliquots may be stored at 4 °C up to 2 weeks; however, this may result in decreased transduction efficiency.
14. Prepare target cell line to be transduced by seeding cells at a density of 100,000 cells per well in a 6-well tissue culture dish, and incubate overnight to allow cells to attach. Include one well that will not be infected with virus; this will serve as a control for future antibiotic selection. Cell culture media should be prepared as appropriate for the cell line chosen. If using the HMT-3522 cell lines, refer Weaver et al. (1997) and Briand et al. (1987) for a detailed description of media composition and cell handling [22, 23].
15. After 24 h, replace media with 1 mL fresh media containing polybrene at a concentration of 12 µg/mL. After virus addition, the final concentration of polybrene will be 8 µg/mL. Depending on the cell line, optimal concentration of polybrene may be within 1–10 µg/mL.
16. Add 500 µL of virus containing media dropwise to each well and gently swirl the plate.
17. Incubate for 24 h at 37 °C and 5% CO₂.
18. Replace virus containing media with fresh media, and allow cells to divide for another 24 h.
19. 48 h after transduction, begin antibiotic selection, and continue for 2 weeks. Transduction rates can be assessed using a fluorescence microscope. The construct should be visibly expressed in cells.
20. (Optional) If reporter expression levels are heterogenous within the population, fluorescence-activated cell sorting (FACS) may be conducted to isolate high reporter expressing cells or cells expressing multiple reporters. Alternative single cell cloning techniques may be employed to establish clonal populations (*see Note 9 Selection of Reporter Cells for discussion of options*).

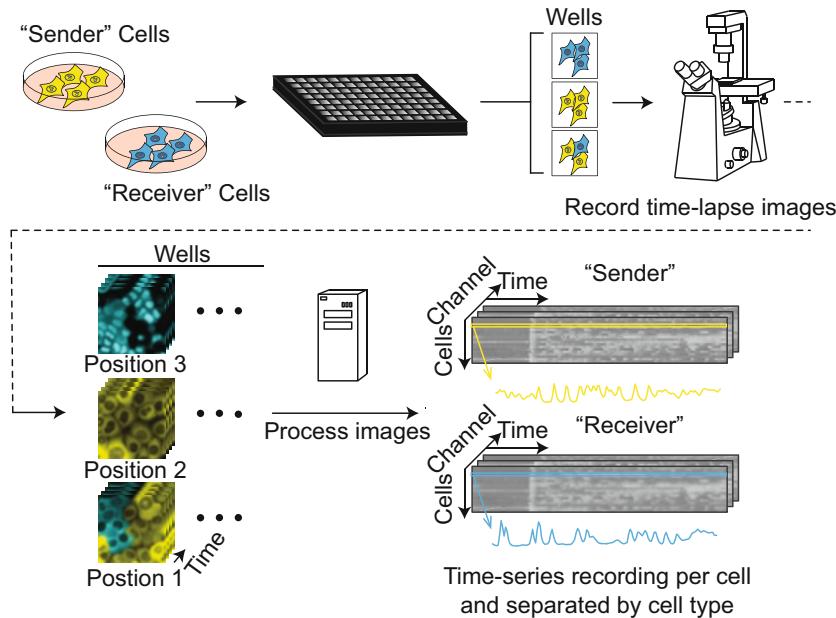


Fig. 3 Schematic example of major protocol steps. The complete experiment consists of (1) acquiring or producing DNA constructs for reporters, (2) inserting constructs into desired cell lines, (3) mixing “sender” and “receiver” cells in the desired experiment format (4) imaging co-cultured cells under desired treatments, (5) processing image data, and (6) analyzing time-series data according to the co-culture conditions used

3.4 Imaging Experiment Preparation

This section provides a basic outline of imaging plate preparation that can be utilized to validate cell lines, establish co-culture conditions, and conduct sender-receiver experiments. As basic diagram is provided in Fig. 3 and will be expanded upon in the following sections.

1. Prepare imaging plate by coating 96-well glass bottom cell culture plate with extracellular matrix. For a discussion of extracellular matrix selection choice, *see Note 10* Extracellular Matrix Choice.
2. If rat tail collagen is chosen, apply 3 μ L of 50 μ g/mL collagen solution (*see Subheading 2*) to the center of a well. Incubate at 37 °C for 30 min. After the incubation, add 100 μ L of PBS into each well to wash out unbound collagen; *see Note 11* Plating Technique for a discussion of spotting technique versus whole well plating.
3. If laminin-111 is chosen, apply 3 μ L of 50 μ g/mL laminin solution as prepared according to MATERIALS. Incubate at 37 °C for overnight. After the incubation, add 100 μ L of PBS into each well to wash out unbound laminin.
4. Trypsinize cells in accordance with the requirements of your chosen cell lines. Neutralize trypsin, and centrifuge cells.

5. Prepare cell culture media. If using the HMT-3522 cell lines, refer to a detailed description of media composition and cell handling [21, 23].
6. Resuspend and count cells using a hemocytometer. Dilute cells to 1000–3000 cells per μL in media. Optimal seeding density depends on cell line.
7. Aspirate the PBS from each well of the imaging plate, and ensure that collagen or laminin spot is not aspirated or dried.
8. Pipette 3 μL of cell suspension directly onto the spot. For best results, aspirate PBS and immediately pipette cells onto each spot. This is best done by aspirating 5–10 wells at a time and pipetting cells with a multi-channel pipette.
9. Incubate the plate at 37 °C and 5% CO₂ for 1 h.
10. Gently add 200 μL of media to each well, and incubate overnight.
11. Cells should be growth factor starved for at least 12 h before conducting and experiment. For example, starvation will decrease activity of ERK and allow for robust stimulation upon growth factor treatment.
12. If there is no nuclear marker stably expressed (e.g., H2B mCerulean), dilute Hoechst 33342 in imaging media to a concentration of 0.1–1 $\mu\text{g}/\text{mL}$ (*see Note 12 Imaging Media*).
13. Replace media with 200 μL of Hoechst containing imaging media, and incubate for 1 h.
14. Place plate on automated stage, and begin time course experiment, image Hoechst, and ERK-KTR reporter every 6 min for several hours (typically 12–24 h). The microscope stage must be equipped with a 37 °C and 5% CO₂ environmental chamber to maintain healthy, viable, cells.

3.5 Validation of Reporter Cells

Once reporter cell lines are generated, one must validate the function of the reporter in both sender and receiver cultures. In the case of ERK-KTR, we utilized EGF to stimulate signaling and establish the maximal and dose-dependent response for the reporter. Conversely, we verified the maximal reporter suppression that can be achieved by inhibiting the activation of ERK-KTR using a MEK inhibitor. The combination of stimulus and inhibitor use allows one to (1) establish the dynamic range and response characteristics of a particular reporter in the cell(s) of interest, (2) establish the baseline signaling activity of a signaling pathway within the receiver cells, and (3) evaluate the expression of a CRISPR tagged gene under these conditions, if present. *See Note 13* for a discussion of baseline signaling activity verification.

1. Prepare your experiment, and begin imaging as outlined in Subheading 3.4.

2. Collect at least 3 h of images before adding growth factor or inhibitors. This will reveal baseline ERK-KTR activity after starvation and allow cells to equilibrate to the imaging conditions.
3. Conduct a dose curve stimulation using EGF to activate ERK if using the ERK-KTR reporter. Final EGF concentrations in the media should range from 0.01 to 100 ng/mL and include a vehicle-only treatment. If using another reporter, choose the appropriate growth factor or stimulus. Add 10 µL of the treatment directly into the media for minimal perturbation of the cells. This 10 µL “spike in” should be 21× the desired final concentration; the final volume in each well will be 210 µL.
4. After 6–12 h of growth factor treatment and imaging, spike in the MEK inhibitor PD0325901 to reach a final concentration of 100 nM. This treatment will fully deactivate ERK signaling. Image for an additional 2 h. If using another reporter, choose the appropriate inhibitor and dosage.
5. After completing the experiment, visually inspect the images collected for nuclear translocation. The ERK-KTR construct should exit the nucleus after activation and enter the nucleus after inhibition.
6. Conduct cell tracking and segmentation using the nuclear marker. ERK-KTR activity can be calculated as the ratio of cytoplasmic/nuclear intensity at each timepoint then dose-dependent activation and deactivation visualized and quantified (*see* Subheading 3.7 Image Processing for details).

3.6 Establishing Co-culture Conditions

Establishment of optimal co-culture conditions is an essential step in this protocol because it directly effects sender-receiver dynamics (Fig. 4) and the quality of data obtained by these assays. To begin, one must consider the optimal seeding density, which will vary by cell line, since some cell lines prefer a high density to grow effectively and thus may need to be plated as such. However, this could lead to overcrowding and layered growth which can impede tracking of individual cell signaling behaviors. Therefore, careful optimization is required to achieve adequate density which promotes a uniform monolayer (unless utilizing confocal microscopy). One must also consider the properties of the sender-receiver relationship. For example, because AREG has a moderate affinity for EGFR, it distributes widely throughout the culture to reach receiver cells. Higher (or lower) affinity ligands may distribute with different kinetics, necessitating that sender-receiver plating ratios be optimized relative to the goals of a particular experiment. A stepwise discussion of our considerations is as follows:

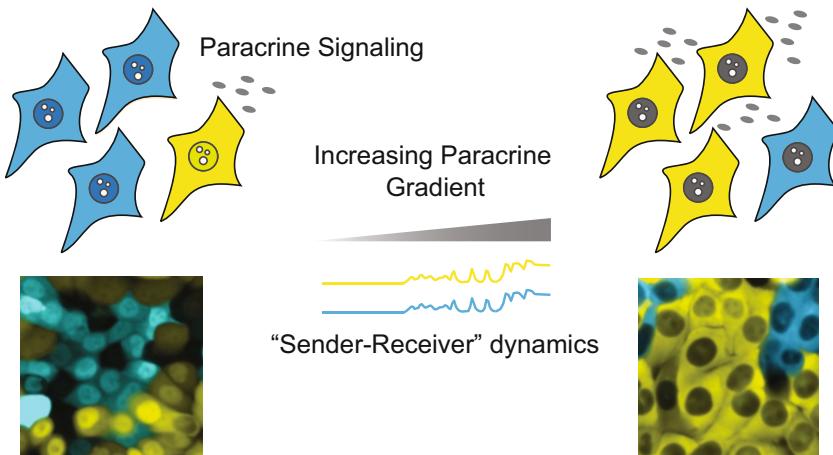


Fig. 4 Co-culture conditions dictate sender-receiver signaling dynamics. Left to right, increasing ratios of sender to receiver cells result in a progressive increase in both sender (autocrine mechanism, yellow cells) and receiver cell (paracrine mechanism, blue cells) signaling dynamics. Such dynamics correspond with increasing ligand abundance that is proportional to sender cell density

1. Follow general cell plating steps as outlined in Subheading 3.4.
2. Design an optimal plating density experiment. We suggest starting with a range of 1000–4000 cells per μL with increments of 500 cells per μL . For our experiments, we found 3500 cells per μL to give optimal plating density with appropriate monolayer formation. Lower cell densities tended to result in round cell morphology forming balls on the plate surface with reduced ERK activity. High cell densities resulted in crowding, layered growth, and impeded segmentation and tracking.
3. Establishing sender-receiver cell mixing ratios. For our experiments, we choose to optimize over a range from 10:90 to 90:10 sender-receiver cell ratios, respectively (*see Fig. 4*). We found an optimal ratio of 70:30 sender to receiver cells which gave dynamic ERK activity in both cell types with comparable signaling frequency and amplitude. Increasing the sender cell ratio resulted in decreased ERK activity, whereas increased sender cell ratios had no appreciable effect beyond a 70:30 fraction. When using this protocol, you will have to determine the optimal cell ratios based on the properties of the cell types used and the particular hypothesis being tested.

3.7 Image Processing

This section provides an overview on the extraction of single-cell data from time lapse microscopy images and distinguishing between nuclear and cytoplasmic subcellular compartments. As this protocol is focused on issues specific to co-cultures, we assume a working knowledge of processing time-lapse imagery. The procedure is largely unaffected by the presence of different cell types in culture, but there are several important considerations. For

protocols detailing lower level details of single-cell image analysis, see Pargett et al. 2017 [18]. While custom processing software is often used, there are a variety of accessible software tools to perform this processing, many of which include documentation (*see Note 3* for comments on software availability).

1. Import or access data with software of choice. Microscope software stores data in a variety of formats, often specific to a manufacturer. However, most bio-image processing software is designed to access many of these formats, often using the Bioformats software tool (www.openmicroscopy.org/bio-formats/) [34]. See software user guides for details.
2. Segment cell nuclei. This is performed for every image, using the color channel detecting a reporter or dye localized to the nucleus or cytoplasm. There are several widely used techniques broadly split between thresholding and machine learning. Thresholding is simpler and useful for images with clearly defined nuclei. Machine learning requires human interaction to “train” the algorithm by manually marking out nuclei in sample images. It is particularly useful when the images are highly complex and nuclei are not clearly distinct from the rest of the image (*see Note 14*). In our recent work, we achieved reliable segmentation using a thresholding approach for images with Hoechst-stained cell nuclei. Different cells in co-culture were identified by the presence of a particular color of ERK-KTR reporter (*see Note 15* on the use of differentially colored segmentation markers).
3. Estimate cytoplasm regions around each nucleus. Some machine learning approaches estimate the entire region of a cell’s cytoplasm, given an adequate membrane stain. However, in practice, it is typically sufficient to sample a ring surrounding the nuclear mask, which is the methods used in Davies et al. (2020) (*see Note 16*) [5].
4. Track nuclei over time. With cells identified in each frame, the software will link them from frame to frame, forming a time series. A variety of different algorithms are employed, each with differing strengths, but basic tracking is consistently well handled. See software user manuals for any details. Problems in tracking can often be mitigated at the experimental level, for example, optimizing cell density such that each cell is easily delineated. For detailed culture and imaging conditions that facilitate this process, *see Subheading 3.6*.
5. Collect and store times series data per tracked cell. Extract the single-cell time series data from the processing software to analyze with software of your choice (typically a scripting language, like Python, MATLAB, and R). The quantity extracted should clearly reflect the underlying feature measured by the

markers or reporters used. In many cases, the average intensity in each mask is appropriate, but other use cases exist. For example, if the reporter forms puncta, the average intensity of puncta only, or the number of puncta per mask area, may be more appropriate. Ultimately, the best metric depends on the nature of the reporter expressed or dye used (*see Note 17*).

3.8 Analysis of Co-culture Signaling and Gene Expression Responses

This section describes analyses of particular importance when assessing signaling between cell types in co-culture. Reporters for both signaling (e.g., ERK-KTR) and gene expression (e.g., CRISPR knock-ins) are handled equivalently. The only processing differences between reporter types occur in how they are initially interpreted (typically, gene expression is recorded as the average fluorescent intensity, while some reporters require additional interpretation, as with the nuclear-to-cytoplasmic ratio of ERK-KTR). A wide array of analyses is possible once the co-culture data can be reliably acquired. Those highlighted here represent a sampling of the most basic and widely applicable. Overall, these analyses allow one to quantitatively attribute differences in signaling and cellular responses to the quantity of different cell types present and to investigate the mechanism of intercellular signal transmission, for example, using drugs to block shedding of particular ligands.

1. Separate tracks from different cell types. To distinguish the differences between multiple cell types in culture, the tracks must be separated according to the marker profile of each cell line. If cells were segmented in separate runs, this is already complete and can be validated as needed. If cells were segmented in a single run, determine the cell type of each track by its marker profile (e.g., with a cyan and a yellow marker, cells will be either high cyan/low yellow or high yellow/low cyan). The best metric to distinguish will depend on the exact marker profiles but is typically straightforward. In the cyan versus yellow example, the ratio of cyan to yellow will show two distinct clusters, one high and one low. Below, we provide sample MATLAB code to set up an index identifying two cell types, assuming a 3D data array, Cells x Time x Channel.

```
cyan = mean(celldata(:,:,:,idx_cyan),2); %Get mean of
cyan intensity for each cell
yellow = mean(celldata(:,:,:,idx_yellow),2); %Get
mean of yellow intensity for each cell
cy_ratio = cyan ./ yellow; %Get ratio of cyan to
yellow
%If markers are similar in intensity range, the cell
identity can be called based on cy_ratio being
greater or less than one. i.e. iscyan = cy_ratio >
1;
```

```
%To be more robust, but a clustering algorithm
[cluster_id, c] = kmeans(cy_ratio, 2); %Get cluster
assignments.
%Make index vector identifying cyan vs. yellow
cells.
if c(1) > c(2) %If first cluster has the larger
ratio, assign as cyan, otherwise cyan is cluster 2.
iscyan = cluster_id == 1;
else
iscyan = cluster_id == 2;
end
```

2. Estimate average density per cell type. While the ratio of cell types is known at the time of plating, it is good practice to check the status at the time of imaging (typically days later). Calculate this estimate as the ratio of average cell number in the frame(s) for each cell type (*see Note 18*).

```
%Estimate the fraction of cells that are
cyan vs. yellow
fraction_cyan = sum(iscyan)./numel(iscyan);
fraction_yellow = 1 - fraction_cyan;
```

3. Analyze the effect of relative density on signaling. In our experience, high concentrations of receiver cells can result in low ERK activity because, as the fraction of sender cells decrease, so too does the effective concentration of ligand in the media. Correspondingly, the reporter activity of both sender and receiver cells is reduced. For our experiments, we defined the optimal ratio of sender to receiver cells at 70:30, resulting in roughly equivalent dynamic ERK activity in both cell types. In each new experimental setting, the sender/receiver ratio (either from original plating, or final estimates of relative density) can be compared with signaling and expression differences (with or without additional stimulus). Because single-cell data have been collected, not only response means, but their distributions (variance, skewness, any bi-modality, etc.) can be compared with the mixing ratios. Because signaling data are taken in a time series in this procedure, time-dependent features may also be compared, such as oscillations or other variability over time (*see Note 19*)

4. Calculate cross-correlation and time lag between cell types. On an average basis, the cell-type-specific responses to a stimulus can be compared, both for how they correlate in time. This is especially useful in a scenario where a receiver cell type may be responding to the activity of a sender cell type. For this average

analysis, it is important to deliver a stimulus that synchronously activates one (or both) cell types. Cross-correlation delivers two values: the maximum correlation between the signals and the lag time between them (e.g., how much later the receiver cells respond, after the sender cells activate).

```
[rho, lag] = xcorr(sender_avg, receiver_avg);
[rhomax, idx_rhomax] = max(rho); %Get maximum correlation
lagmax = lag(idx_rhomax); %Get lag corresponding to maximum correlation
```

5. Estimate the dependence of cellular responses on the spatial organization of cells. Knowing the position of each tracked cell in the culture, it is possible to investigate if signaling appears dependent on which other cells are nearby. One approach is to compare each cell's signaling trace with its proximity to other cells of each type. If there is a cell-type-specific interaction, cells surrounded by their own cell type would be expected to respond differently from those surrounded by other cell types. On a per cell basis, estimate the local density of each cell type, for example, as the number of tracked cells of each type within a chosen radius (*see Notes 20 and 21*).

```
%Calculate average signal for each cell
avg_signal = mean(celldata(:,:,:reporter_idx),2);
%Calculate average local density around each cell
cellxy = mean(celldata(:,:,:,[x_idx, y_idx]),2); % Get average x, y location values per cell
for s = 1:size(,1)
    cell_dist = sqrt(sum((cellxy(s,:)-cellxy).^2,2));
    %Distance to each other cell
    nearby_cells = cell_dist <= local_radius; %Flag each cell as local or not
    ldens(s,1) = sum(nearby_cells(iscyan)); %Count local cyan cells
    ldens(s,2) = sum(nearby_cells(isyellow)); %Count local yellow cells
end
%Correlate signal with local density
[rho, pval] = corr(avg_signal, ldens);
```

6. Extensions to mathematical modeling. The data gathered with this protocol open a wide range of options to inform and extend mathematical modeling studies. While these applications are outside the scope of this protocol, we briefly discuss

the potential. A typical scenario would involve a paracrine signaling model, describing the receiver response to a ligand shed by sender cells in the culture, potentially including expression of a downstream gene product. Dynamic signaling models may take a variety of forms, from more generic ARMAX (auto-regressive moving average exogenous) models to detailed ODEs (ordinary differential equations), or even spatial PDEs (partial differential equations). The data gathered via this protocol are applicable to all of these types of models. After formulating or adopting a relevant model, basic parameters may be fitted via dedicated datasets (*see Note 22*). Remaining parameters can then be fitted to the single-cell dataset(s) and model analyses performed (*see Note 23*). Single-cell interaction data allow these models to be tested at much higher resolution than was previously feasible.

4 Notes

1. Choice of Cell Lines. When choosing cell lines for sender-receiver studies the following must be considered: (1) Properties of the ligand will dictate the suitability of a particular sender cell line. Consider the disposition of the ligand. Is it secreted or membrane bound? If secreted, does the molecule freely diffuse or is it maintained bound to the local ECM? Such properties will determine the spatial range of ligand activity and will dictate the sender-receiver cell ratios to be determined later in this protocol. If the ligand is membrane bound, it will be important to determine if the ligand is accessible to stimulate receiver cell signaling and if the quantity expressed is capable of promoting detectable changes in signaling. This last consideration is less important for secreted ligands because most are secreted in excess within the microenvironment. Finally, does the ligand, whether membrane bound or secreted, require activation. Some ligands are secreted in an inactive form that requires enzymatic activation. In such cases, if the sender and receiver cells do not express this enzyme, signaling activity may not occur. (2) One must also consider ligand-receptor affinity. For certain high-affinity ligands that stimulate autocrine activation, secreted ligand may bind sender cell receptors, prior to diffusing, leading to weak or no detectable paracrine activity. (3) In choosing a receiver cell, one must consider if the cognate receptor is expressed in the cells. Usually simple confirmation using western blotting to verify receptor expression is sufficient. (4) Confirm that the secreted ligand is not produced by the receiver cell. If the ligand is expressed, one must consider if it is abundant enough to mask the effects of ligand produced by the sender cell. In cases where ligand is expressed at relatively

low levels and signaling is weakly activated, one can work around the elevated baseline by adapting the computational methods outlined in this protocol.

2. Choice of Genetically Encoded Reporters. The choice of reporter should be made based on the signaling network of interest and the overall experimental hypothesis. For example, binding of AREG to EGFR results in activation of both Ras-ERK and PI3K-Akt. Since several Akt and ERK reporters exist, either or both can be used depending upon the goals of the experiment. Consideration of FRET-based versus translocation reporters is more nuanced on the technical level and has been discussed elsewhere [14]. However, in general, translocation reporters are more easily employed, and the same reporter (e.g., ERK-KTR) is available or can be sub-cloned to produce reporters of different colors without effecting function.
3. Image Processing Software. Image processing software for live-cell microscopy is a rapidly evolving industry. At the time of this publication, there are many published implementations and a wide variety of additional techniques under development. Major microscope manufacturers (Nikon, Leica, etc.) and bio-science companies (ThermoFisher, PerkinElmer, etc.) offer image analysis software. Open-source implementations include Fiji, CellProfiler, BioImageXD, 3D Slicer, Icy, Ilastik, and many others. For those looking to set up a new software pipeline, we recommend performing a search to ensure an up-to-date survey of the field. Include any keywords related to additional complexities of the intended experiments, such as “puncta,” “3D,” or “mitochondria.” As many implementations are now formally distributed, they are accompanied with instructions for installation and use.
4. Considerations for CRISPR Gene Editing. Before proceeding, there are some important considerations that may affect the success of the gene knock-in study. First, we recommend generating knock-in cells prior to adding genetically encoded reporters. This process can be time-consuming, and generation of knock-in lines prior to reporter line generation has added flexibility – allowing for the same lines to be used with many different reporter combinations if desired. Second, we recommend considering the expected expression and dynamics of the POI and what property of protein expression is being studied. Examples of key properties include subcellular translocation, using dynamics of expression (on/off rates), or overall expression. Research prior Western blot, immunofluorescence, or mRNA expression studies to confirm that tagging the gene of interest will reveal useful information. If there is no published work, consider conducting a time course perturbation

experiment and measure the expression of the POI via Western blot or immunofluorescence. Third, consider the optimal color of the fluorescent protein (tag) that will be fused to the POI. If the target cell line will have another reporter integrated, the tag should be spectrally compatible with the reporter. The selected fluorescent protein should be fast folding to ensure the POI expression dynamics remain similar after knock-in. Additionally, the fluorescent protein should have a high quantum yield to allow for a bright signal during live-cell measurements. Furthermore, consider which end (C- or N-terminal) of the POI will be fused to the tag. Some studies suggest the C-terminal fusions are more likely to mimic native localization than N-terminal fusions [35]. In some cases, proteins are cleaved on the C-terminus (e.g., Ras); therefore, N-terminal tagging will be the only option [36]. If available, check the structure of the protein to examine the flanking residues of the POI. If these residues are tucked within the protein, fusing a fluorescent protein to that edge may disrupt proper folding, expression, or localization. In general, C-terminal fusions will be the preferred option; therefore, this protocol will describe a C-terminal tagging attempt. Of note, Koch et al. report that about 25% of genes cannot be functionally tagged; therefore, consider pursuing multiple proteins of interest when performing this protocol.

5. Gene Editing Efficiency. It is possible to increase the rate of homology-directed repair to increase knock-in efficiency by using inhibitors of the nonhomologous end joining pathway or activators of HDR [26, 37, 38]. Furthermore, other studies report a minor increase in homozygous insertion using serum starvation and release during the transfection [25].
6. Most fluorescent proteins are about 27 kDa; knock-in cell lines should have protein bands that are heavier than the non-tagged protein. Heterozygous knock-in cells are indicated by two bands (at the native and shifted sizes). Homozygous insertions are indicated by a single band at the shifted size and the absence of the native protein band. Samples should also be probed using a primary antibody specific to the fluorescent protein. Ensure that fluorescent protein-specific band is same size as the shifted POI band. There should not be other bands, which would suggest off-target knock-in events. Use the negative control (non-tagged) cell lines to test for nonspecific bands from the anti-fluorescent protein antibody.
7. When evaluating PCR product size, heterozygous insertions are indicated by the presence of both the native (non-tagged) size and larger (knock-in) DNA band. The larger band should be about 700 base pairs (length of the fluorescent protein) larger than the native band. Homozygous insertions are

indicated by the absence of the native size band and the presence of only the larger band.

8. The choice of fluorescent label(s) for secondary antibodies should be made to allow spectral distinction from the CRISPR-tagged gene(s) and any other reporters expressed in your cell line. Cell fixation and staining can be performed using standard immunofluorescence staining methods.
9. Selection of Reporter Cells. Once reporter cells are generated, a decision has to be made whether clonal or polyclonal populations are more desirable. Generating clonal populations by limited dilution cloning can provide an easy way to generate a population of cells that uniformly express the reporter(s) of interest. If choosing to create clonal populations, one must consider if the cell line is amenable to this process. Some cell lines do not grow well at low densities and may require the addition of conditioned media or half-media changes. Additionally, generating clones will reduce the level of heterogeneity within a cell line and essentially create a derivative cell lineage with altered slightly properties. This is especially true in some cancer cell lines which exhibit high levels of heterogeneity. Our preferred method is flow-based sorting of reporter (+) cell lines. In taking this approach, much of the population-level heterogeneity is preserved, and the vast majority of isolated cells express a moderate level of reporter expression. Since translocation and FRET reporter data is obtained in a ratio-metric manner, small fluctuations in reporter expression from cell to cell within these populations does not affect the data interpretation.
10. Extracellular Matrix Choice. The primary goal in choosing a particular ECM for plating is to try and recapitulate the host microenvironment of the cells selected. In our case, we have used collagen I which is abundant in breast cancer stroma or laminin-111 which is a predominant component of normal basement membrane in breast tissue. In our experience, choice of ECM *will* affect signaling dynamics. For example, we found that laminin results in higher frequency pulsatile EKR dynamics as compared to collagen I in the HMT-3522 cells lines. In designing experiments using this protocol, ECM choice should be made based on considerations of native microenvironment and experimental question.
11. Plating Technique. In our experiments, we prefer to use a “spotting” method as opposed to coating the whole well with ECM and cells. We have found several benefits to the method: (1) The plating area can be more precisely controlled with respect to the placement of cells in the center or the wells as opposed to whole well plating which leads to cell clustering at

the edges. (2) Precise cell distribution on the plate results in increased experimental repeatability. This consideration should not be underestimated because it can have a profound impact on plating density variation, formation of a uniform monolayer, and variation from well to well and experiment to experiment and can reduce the need for troubleshooting during data processing, particularly during segmentation steps. (3) Plating a small number of cells relative to the quantity of media present in a well ensures adequate nutrients are provided to the cells for the duration of the experiment. This is an important consideration in longer experiments, such as those lasting >48 h, resulting in fewer media changes.

12. Imaging Media. For optimal fluorescent imaging, it is recommended to use imaging media that do not autofluoresce in the spectra of interest. In particular, phenol red, riboflavin, folic acid, and serum often contribute to background fluorescence. Use media lacking for these components to lower background intensity and improve the ability to resolve faint expression. If any of these components are necessary for a cell line of interest, titrate it back into the base imaging media to identify the lowest concentration at which cell behave normally for the duration of the experiment.
13. Baseline Signaling Identification. An important step in reporter line and assay validation is to measure baseline activity of receiver cells. As discussed in **Note 1**, some receiver cells will have baseline activity of the pathway of interest. This can occur through several mechanisms, including (1) the presence of autocrine signaling in receiver cells resulting from secretion of the same ligand produced by sender cells, but at a relatively low level, and/or (2) activation of alternative pathways that converge on the signaling pathway of interest. For example, ERK can be stimulated by multiple signaling pathways, interactions with the ECM, and cell migration. Baseline signaling can be identified by experiments using a pathway inhibitor. Involvement of particular receptors and ligands can be assayed if suitable inhibitors or competing antibodies are available. When baseline signaling characteristics have been measured, the effects of sender-receiver interactions can be better quantified by comparison with the baseline distribution rather than a theoretical absence of all activity.
14. Regardless of the software platform used, it is advantageous to double-check the quality of the segmentation. Nuclear masks should be consistently within the nucleus, such that average intensities reflect only nuclear pixels. Assume that there will always be some fraction of cells that are not segmented and some rate of poor-quality masks. In many cases, the fraction

segmented is not critical, as long as most segmentations are of acceptable quality. However, if the intent is to perform local spatial analyses on a single-cell basis, more complete segmentation is increasingly important, as the analysis relies on knowledge of all nearby cells. It is therefore important to ensure that cultures are free of debris and not overcrowded and that the segmentation marker is distinct.

15. If the different colored markers used to distinguish cell types are fundamentally localized to the nucleus (or cytoplasm), cells of different types may be segmented in two separate processing runs, one for each color marker. Alternatively, the color channels for the two markers may be merged to make a joint segmentation channel. Color channels may be joined in a variety of ways, for example, by summing intensities or using the max of the two channels per pixel. If the intensities of the two channels differ greatly, they should be normalized so that the resulting joined images are more uniform. The choice of joining function should be made based on contrast in the final image. For example, if debris tend to autofluoresce in both channels, the sum will result in higher debris intensity, and a max projection may be preferred.
16. For cytoplasm sampling from a perinuclear ring, common mask errors can be corrected by avoiding pixels with high values for a nuclear marker, as well as those with background values for a cytoplasmic marker. Depending on the arrangement of colors across two (or more) cell types, these distinctions may or may not be available. Keep in mind that issues with cells overlapping can often be avoided by carefully managing density and plating conditions.
17. In many cases, the clarity of data can be improved via post-processing of signals using calibrations and/or mathematical models of the reporter's function. This is especially true of genetically expressed reporters, as in [39].
18. If the segmentation is relatively poor, the ratio may still be estimated based on the average intensities in the whole frame, provided that neither channel has large amounts of autofluorescent debris. For each channel, estimate the net intensity per cell by taking the sum of all pixels in the masks (or the average intensity per mask times that mask area) and dividing by the number of masks. To estimate the number of each cell type in the frame, divide the sum of the intensity over the entire frame by the intensity per cell.
19. The basic endpoint for effect of density is average signaling level (or gene expression) as a function of the mixing ratio. This endpoint should be considered independently for each cell type; effects on receiver cells reflect their potential to be directly

by this paracrine signaling, while effects on sender cells may reflect their reliance of para- or autocrine signaling and how their behavior may change when diluted by different cell types.

20. To be robust to poorer segmentation, estimate local density as a weighted sum of the intensity of each cell type marker. A typical weighting scheme is a Gaussian function of the distance from the target cell's nucleus. This is equivalent to sampling a Gaussian filtered image.
21. For a more generalized approach considering different features of the signaling response, use a partial least squares regression (PLSR). PLSR allows for simultaneously testing the correlation among many variables (such as the mean, max, and frequency of a signal vs. the local density of two different cell types).
22. Wherever possible, it is recommended to use independent data to determine key parameters (or parameter ranges) in a model. For example, the average decay rate of a protein may be estimated by time series Western blot and that for an mRNA by qPCR. Making these measurements independently allows the main dataset to be used to estimate the more difficult to observe regulatory interactions.
23. Using single-cell data opens opportunities as well as new challenges. From a modeling perspective, we are forced to acknowledge that each individual cell may have a different amount of each protein (receptors, kinases, etc.) and, as a result, different apparent kinetic rates for complex reactions. Therefore, each single-cell trace potentially reflects different parameter values in a model (how different depends on the context of the system). Each cell may also be experiencing a different signaling environment (different amounts of a ligand at any point in time). To make the fullest use of single-cell data, modeling studies should be prepared to estimate different parameter values for each cell. As this can become computationally infeasible with many individual cells, a subset of representative cells may be chosen for explicit modeling (e.g., based on observed clusters in the dataset).

Acknowledgments

These methods were developed in part by support from the Office of Research Infrastructure Programs of the National Institutes of Health under award number K01OD031811-01 and The Ohio State University Comprehensive Cancer Center and National Institutes of Health under grant number P30 CA016058.

References

1. Davies AE, Albeck JG (2018) Microenvironmental signals and biochemical information processing: cooperative determinants of intra-tumoral plasticity and heterogeneity. *Front Cell Dev Biol* 6:44
2. de la Cova C et al (2017) A real-time biosensor for ERK activity reveals signaling dynamics during *C. elegans* cell fate specification. *Dev Cell* 42(5):542–553 e4
3. Hamilton WB et al (2019) Dynamic lineage priming is driven via direct enhancer regulation by ERK. *Nature* 575(7782):355–360
4. Kenny PA, Bissell MJ (2007) Targeting TACE-dependent EGFR ligand shedding in breast cancer. *J Clin Invest* 117(2):337–345
5. Davies AE et al (2020) Systems-level properties of EGFR-RAS-ERK Signaling amplify local signals to generate dynamic gene expression heterogeneity. *Cell Syst* 11(2):161–175 e5
6. Inman JL et al (2015) Mammary gland development: cell fate specification, stem cells and the microenvironment. *Development* 142(6):1028–1042
7. Matos I et al (2020) Progenitors oppositely polarize WNT activators and inhibitors to orchestrate tissue development. *elife* 9:e54304
8. Yang H et al (2017) Epithelial-mesenchymal micro-niches govern stem cell lineage choices. *Cell* 169(3):483–496 e13
9. Luetke NC et al (1999) Targeted inactivation of the EGF and amphiregulin genes reveals distinct roles for EGF receptor ligands in mouse mammary gland development. *Development* 126(12):2739–2750
10. Wiesen JF et al (1999) Signaling through the stromal epidermal growth factor receptor is necessary for mammary ductal development. *Development* 126(2):335–344
11. Ciarloni L, Mallepell S, Brisken C (2007) Amphiregulin is an essential mediator of estrogen receptor alpha function in mammary gland development. *Proc Natl Acad Sci U S A* 104(13):5455–5460
12. Meier DR et al (2020) Amphiregulin deletion strongly attenuates the development of estrogen receptor-positive tumors in p53 mutant mice. *Breast Cancer Res Treat* 179(3):653–660
13. Mao SPH et al (2018) Loss of amphiregulin reduces myoepithelial cell coverage of mammary ducts and alters breast tumor growth. *Breast Cancer Res* 20(1):131
14. Sparta B et al (2015) Receptor level mechanisms are required for Epidermal Growth Factor (EGF)-stimulated Extracellular Signal-regulated Kinase (ERK) activity pulses. *J Biol Chem* 290(41):24784–24792
15. Bertolin G et al (2019) Optimized FRET pairs and quantification approaches to detect the activation of Aurora Kinase A at mitosis. *ACS Sens* 4(8):2018–2027
16. Regot S et al (2014) High-sensitivity measurements of multiple kinase activities in live single cells. *Cell* 157(7):1724–1734
17. Pargett M, Albeck JG (2018) Live-cell imaging and analysis with multiple genetically encoded reporters. *Curr Protoc Cell Biol* 78(1):4 36 1–4 36 19
18. Pargett M et al (2017) Single-cell imaging of ERK signaling using fluorescent biosensors. *Methods Mol Biol* 1636:35–59
19. Kudo T et al (2018) Live-cell measurements of kinase activity in single cells using translocation reporters. *Nat Protoc* 13(1):155–169
20. Gillies TE et al (2017) Linear integration of ERK activity predominates over persistence detection in Fra-1 regulation. *Cell Syst* 5(6):549–563 e5
21. Rizki A et al (2008) A human breast cell model of preinvasive to invasive transition. *Cancer Res* 68(5):1378–1387
22. Weaver VM et al (1997) Reversion of the malignant phenotype of human breast cells in three-dimensional culture and in vivo by integrin blocking antibodies. *J Cell Biol* 137(1):231–245
23. Briand P, Petersen OW, Van Deurs B (1987) A new diploid nontumorigenic human breast epithelial cell line isolated and propagated in chemically defined medium. *In Vitro Cell Dev Biol* 23(3):181–188
24. Stewart-Ornstein J, Lahav G (2016) Dynamics of CDKN1A in single cells defined by an endogenous fluorescent tagging toolkit. *Cell Rep* 14(7):1800–1811
25. Koch B et al (2018) Generation and validation of homozygous fluorescent knock-in cells using CRISPR-Cas9 genome editing. *Nat Protoc* 13(6):1465–1487
26. Bottcher R et al (2014) Efficient chromosomal gene modification with CRISPR/cas9 and PCR-based homologous recombination donors in cultured *Drosophila* cells. *Nucleic Acids Res* 42(11):e89
27. Chiang T-WW et al (2016) CRISPR-Cas9D10A nickase-based genotypic and phenotypic screening to enhance genome editing. *Sci Rep* 6(1):24356

28. Joung J et al (2017) Genome-scale CRISPR-Cas9 knockout and transcriptional activation screening. *Nat Protoc* 12(4):828–863
29. Kime C et al (2016) Efficient CRISPR/Cas9-based genome engineering in human pluripotent stem cells. *Curr Protoc Hum Genet* 88: 21.4.1–21.4.23
30. Chen X, Zaro JL, Shen WC (2013) Fusion protein linkers: property, design and functionality. *Adv Drug Deliv Rev* 65(10):1357–1369
31. van Rosmalen M, Krom M, Merkx M (2017) Tuning the flexibility of glycine-serine linkers to allow rational design of multidomain proteins. *Biochemistry* 56(50):6565–6574
32. Maryu G, Matsuda M, Aoki K (2016) Multiplexed fluorescence imaging of ERK and Akt activities and cell-cycle progression. *Cell Struct Funct* 41:16007
33. Dessauges C, Pertz O (2017) Developmental ERK Signaling goes digital. *Dev Cell* 42(5): 443–444
34. Linkert M et al (2010) Metadata matters: access to image data in the real world. *J Cell Biol* 189(5):777–782
35. Palmer E, Freeman T (2004) Investigation into the use of C- and N-terminal GFP fusion proteins for subcellular localization studies using reverse transfection microarrays. *Comp Funct Genom* 5(4):342–353
36. Manolaridis I et al (2013) Mechanism of farnesylated CAAX protein processing by the intramembrane protease Rce1. *Nature* 504(7479): 301–305
37. Srivastava M et al (2012) An inhibitor of non-homologous end-joining abrogates double-strand break repair and impedes cancer progression. *Cell* 151(7):1474–1487
38. Nambiar TS et al (2019) Stimulation of CRISPR-mediated homology-directed repair by an engineered RAD18 variant. *Nat Commun* 10(1):3395
39. Gillies TE et al (2020) Oncogenic mutant RAS signaling activity is rescaled by the ERK/MAPK pathway. *Mol Syst Biol* 16(10):e9518



Chapter 14

Application of Optogenetics to Probe the Signaling Dynamics of Cell Fate Decision-Making

Heath E. Johnson

Abstract

The development of optogenetic control over signaling pathways has provided a unique opportunity to decode the role of signaling dynamics in cell fate programming. Here I present a protocol for decoding cell fates through systematic interrogation with optogenetics and visualization of signaling with live biosensors. Specifically, this is written for Erk control of cell fates using the optoSOS system in mammalian cells or *Drosophila* embryos, though it is intended to be adapted to apply generally for several optogenetic tools, pathways, and model systems. This guide focuses on calibrating these tools, tricks of their use, and using them to interrogate features which program cell fates.

Key words Optogenetics, Cell signaling, Cell fate, Signaling dynamics, Erk, optoSOS

1 Introduction

For decades, we have known the signaling pathways required for many cell fates, though for very few of these fates do we know the precise signaling dynamics which are required. In the last decade, optogenetic tools have been developed that allow us to directly control signaling pathways. With these tools, we now have the potential to resolve the underlying dynamics responsible for inducing fates, a critical piece of information in quantitatively understanding biology. Here I describe a general protocol for elucidating the signaling requirements for an underlying cell fate using optogenetics. Every cell fate will have its own unique challenges associated with it, but this protocol is written in an attempt to generalize these so that it can be done with varied optogenetic tools and systems. This protocol has been written specifically for use with the blue-light optoSOS [1–3] system to control Erk activity in mammalian cells or *Drosophila*. However, with some small adjustments, it could be applied to several other optogenetic systems as well.

The optoSOS system used herein is a heterodimerizing blue-light optogenetic system. The light-sensitive component, iLID [4], is fused to plasma membrane. Upon blue light stimulation, it changes conformation, allowing it to heterodimerize with its cytosolic binding partner, SSPB. SSPB is fused to the catalytic domain of SOS, which can activate Ras when it is localized to the membrane, leading to Erk activation. Additionally, the SSPB component is fused to a red fluorophore to enable visualization of this translocation to aid in the calibration of the light response of the system. By combining this tool with a kinase translocation reporter for Erk, we can both control and visualize the input to the system. Thus, in the absence of an uncontrolled endogenous Erk signal, we can clearly define the parameters of signaling which are responsible for programming a cell fate.

2 Materials

1. OptoSOS [1, 2, 5] (*see Note 1*) expressing cells or tissues (Addgene #86439) or another optogenetic tool with a high dynamic range to control your signaling pathway of interest (*see Note 2*).
2. Red- or infrared-tagged kinase translocation reporter (KTR) for Erk [6–9] (Addgene #111510 or #90231) (*see Note 3*).
3. A marker of cell fate or phenotype of interest (*see Note 4*).
4. A confocal microscope with a 561 laser and filters and at least 1 of the following in order of preference:
 - 450 nm LED coupled digital micromirrors attached (*see Note 5*).
 - A shutterable epi-illumination or transmitted light source that can produce ~450 nm light.
 - A 488 laser.
5. Red LED panel(s) (*see Note 6*).
6. Red filter glass or transparencies (*see Note 7*).
7. (Optional, *see Note 8*) Micro-controlled LED light plates. Can be constructed [10] or purchased (e.g., Amuza Inc).

3 Methods

3.1 Calibrating of Optogenetic Tools

In order to dissect cell fates, we must first calibrate our tools to determine the relationship between light and signaling.

1. Set up an environment where you can care for, mount, and image samples without exposing them to blue (<500 nm) light. Perform subsequent steps in this environment to avoid unintentional activation (*see Note 9*).

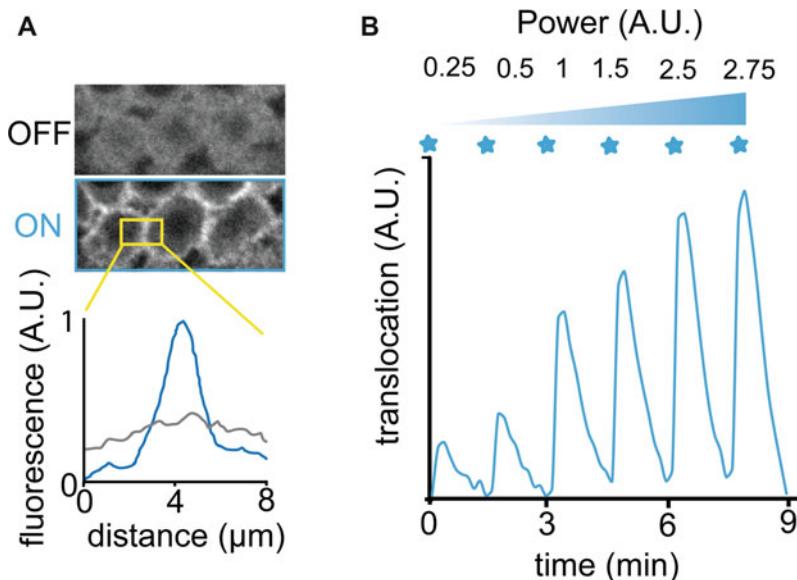


Fig. 1 Calibrating translocation. **(a)** Translocation of optoSOS to the plasma membrane in response to blue light. Performing a line scan across the region yields the resulting curve for pre- and poststimulation. **(b)** These curves can be generated for pulses of varying intensity to generate a light-translocation calibration curve. Note that you should continue this curve until saturation is achieved (not shown here)

2. Determine the phototoxicity limit for your model system at the activating wavelength (see Note 10). Place samples not containing the optogenetic construct in varying levels of light for the maximum duration that would be needed to assess a fate change and determine the highest level of light that is still viable.
3. Determine the level of light at which optogenetic tool or controlled signaling pathway is saturated. At an intensity of light just below the sustained light phototoxicity limit determined in step 2, illuminate your sample while imaging translocation and determine the time of illumination at which it saturates. If you are expressing the Erk-KTR in the same cells, you can perform step 7 simultaneously.
4. Measure the translocation of SSPB by computing the ratio between membrane intensity in the light-stimulated image versus the dark state image (Fig. 1a).
5. Reduce the intensity of the light at this same duration until there is a reduction in translocation. Return to the level of light just before this occurred, *this will be the maximum intensity of light used from here on*. This light load can be further reduced if phototoxicity or spatial precession is an issue (see Note 11 and 12).

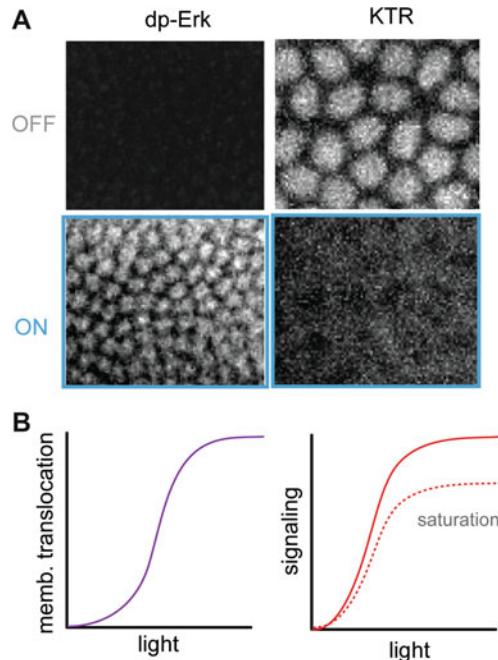


Fig. 2 Developing calibration curves for signaling. **(a)** Phospho-Erk staining and KTR translocation in response to optoSOS activation. Note that the KTR is expressing in the same channel as optoSOS, and thus membranes are more visible when it is activated. **(b)** Example of calibration curves generated through these and the translocation experiments. Saturation of your pathway or biosensor may result in a signaling curve that saturates prematurely (dashed line) compared to the translocation calibration curve. If this happens, you should use phospho-staining to determine if this is due to the biosensor or the actual pathway

6. Apply pulses of light at the previous duration, titrating the level of light using the pulsing method (*see Note 13*) or by modulating the amplitude directly each time until translocation is no longer distinguishable from no stimulation. You now should have a series of points to build a calibration curve of translocation → light (Fig. 1b).
7. Measure the signaling delay by applying saturating light until the KTR is fully excluded (*see Note 14*, Fig. 2a).
8. Determine the calibration of light to signaling using the KTR (also possible with staining for marker of activated signaling). Apply pulses along your light-to-translocation curve, and measure the translocation of the KTR. It is best to apply this to a fresh field each time in case of feedback (Fig. 2b).
9. Calculate the signaling based on the inverse change KTR fluorescence (*see Note 15*).

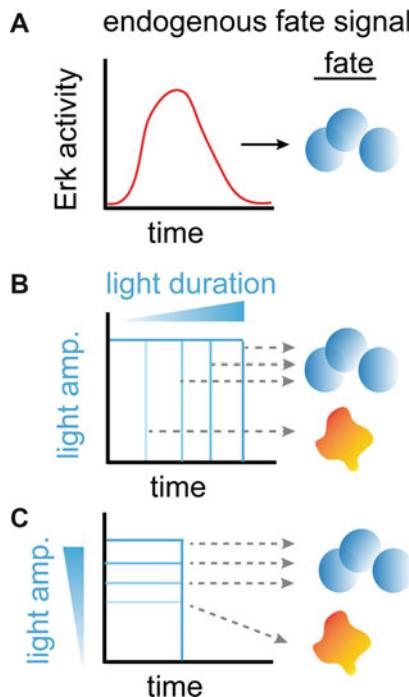


Fig. 3 Determining the minimum requirements for cell fate. (a) Measure the normal/endogenous dynamics that you know generate your fate of interest using the KTR. (b) Starting at a maximum amplitude and duration of the normal dynamics, systematically reduce the duration until the fate is switched/lost. (c) At the shortest duration that gave the original fate, reduce the amplitude of the signal (through pulsing or amplitude of light) until the fate is no longer maintained

10. Apply a sustained activation (on the timescale of your cell fate) to make sure that you are measuring the peak activity and feedback is not going to be a large issue (*see Note 16*).

3.2 Determining the Minimum Signaling Dynamics for Cell Fate

Now that you know the range and capabilities of your optogenetic tool, you can begin to use to attempt to program cell fates. Some of the details here will depend on the specifics of your fate assay.

1. A good starting point for programming cell fates is often to use the dynamics of signaling that is known to trigger these fates normally as a guide (Fig. 3a). Use the KTR to measure the dynamics which normally induce your fate (either from an endogenous signal or an added ligand). If multiple pulses are observed, *see Note 17*.
2. Apply light for a duration (if known, otherwise use the entire time over which the fate could be triggered) of the wild-type signaling at the maximum stimulation level. If there are more than one fate-triggering signaling event, mimic the longer one. If this fails to trigger a fate change, *see Note 18*.

3. Systematically reduce the duration of light until there is a change in fate, or a lack of programming of fate (Fig. 3b). If you are using a developing system, *see Note 19*.
4. Using the minimum duration determined in the step above, reduce the level in small increments, by changing amplitude of the light or by the pulsing method (*See Note 13*), until there is a change in the fate observed (Fig. 3c).
5. In order to distinguish between more complex signal interpretation, repeat the 3.3 and 3.4 in reverse order, first finding the minimum amplitude and then minimum duration at that amplitude.
6. If the same result was achieved, it appears a single threshold of Erk determines your cell fate.
7. If you did not get the same result in 3.5: If cumulative load is being read, it should not matter how the load is delivered. If this is a reasonable hypothesis, you can explore it further by splitting up the signal. Using the minimum level and total duration from 3.4 or 3.5, split light into multiple pulses (you can start with 2 or 3 and increase after if desired), leaving a period of at least the known off time for the pathway between them keeping the total duration equal to the determined minimum. If the fate is maintained, you may want to increase the distance between pulses and the number of pulses to explore its limits.

Taking the data you have acquired in, you should now have a very clear idea of the minimum signaling requirements of your cell fate and thus insight into how it is triggered!

4 Notes

1. The protocol is written mostly based on the *Drosophila* blue-light version of the OptoSOS system but is intended to work for the mammalian version as well. The red-light (phytochrome B/PIF6 based) OptoSOS system [5] can also be used, but changes will need to be made to include the phyco-cyanobilin (PCB) chromophore and appropriate wavelengths of light.
2. A wealth of off-the-shelf optogenetic tools is already available to control signaling. Numerous reviews exist on the construction, choice, and usage [11–15], of the tools. In order to attempt to decode the signaling dynamics of a cell fate, you will need to have control over a reasonable dynamic range of signaling and temporal dynamics. Several of the activating tools work at the receptor level so one must be aware of the effects of parallel pathways being activated as well. Inhibitors or mutants may be used to isolate the effects of a single pathway.

3. These are not strictly required. Phospho-staining for active, doubly phosphorylated Erk (e.g., using Cell Signaling Technologies anti-dp-Erk antibody #4370, which works for both mammalian and *Drosophila* cells) can be used instead, but a live readout certainly makes things easier. Note that the mammalian and fly KTR versions are different and will have different kinetics. Fluorescently tagged Erk is not recommended due to its limited sensitivity and strong variation in response depending on expression level [16]. You will want to be able to image these without activating your tool, so they will be ideally tagged with a red or infrared fluorescent protein. It is possible to image optoSOS and KTR in the same channel as optoSOS does not seem to localize to the nucleus. Fluorescently tagging the membrane component of the optogenetic system is not necessary, as it does not translocate.
4. At a minimum, you also need a way of tracking your fate. This can often be a visible morphological change, such as with growth or differentiation, or in vivo phenotype. In lieu of an obvious visible change, one can use transcriptional reporters [17] or fix and stain for expression of genetic markers.
5. Depending on what you have available to you, different methods for activation of your optogenetic tool can be used. Since many of the suggested experiments are performed on the microscope, the simplest solution is to use an illumination source on the microscope to simulate the optogenetic tool. Digital micromirror devices are ideal for this can allow you to apply multiple inputs in a single field, allowing you to perform different activation parameters within a single field. If you use the laser, be sure not to use too much power as this can destroy the photoswitching capabilities of many optogenetic tools making them appear to not be functional. If you use the transmitted light illumination, this can often hit the entire sample, not just what is under the objective, and can potentially decouple stimulation from imaging. However, you may want to place filter glass (filtering out non-activating wavelengths) over it to prevent excess light in other wavelengths from hitting the sample. You simulate your sample simply by imaging it, or toggle the shutter using a macro, without actually acquiring a separate image.
6. These do not really need to be anything special, and they just need to not activate your samples and provide some light to be able to set up your experiment. LED panels with only red lights generally work well for this purpose.
7. You will want to filter out the lower wavelengths of your transmitted light source on the microscope as to not stimulate a sample when locating a field of cells. You can simply put filters or transparencies into the light path to accomplish this.

8. You may use these plates to perform experiments in a higher throughput manner and avoid tying up the microscope once you have performed the initial calibration. As long as the light is saturating in both the light box and the microscope, you can switch between devices with the same calibration.
9. You will need a space where you can turn off the lights and close the blinds. If this cannot be done due to a shared space, a curtain can be used to partition a space in a room to be used. Alternatively, you may use transparent tents to filter out undesired wavelengths coming from lights or windows, though this is less desirable if the room is used for other purposes. You will likely want to put filter glass or transparencies over the transmitted light source on your microscope so you can locate your sample without stimulating it. Note that for iLID as with many blue light systems, there may still be some absorbance above 500 nm, so it is advisable to go well above that if possible. Once you have a dark space, you will want to install some sort of lights in a wavelength which does not activate your optogenetic tools. Plant-growing LED panels can be purchased in red or blue wavelengths cheaply for this purpose. You will then want to expose a sample to your safe area for some time and then determine if any significant activation has taken place (via staining or phenotype).
10. You should benchmark your system to determine the maximum tolerable light levels/load at the wavelength you use. Lower wavelengths tend to be more toxic, but far-red wavelengths are absorbed by water and can heat your sample, making them indirectly phototoxic. Also, be aware that if you are using LEDs which are physically close to your sample, they often produce a good deal of heat on their own, so make sure they are sufficiently far from the sample and ventilated such that hot air does not build up, specifically with incubators which cannot cool and only heat.
11. Photoswitching of the iLID system, like most optogenetic tools, is extremely fast, but the timescales of binding and reversion are generally much slower. Therefore, one can significantly reduce the amount of light needed to photoactivate by using very short light pulses in rapid succession (Fig. 4). For iLID, pulsing every 5 s for ~100 ms should yield nearly maximal activity but would use 50-fold less light than constant on light at the same intensity.
12. In many cases, it might be desirable to only optogenetically activate subsets of cells. The simplest way to do this is to only express your construct in these cells, using a local driver. However, this is not always an option; thus, you will want to control the spatial range of the light itself. Digital micromirrors are

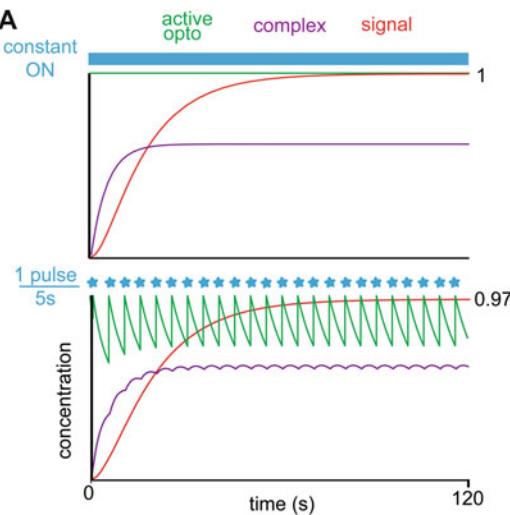


Fig. 4 Reduction of light load through rapid pulsing. While pulsing can be used to modulate amplitude, at frequencies faster than there is significant reversion of the optogenetic tool, the effect on signaling is negligible, but the light load can be drastically reduced. This figure shows the modeled kinetics for constant light (top) vs arbitrarily short light pulses every 5 s (bottom). This can significantly reduce light exposure, but it results in effectively the same signaling outcome (97% of constant light in this case)

generally the best way to do this; though, if you do not have access to these, a point scanner may be used. In most cases, you will want to use nearly the lowest laser power you can, since it is quite bright and scattering will activate cells in the surrounding region.

13. It is often desirable to change the amplitude of the signaling applied optogenetically. This can be done by changing the intensity of a continuous light source applied to the sample (intensity variation). However, intensity variation presents additional challenges. The intensity of the light is a strong function of distance from the source (varying with the square of distance) and can differ substantially between light sources. Furthermore, the light intensity produced by bulbs at a given voltage depends on the resistance, which can vary throughout the lifetime of a single device. Luckily, there is a way to avoid these complications in most systems by replacing intensity changes with a bright light source that is pulsed on and off to control the amplitude of the resulting signal (pulse width modulation). As long as each pulse fully converts the photoactive protein to its illuminated state, varying the timing between pulses will vary the time-averaged activity of the protein and drives intermediate levels of downstream pathway activity that do not depend on the light source intensity. An

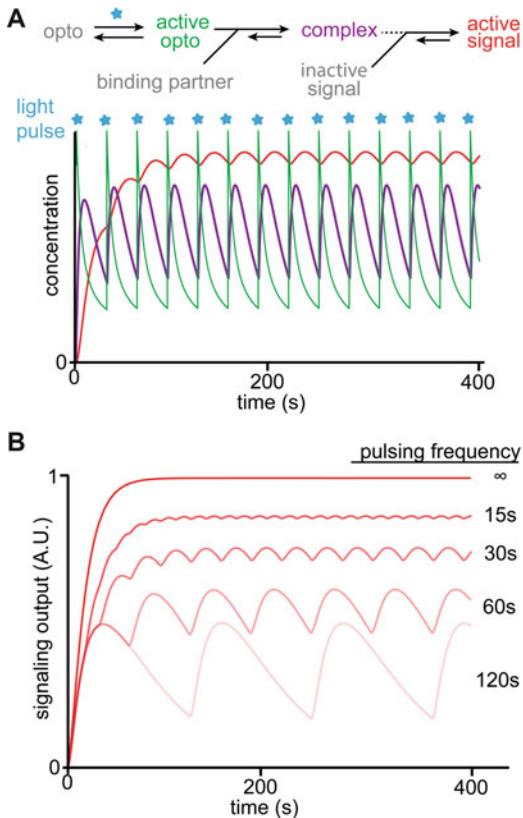


Fig. 5 Simple model of controlling signaling amplitude through pulsing. **(a)** Schematic of the model of the response of optoSOS to light. Light switches the optogenetic tool to its active form near instantaneously, which it can bind its target. Once it is in this complex, it can initiate signaling which decays with time. **(b)** Varying period between pulses can be used to achieve different levels of signaling at steady state. As the time between pulses becomes on similar timescales as the optogenetic dark reversion ~ 120 s, the signaling level becomes more irregular and no longer suitable for varying amplitude. Note that these model parameters are rough estimates based on the systems use in *Drosophila* embryos, and these pulsing times will vary in your system and should be measured

example of this is shown in Fig. 5 for a heterodimerizing optogenetic tool which binding induces signaling; however, this would be similarly true for a homodimerizing tool or potentially for a deactivating tool depending on the background signaling and its dynamics. This should work as long as your optogenetic system is faster than the response of the pathway, so that downstream signaling averages out the pulses into a more uniform response. This method will start to break down if the intensity is sufficiently low or the duration of the pulses becomes sufficiently short so that each pulse is no longer saturating.

14. Depending on your relative expression levels, the KTR may saturate before the actual pathway does. Direct phospho-staining for the active, dually phosphorylated form of Erk can be used to control for this.
15. Since the KTR leaves the nucleus in response to Erk activity, it is the inverse of the change which you are measuring. The change in cytosol is often much harder to measure and is not necessary. Co-imaging of a nuclear marker (often in the infrared space, e.g., mIFP-H2B) can make automated measurements much easier. There are more advanced methods that can be used for quantification such as gradient changes [1] if these results are too noisy.
16. Signaling pathways generally have numerous sources of feedback that your tool may or may not be shielded from. The further downstream in the pathway your tool acts, the less of an issue this is, though it is unlikely any system is completely free of feedback. OptoSOS seems to be mostly insulated from this, but this is not necessarily true in every context, and thus a good idea to check.
17. This is written for a single pulse of Erk activity; if you have multiple, you will need to test them individually and as a single long pulse.
18. You may find that your signaling pathway alone is insufficient to program the cell fate on its own. Receptors often activate multiple signaling pathways, and optogenetic activation downstream in a single pathway may not fully recapitulate the effects. You may be able to get around this by providing the signal which you cannot control ubiquitously and manipulating the other. Alternatively, in some situations, you could use a tool higher up in the pathway or at the receptor level to attempt to restore the crosstalk in the native system.
19. Here there is an additional variable of developmental timing. You can start with the natural time window for signaling. When reducing the duration, you will need to apply these pulses at different points throughout the time window as timing may be as important as duration.

References

1. Johnson HE, Goyal Y, Pannucci NL et al (2017) The spatiotemporal limits of developmental Erk signaling. *Dev Cell* 40:185–192. <https://doi.org/10.1016/j.devcel.2016.12.002>
2. Goglia AG, Wilson MZ, Jena SG et al (2020) A live-cell screen for altered Erk dynamics reveals principles of proliferative control. *Cell* 10:240–253.e6. <https://doi.org/10.1016/j.cels.2020.02.005>
3. Johnson HE, Toettcher JE (2019) Signaling dynamics control cell fate in the early Drosophila embryo. *Dev Cell* 48:361–370.e3. <https://doi.org/10.1016/j.devcel.2019.01.009>
4. Guntas G, Hallett RA, Zimmerman SP et al (2015) Engineering an improved light-

- induced dimer (iLID) for controlling the localization and activity of signaling proteins. *Proc Natl Acad Sci U S A* 112:112–117. <https://doi.org/10.1073/pnas.1417910112>
- 5. Toettcher JE, Weiner OD, Lim WA (2013) Using Optogenetics to interrogate the dynamic control of signal transmission by the Ras/Erk module. *Cell* 155:1422–1434. <https://doi.org/10.1016/j.cell.2013.11.004>
 - 6. Regot S, Hughey JJ, Bajar BT et al (2014) High-sensitivity measurements of multiple kinase activities in live single cells. *Cell* 157: 1724–1734. <https://doi.org/10.1016/j.cell.2014.04.039>
 - 7. Kudo T, Jeknić S, Macklin DN et al (2018) Live-cell measurements of kinase activity in single cells using translocation reporters. *Nat Protoc* 13:155–169. <https://doi.org/10.1038/nprot.2017.128>
 - 8. Dine E, Gil AA, Uribe G et al (2018) Protein phase separation provides long-term memory of transient spatial stimuli. *Cell Syst* 6:655–663.e5. <https://doi.org/10.1016/j.cels.2018.05.002>
 - 9. Moreno E, Valon L, Levillayer F, Levayer R (2019) Competition for space induces cell elimination through compaction-driven ERK downregulation. *Curr Biol* 29:23–34.e8. <https://doi.org/10.1016/j.cub.2018.11.007>
 - 10. Bugaj LJ, Lim WA (2019) High-throughput multicolor optogenetics in microwell plates. *Nat Protoc* 14:2205–2228. <https://doi.org/10.1038/s41596-019-0178-y>
 - 11. Johnson HE, Toettcher JE (2018) Illuminating developmental biology with cellular optogenetics. *Curr Opin Biotechnol* 52:42–48. <https://doi.org/10.1016/j.copbio.2018.02.003>
 - 12. Goglia AG, Toettcher JE (2019) A bright future: optogenetics to dissect the spatiotemporal control of cell behavior. *Curr Opin Chem Biol* 48:106–113. <https://doi.org/10.1016/j.cbpa.2018.11.010>
 - 13. Krueger D, Izquierdo E, Viswanathan R et al (2019) Principles and applications of optogenetics in developmental biology. *Development* 146. <https://doi.org/10.1242/dev.175067>
 - 14. Repina NA, Rosenbloom A, Mukherjee A et al (2017) At light speed: advances in optogenetic systems for regulating cell signaling and behavior. *Ann Rev Chem Biomol Eng* 8:13–39. <https://doi.org/10.1146/annurev-chembioeng-060816-101254>
 - 15. Tischer D, Weiner OD (2014) Illuminating cell signalling with optogenetic tools. *Nat Rev Mol Cell Biol* 15:551–558. <https://doi.org/10.1038/nrm3837>
 - 16. Wilson MZ, Ravindran PT, Lim WA, Toettcher JE (2017) Tracing information flow from Erk to target gene induction reveals mechanisms of dynamic and combinatorial control. *Mol Cell* 67:757–769.e5. <https://doi.org/10.1016/j.molcel.2017.07.016>
 - 17. Pichon X, Lagha M, Mueller F, Bertrand E (2018) A growing toolbox to image gene expression in single cells: sensitive approaches for demanding challenges. *Mol Cell* 71:468–480. <https://doi.org/10.1016/j.molcel.2018.07.022>

Part III

Application of Integrative Modelling and Analysis of Signalling Networks in Diseases



Chapter 15

Computational Random Mutagenesis to Investigate RAS Mutant Signaling

Edward C. Stites

Abstract

This chapter describes how mathematical models can be used to investigate the possible range of behaviors for mutant forms of a protein. A mathematical model of the RAS signaling network that has previously been developed and applied to specific RAS mutants will be adapted for the process of computational random mutagenesis. By using this model to computationally investigate the range of RAS signaling outputs that would be anticipated over a wide range of the relevant parameter space, one can gain intuition about the types of behaviors that would be demonstrated by biological RAS mutants.

Key words Computational biology, Systems biology, Mutation

1 Introduction

Mutations within the RAS genes *KRAS*, *NRAS*, and *HRAS* are among the most common acquired activating mutations that promote cancer [1]. Many different RAS mutant alleles have been observed in human cancer, and the specific alleles observed can vary largely between tumor type [2]. A variety of experiments suggest that there is significant selection for which specific mutations have the potential to promote cancer and suggest that this selection may vary between genetic and tissue contexts [3, 4]. The specific factors that influence which mutant alleles are capable of generating a tumor in a given tissue or genetic background remain undetermined.

A variety of methods have been applied to studying different RAS mutant alleles. Structures of a variety of mutant forms have been solved [5, 6]. Biochemical kinetics have been measured for a variety of RAS mutants [6, 7]. Isogenic cell lines that include different RAS alleles have helped illuminate cell biological differences [8–12]. Cell line models have also been useful for characterizing, at scale, a wide variety of RAS mutations expressed ectopically [13, 14]. Organoids that express different RAS alleles are an

important emerging approach, and such organoids have demonstrated the ability to reproduce mutant allele-specific effects including responses to targeted therapies [15]. Mouse models that compare different mutant alleles have also been useful for revealing differences in the ability to promote cancer [15, 16]. This chapter will focus on mathematical modeling as an alternative approach that can also enable exploration of the potential behaviors of mutant proteins.

The mathematical model is used to study how the biochemical consequences of a mutation influence biochemical phenotypes. When a protein is mutated, it may have an altered sequence of amino acids and/or altered expression levels. These changes can influence protein-protein interaction kinetics and/or the absolute abundance of the mutated protein. These quantifiable biological properties are also often properties of a mechanistic computational model. With such a model, computational exploration of how the model outputs change in response to parameter variation may be able to give insight into the behaviors that would be realized in an actual biological experiment (*See Note 1*).

2 Materials

2.1 RAS Model

Just as one would need to determine which mouse model or cell line model they would like to use for a specific experimental study, one must also determine which computational model to use for a theoretical study. As with experimental studies, one could either utilize an existing model system or one could generate a new model system. For the demonstration of this method, we will use an existing model of the RAS signaling network [17–19] (*See Note 2*). This model has proven useful for investigating a variety of problems involving RAS signaling, and this model has made a variety of prospective predictions about pathogenic RAS mutants that were nonobvious and were then experimentally observed [11, 12, 17, 20, 21]. The model is limited to RAS (separate pools of WT and mutant RAS), GAPs, GEFs, and effectors.

This RAS model used was developed at the same “resolution” as available biochemical characterizations of RAS proteins and their mutant forms. In other words, the parameters of the model are the same biochemical rate constants and enzymatic parameters that biochemists and biophysicists had measured for RAS GTPases. For example, GAP activity on RAS is modeled with Michaelis-Menten kinetics because experimentalists often measure and provide K_m and k_{cat} values. WT RAS and mutant RAS proteins can have different values for these rate constants because a reaction may proceed faster (or slower) for a mutant relative to the reaction speed for WT RAS. The outputs of the model are quantities like total RAS-GTP and total RAS-GTP-effector complex, which

generally and intuitively correspond to signaling outputs. The RAS model used here, and that we have used most commonly in the literature, has one pool of WT RAS, one pool of mutant RAS, and a single pool of each of GAP, GEF, and effector species.

2.2 Parameter Variation Plan for Computational Random Mutagenesis

Computational models generally have many different parameters. To perform a computational random mutagenesis, one must first determine which parameters are likely to vary with mutation (*See Note 3*). The RAS model being used in this example is limited to the reactions that directly regulate RAS. The model is subdivided into wild-type and mutant RAS pools (*See Note 4*). In this demonstration of a computational random mutagenesis, we will vary all parameters that involve modeled reactions that involve the mutant RAS. In this way, we will be attempting to explore the full range of behaviors that RAS mutants could potentially exhibit (*See Note 5*). For this demonstration, we will vary all parameters simultaneously. We will specify the values of each mutant parameter as a multiple of the wild-type parameter, where the scalar multiplying the wild-type parameter is a random number chosen from a log-normal distribution (*See Note 6*).

In addition to the parameters of the model that vary for the computational random mutagenesis, it should be predetermined whether there are other parameters that need to vary as part of the computational experiment. For example, in a previous study, we used computational random mutagenesis to evaluate the possible behaviors of RAS mutants when they occur in a cell with the full amount of GAP activity, and when they occur in a cell where the activity of a RAS GAP like NF1 has been lost to mutation [20]. This computational study involved us performing computational random mutagenesis on our baseline RAS model and on our baseline model with 50% of baseline GAP activity reduced to model the NF1/GAP impaired state. In another study, we evaluated the predicted drug-dose responses of cells containing computational random mutagenesis-generated RAS mutants [12]. In this more complicated computational experiment, we generated computational random mutants, filtered out only the mutants that were within a range of activation similar to the common oncogenic RAS mutants, and then we used the model to find predicted levels of RAS-GTP for different levels of pathway activation.

2.3 Software to Implement the Model and Hardware for Simulations

MATLAB software was used to develop and simulate the RAS model. Previous versions of the RAS model can be downloaded as part of previous publications [12, 22]. Implementation of the computational random mutagenesis can be implemented in MATLAB, as can data analysis. Simulations and analysis can be performed on a common laptop computer. Alternatively, one could code the same model into another programming language by utilizing the previously published supplemental methods and available code as references [11, 12, 17, 20, 22].

3 Methods

- 3.1 Specify the parameters that will vary as part of the computational random mutagenesis.
- 3.2 Specify the random number distribution. In our previous work, we used a log-normal distribution [20].
- 3.3 Generate the random parameters. In our previous instantiation, we generated a set of random numbers, where each random number in the vector of parameters for an individual computational random mutant was from a log-normal distribution.
- 3.4 Generate the mutant parameters. For each computational random mutant, multiply the WT parameter value by the next random numbers to obtain the computational random mutant.
- 3.5 Computationally find the output behavior of interest. We used steady-state levels of total RAS-GTP-effector complex as our metric of the signal strength that results from the computational mutant (*See Note 7*).
- 3.6 Plot results and evaluate the distribution of results.

4 Notes

1. Whether the model gives good insights for the problem will depend on the quality of the model. Although modeling biochemical networks has been an active area of research for approximately two decades, detailed studies of pathogenic mutations with these models have received much less attention. The best practices for modeling pathogenic mutations remain to be determined.
2. There are other available models of approximately similar scope and detail and that have been applied to the study of pathogenic mutants [23–27]. A comparison and contrast of some of these models and their application to modeling pathogenic mutants have previously been published [28]. Just as one should be well-versed in the pros and cons of an experimental system, one should also be well-versed in the pros and cons of the chosen computational model. For example, it is important to consider whether the question one is asking of a model is likely to be influenced by the simplifications of the model.
3. In a large pathway model, it will only be a subset of parameters that vary with mutation. However, in a large pathway model, each protein in the network may be modeled very simply, and there may therefore be very few parameters that involve the

mutated protein. In this case, the computational random mutagenesis is less likely to be informative with respect to the actual biological range of behaviors.

4. There will not already be separate pools of wild-type and mutant protein in most currently available computational models. In that case, one may model all of the protein as mutated, or one may develop a derivative version of the model that includes wild-type and mutant pools of protein.
5. One could envision a focused computational random mutagenesis, where mutations within one are of the protein are the focus, and where biophysical evidence suggests that the mutation would only disrupt specific protein-protein interactions and/or specific biochemical reactions. In such a situation, the computational random mutagenesis might focus on those parameters only and assume the mutations do not have any long-range interactions that influence other reactions of this protein.
6. One could perform computational random mutagenesis in a variety of ways. One might only sample small changes from baseline parameters (+/- 10%) or one might use larger changes (multiple orders of magnitude). Experimenters may search local mutation space (DNA and peptide sequence mutation searches that look at only single substitutions), and they may search large mutation space (all n-mers). Just as the experimental approaches may offer different benefits, so, too, may different computational random mutagenesis schemes.
7. RAS-GTP-effector may be the better measure of RAS activation than RAS-GTP when performing a computational random mutagenesis for RAS. This is because mutants that bind poorly to effectors, but are GAP insensitive, could result in high levels of RAS-GTP but still result in low levels of the RAS-GTP-effector complex and signal poorly.

Acknowledgments

This work was supported by NIH grant DP2 AT011327 and DoD grant W81XWH-20-10538.

References

1. Mendiratta G, Ke E, Aziz M, Liarakos D, Tong M, Stites EC (2021) Cancer gene mutation frequencies for the U.S. population. *Nat Commun* 12(1):5961. <https://doi.org/10.1038/s41467-021-26213-y>
2. Moore AR, Rosenberg SC, McCormick F, Malek S (2020) RAS-targeted therapies: is the undruggable drugged? *Nat Rev Drug Discov* 19(8):533–552. <https://doi.org/10.1038/s41573-020-0068-6>
3. Westcott PM, Halliwill KD, To MD, Rashid M, Rust AG, Keane TM, Delrosario R, Jen KY, Gurley KE, Kemp CJ, Fredlund E, Quigley DA, Adams DJ, Balmain A (2015) The mutational landscapes of genetic and chemical models of Kras-driven lung cancer. *Nature*

- 517(7535):489–492. <https://doi.org/10.1038/nature13898>
4. Pershing NL, Lampson BL, Belsky JA, Kaltenbrun E, MacAlpine DM, Counter CM (2015) Rare codons capacitate Kras-driven de novo tumorigenesis. *J Clin Invest* 125(1): 222–233. <https://doi.org/10.1172/JCI77627>
 5. Krengel U, Schlichting I, Scherer A, Schumann R, Frech M, John J, Kabsch W, Pai EF, Wittinghofer A (1990) Three-dimensional structures of H-ras p21 mutants: molecular basis for their inability to function as signal switch molecules. *Cell* 62(3):539–548. [https://doi.org/10.1016/0092-8674\(90\)90018-a](https://doi.org/10.1016/0092-8674(90)90018-a)
 6. Hunter JC, Manandhar A, Carrasco MA, Gurbani D, Gondi S, Westover KD (2015) Biochemical and structural analysis of common cancer-associated KRAS mutations. *Mol Cancer Res* 13(9):1325–1335. <https://doi.org/10.1158/1541-7786.MCR-15-0203>
 7. Gremer L, Merbitz-Zahradnik T, Dvorsky R, Cirstea IC, Kratz CP, Zenker M, Wittinghofer A, Ahmadian MR (2011) Germ-line KRAS mutations cause aberrant biochemical and physical properties leading to developmental disorders. *Hum Mutat* 32(1): 33–43. <https://doi.org/10.1002/humu.21377>
 8. De Roock W, Jonker DJ, Di Nicolantonio F, Sartore-Bianchi A, Tu D, Siena S, Lamba S, Arena S, Frattini M, Piessevaux H, Van Cutsem E, O'Callaghan CJ, Khambata-Ford S, Zalcberg JR, Simes J, Karapetis CS, Bardelli A, Teijpar S (2010) Association of KRAS p.G13D mutation with outcome in patients with chemotherapy-refractory metastatic colorectal cancer treated with cetuximab. *JAMA* 304(16):1812–1820. <https://doi.org/10.1001/jama.2010.1535>
 9. Hood FE, Klinger B, Newlaczyl AU, Sieber A, Dorel M, Oliver SP, Coulson JM, Bluthgen N, Prior IA (2019) Isoform-specific Ras signaling is growth factor dependent. *Mol Biol Cell* 30(9):1108–1117. <https://doi.org/10.1091/mbc.E18-10-0676>
 10. Mageean CJ, Griffiths JR, Smith DL, Clague MJ, Prior IA (2015) Absolute quantification of endogenous Ras isoform abundance. *PLoS One* 10(11):e0142674. <https://doi.org/10.1371/journal.pone.0142674>
 11. McFall T, Diedrich JK, Mengistu M, Littlechild SL, Paskvan KV, Sisk-Hackworth L, Moreesco JJ, Shaw AS, Stites EC (2019) A systems mechanism for KRAS mutant allele-specific responses to targeted therapy. *Sci Signal* 12(600). <https://doi.org/10.1126/scisignal.aaw8288>
 12. McFall T, Stites EC (2021) Identification of RAS mutant biomarkers for EGFR inhibitor sensitivity using a systems biochemical approach. *Cell Rep* 37(11):110096. <https://doi.org/10.1016/j.celrep.2021.110096>
 13. Bandaru P, Shah NH, Bhattacharyya M, Barton JP, Kondo Y, Cofsky JC, Gee CL, Chakraborty AK, Kortemme T, Ranganathan R, Kuriyan J (2017) Deconstruction of the Ras switching cycle through saturation mutagenesis. *eLife* 6. <https://doi.org/10.7554/eLife.27810>
 14. Ursu O, Neal JT, Shea E, Thakore PI, Jerby-Arnon L, Nguyen L, Dionne D, Diaz C, Bauman J, Mosaad MM, Fagre C, Lo A, McSharry M, Giacomelli AO, Ly SH, Rozenblatt-Rosen O, Hahn WC, Aguirre AJ, Berger AH, Regev A, Boehm JS (2022) Massively parallel phenotyping of coding variants in cancer with Perturb-seq. *Nat Biotechnol*. <https://doi.org/10.1038/s41587-021-01160-7>
 15. Zafra MP, Parsons MJ, Kim J, Alonso-Curbelo D, Goswami S, Schatoff EM, Han T, Katti A, Fernandez MTC, Wilkinson JE, Piskounova E, Dow LE (2020) An *in vivo* Kras Allelic series reveals distinct phenotypes of common oncogenic variants. *Cancer Discov* 10(11): 1654–1671. <https://doi.org/10.1158/2159-8290.CD-20-0442>
 16. Burd CE, Liu W, Huynh MV, Waqas MA, Gilligan JE, Clark KS, Fu K, Martin BL, Jeck WR, Souroullas GP, Darr DB, Zedek DC, Miley MJ, Baguley BC, Campbell SL, Sharpless NE (2014) Mutation-specific RAS oncogenicity explains NRAS codon 61 selection in melanoma. *Cancer Discov* 4(12):1418–1429. <https://doi.org/10.1158/2159-8290.CD-14-0729>
 17. Stites EC, Trampont PC, Ma Z, Ravichandran KS (2007) Network analysis of oncogenic Ras activation in cancer. *Science* 318(5849): 463–467. <https://doi.org/10.1126/science.1144642>
 18. Stites EC, Ravichandran KS (2012) Mathematical investigation of how oncogenic ras mutants promote ras signaling. *Methods Mol Biol* 880: 69–85. https://doi.org/10.1007/978-1-61779-833-7_5
 19. Stites EC (2021) Mathematical modeling to study KRAS mutant-specific responses to pathway inhibition. *Methods Mol Biol* 2262:311–321. https://doi.org/10.1007/978-1-0716-1190-6_19

20. Stites EC, Trampont PC, Haney LB, Walk SF, Ravichandran KS (2015) Cooperation between noncanonical Ras network mutations. *Cell Rep* 10(3):307–316. <https://doi.org/10.1016/j.celrep.2014.12.035>
21. Stites EC (2014) Differences in sensitivity to EGFR inhibitors could be explained by described biochemical differences between oncogenic Ras mutants. *bioRxiv*. <https://doi.org/10.1101/005397>
22. Stites EC, Shaw AS (2018) Quantitative systems pharmacology analysis of KRAS G12C covalent inhibitors. *CPT Pharmacometrics Syst Pharmacol* 7(5):342–351. <https://doi.org/10.1002/psp4.12291>
23. Donovan S, Shannon KM, Bollag G (2002) GTPase activating proteins: critical regulators of intracellular signaling. *Biochim Biophys Acta* 1602(1):23–45
24. Markevich NI, Moehren G, Demin OV, Kiyatkin A, Hoek JB, Kholodenko BN (2004) Signal processing at the Ras circuit: what shapes Ras activation patterns? *Syst Biol (Stevenage)* 1(1):104–113
25. Kiel C, Serrano L (2014) Structure-energy-based predictions and network modelling of RASopathy and cancer missense mutations. *Mol Syst Biol* 10:727. <https://doi.org/10.1002/msb.20145092>
26. Wolf J, Dronov S, Tobin F, Goryanin I (2007) The impact of the regulatory design on the response of epidermal growth factor receptor-mediated signal transduction towards oncogenic mutations. *FEBS J* 274(21):5505–5517. <https://doi.org/10.1111/j.1742-4658.2007.06066.x>
27. Wey M, Lee J, Jeong SS, Kim J, Heo J (2013) Kinetic mechanisms of mutation-dependent Harvey Ras activation and their relevance for the development of Costello syndrome. *Biochemistry* 52(47):8465–8479. <https://doi.org/10.1021/bi400679q>
28. Stites EC, Ravichandran KS (2012) Mechanistic modeling to investigate signaling by oncogenic Ras mutants. *Wiley Interdiscip Rev Syst Biol Med* 4(1):117–127. <https://doi.org/10.1002/wsbm.156>



Chapter 16

Mathematically Modeling the Effect of Endocrine and Cdk4/6 Inhibitor Therapies on Breast Cancer Cells

Wei He, Ayesha N. Shajahan-Haq, and William T. Baumann

Abstract

Mathematical modeling of cancer systems is beginning to be used to design better treatment regimens, especially in chemotherapy and radiotherapy. The effectiveness of mathematical modeling to inform treatment decisions and identify therapy protocols, some of which are highly nonintuitive, is because it enables the exploration of a huge number of therapeutic possibilities. Considering the immense cost of laboratory research and clinical trials, these nonintuitive therapy protocols would likely never be found by experimental approaches. While much of the work to date in this area has involved high-level models, which look simply at overall tumor growth or the interaction of resistant and sensitive cell types, mechanistic models that integrate molecular biology and pharmacology can contribute greatly to the discovery of better cancer treatment regimens. These mechanistic models are better able to account for the effect of drug interactions and the dynamics of therapy. The aim of this chapter is to demonstrate the use of ordinary differential equation-based mechanistic models to describe the dynamic interactions between the molecular signaling of breast cancer cells and two key clinical drugs. In particular, we illustrate the procedure for building a model of the response of MCF-7 cells to standard therapies used in the clinic. Such mathematical models can be used to explore the vast number of potential protocols to suggest better treatment approaches.

Key words Mathematical modeling, Breast cancer, MCF-7 cells, Endocrine therapy, Cdk4/6 inhibition, Palbociclib

1 Introduction

Mathematical models can contribute to cancer treatment by systematically simulating hundreds of thousands of possible therapy regimens to determine the most effective treatment possibilities. The effectiveness might be defined in many ways, from minimizing toxicity through limiting dosages and application windows to holding off resistance by not continuous targeting the same pathway and to maximally prolonging patient life span. In [1], the authors validate a dose-dense chemotherapy treatment schedule suggested from their own mathematical model prediction published in 1977 [2]. In their randomized clinical trial, this dose-dense treatment

improved disease-free survival (DFS) and overall survival (OS) of women with axillary node-positive breast cancer. The 4-year DFS is 82% for the dose-dense regimens and 75% for the other regimens. Their approach is considered the first mathematical model providing clinically validated predictions [3]. Another example is a model that uses evolutionary dynamics to show that adaptive treatment can be more effective than maximum tolerated dosage treatment in prostate cancer [4]. While these examples consider pharmacokinetics and drug interaction effects without modeling the mechanism of these drugs, they show how a mathematical model can be valuable for accomplishing simulation and optimization tasks, especially when testing the various drug effects is likely to be experimentally impractical based on the large number of variations in combinations, dosing and timing of these drugs [5–7].

A mechanism-based mathematical model can represent the numerous cellular system components, the dynamic relationships among them, and their interactions with different drugs, which give rise to the systems level behaviors observed during treatment. The components could include organs, tissues, cells, proteins, metabolites, RNA, and DNA. A mechanism-based model can easily look for synergism among a variety of drugs and suggest the best functional proteins or molecules to target in a signaling network. Incorporating experimental and clinical data into a mechanism-based mathematical model can enable it to predict the response to many different treatments, which cannot be efficiently tested one by one through experiments [8]. Although finding optimal dosing frequencies will benefit clinical treatment [9, 10], it is often impractical or unethical to test different kinds of combination and dosing strategies in preclinical and clinical settings.

The aim of this chapter is to provide an example of building a mechanism-based ODE (ordinary differential equation) model of the dynamic interactions between proteins and drugs in breast cancer cells [11]. The model uses known mechanisms from the literature to describe the changes in protein level and proliferation of an asynchronous population of MCF-7 breast cancer cells in response to various clinically relevant therapies. Although mathematical models have been applied in various types of cancers [12–15], there is no mechanism-based mathematical model for estrogen receptor positive/human epidermal growth factor receptor 2 negative (ER+/HER2-) luminal A subtype breast cancer cells represented by the MCF-7 cell line.

There are four main molecular subtypes of breast cancer cells, and they differ markedly with respect to therapeutic treatment methodology. The four main subtypes are distinguished by clinical evaluation of biomarkers for the ER, PR (progesterone receptor), HER2, and Ki67, which is a marker for proliferation. The luminal A subtype is the most common subtype of breast cancer and is characterized by being hormone receptor positive (HR+; meaning

either ER+ or ER+ and PR+) and HER2-. The luminal A has low levels of Ki67, indicating slower cell growth and is associated with the best prognosis [16]. Estrogens, which are steroid hormones with the most potent estrogen being 17 β -estradiol (E2), are the regulators of ER [17]. The main effect of E2 is to regulate growth and proliferation [18, 19]. It binds to nuclear receptor ER α , which is the major type of ER commonly overexpressed in ER+ breast cancer cells [20] and the major oncogene responsible for E2-induced enhancement of cell proliferation [21]. E2 binds to ER α in the cytoplasm, producing conformational changes that result in homodimerization, translocation to the nucleus, and binding to estrogen response elements (ERE) in the promoter regions of estrogen-responsive genes to regulate transcription [21].

Because ER is a key to ER+ breast cancer cell growth and proliferation, endocrine therapy, which interferes with E2 and ER signaling, is extensively used to treat the ER+ clinical subtype [22, 23]. Endocrine therapy is a type of targeted therapy that can dramatically improve long-term survival rates and avoid the toxicity of chemotherapies. First type of endocrine therapy involves decreasing E2 and ER signaling by depriving ER of its ligand E2 using an aromatase inhibitor (AI). AIs, like letrozole or anastrozole, lower the E2 level by inhibiting aromatase, which is an enzyme responsible for a key step in the synthesis of E2 [24]. Second type of endocrine therapy involves competitively inhibiting the binding of E2 to ER (the mode of action of tamoxifen) [23]. Third type of endocrine therapy involves increasing the degradation of ER using ICI 182,780 (ICI; Faslodex/Fulvestrant). This drug binds to the ER, blocking E2 binding and causing a proteasome-dependent degradation of the receptor [25]. In addition to endocrine therapy, which had been the standard of care for patients with ER+ cancer since the 1970s, selective cyclin-dependent kinase 4 and 6 (Cdk4/6) inhibitors have become significant for treatment of ER+/HER2-breast cancer in recent years. Adding Cdk4/6 inhibitors to endocrine therapy is now the first-line treatment based on the substantial improvement in survival outcomes from clinical trials [25]. Palbociclib is one of the Cdk4/6 inhibitors and was initially approved in 2015 by the FDA to be used in combination with the AI letrozole. Palbociclib has low IC₅₀ (half maximal inhibitory concentration) values in the nanomolar range and has become as essential in ER+/HER2- breast cancer patient treatment as endocrine therapy.

To create a simulation platform for exploring therapeutic protocols, we built an ODE model of MCF-7 cells' changes in protein levels and proliferation in response to endocrine therapy and the Cdk4/6 inhibitor palbociclib. The model was calibrated using experiments on MCF7 cells in culture. Because there is no aromatase in the culture system, E2 deprivation ($-E_2$) is used as a surrogate for treatment with aromatase inhibitors [26]. The three therapies considered by the model were $-E_2$, ICI, and palbociclib.

2 Material

MCF-7 cells were grown in phenol red-free medium with 10% charcoal-stripped calf serum and supplemented with 10 nM E2. – E2 was obtained by washing cells 24 h post-plating ($t = 0$) with PBS (phosphate-buffered saline) and adding complete medium without E2 for the indicated times. Cell number was counted using a Coulter Counter. The following antibodies were used to measure protein levels using Western blot analysis: c-Myc, cyclinD1, RB1, ESR α , RB1-phosphorylated on Ser612, actin, and β -tubulin.

3 Methods

3.1 Signaling Wiring Diagram and Model Structure

The first step in creating a model is to determine the key pathways to be modeled and create a wiring diagram for the interactions of the components of the model. Because of the enormous complexity of biological systems, the models are almost always vast simplifications of the actual system, and the modeler must choose a subset of the potential components to retain in the model. The components and mechanisms retained in our model are based on major, known mechanisms from the literature [21]. For this project, the most important aspect of the E2/ER signaling pathway is that it regulates the G1-S transition of the cell cycle.

The interactions of the major signaling pathway elements are shown in Fig. 1. The meaning and justification for our choices of each of the numbered interactions in the figure are as follows: (1) – E2 decreases the level of estrogen in the medium [26]. (2) E2 binds to ER and forms the transcription factor E2:ER [20]. (3) ICI binds with ER and increases the degradation of ER and blocks its activity [25]. (4) E2:ER transcriptionally upregulates c-Myc [27]. (5) E2:ER transcriptionally upregulates cyclinD1 [27]. (6) c-Myc represses transcription of p21 [28]. (7) CyclinE binds to Cdk2 and forms the cyclinE:Cdk2 kinase [27]. (8) p21 inhibits cyclinE:Cdk2 kinase activity [29]. (9) CyclinD1 binds to Cdk4/6 and forms the cyclinD1:Cdk4/6 kinase [30]. (10) p21 inhibits cyclinD1:Cdk4/6 kinase activity [31]. (11) Palbociclib binds to Cdk4/6 and inactivates it [25]. (12) RB1 binds to E2F and inactivates it [28]. (13) CyclinD1:Cdk4/6 phosphorylates RB1 [27]. (14) RB1-p (hypophosphorylated RB1) binds to E2F and inactivates it [28]. (15) CyclinE:Cdk2 phosphorylates RB1-p [27]. (16) free E2F enhances production of c-Myc [32]. (17) c-Myc enhances cyclinE:Cdk2 kinase activity [33]. (18) free E2F enhances production of RB1 [33]. (19) CyclinD1:Cdk4/6 phosphorylates p107 and increases its degradation [33, 34]. (20) CyclinD1:Cdk4/6 phosphorylates

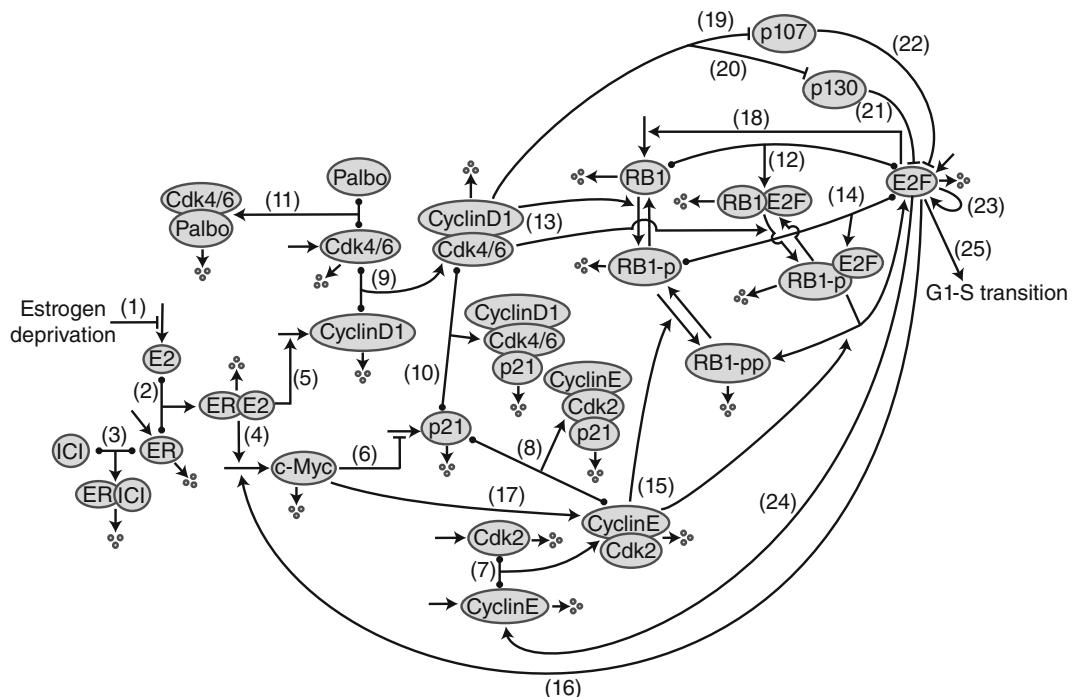


Fig. 1 Wiring diagram of the biological mechanism. The lines with arrowheads represent enhancement and blunt heads represent inhibition. (Reproduced from ref. 9 with permission from the Royal Society)

p130 and increases its degradation [33, 35]. (21) p130 binds with E2F and inhibits its transcriptional activity [33]. (22) p107 binds with E2F and inhibits its transcriptional activity [33]. (23) Free E2F enhances its self-expression [33]. (24) Free E2F enhances production of cyclinE [36]. (25) Free E2F drives the G1-S transition and proliferation [37]. The arrows pointing to proteins represent the production of the proteins, and the arrows pointing to three dots from the proteins represent the degradation of the proteins.

We note that some of the interactions in Fig. 1 are actual reactions, such as the binding and unbinding of E2 and ER in [2], while others are indirect interactions in which we have chosen not to model the intermediate steps, such as c-Myc enhancing the activity of cyclinE:Cdk2 in [17]. Even with the simplification inherent in Fig. 1, the practical limits on the experimental data we could collect forced us to make further simplifications to the model. The simplified wiring diagram on which the mathematical model is based is shown in Fig. 2.

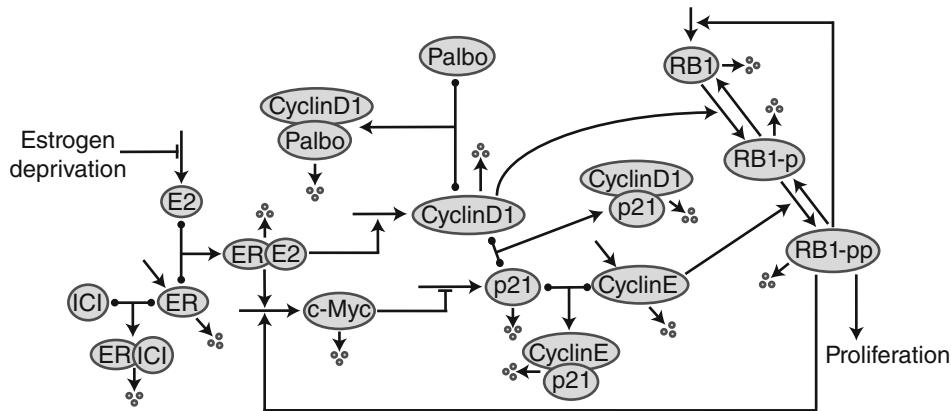


Fig. 2 Wiring diagram of the mathematical model. The meaning of the lines is the same as Fig. 1. (Reproduced from ref. 9 with permission from the Royal Society)

3.2 Model Implementation and Parameter Estimation

The mathematical model consists 14 ODE equations and has 62 parameters. It is implemented in MATLAB. The ODEs are solved numerically using the `ode23tb` function. Degradation rates are set according to the half-lives from the literature. The other parameters are optimized to minimize the difference between the model simulation and the experimental results. The cost function is

$$M(p) = \sum_{i=1}^n \sum_{j=1}^m \frac{(y_{ij}^E(t_j) - y_{ij}(t_j, p))^2}{\sigma_{ij}^2}$$

where i indexes the measured quantities (the protein level or proliferation), j indexes the time points, $y_{ij}^E(t_j)$ is the experimental measurement of the i th variable at time j , $y_{ij}(t_j, p)$ is the simulation result from the model of the i th measurement at time j using parameter vector p , and σ_{ij} is the standard deviation of the experimental results based on three replicates. Parameter estimation is performed using MATLAB. The default genetic algorithm function, `ga`, in the global optimization toolbox as well as the `fminsearch` function are used.

3.3 Local Sensitivity Analysis

We use the local sensitivity to evaluate the sensitivity of the model outputs to changes in the input parameters [38]. The local sensitivity of cell proliferation at day 7 is calculated with respect to the parameters [39].

$$s_i = \frac{\partial \log(X)}{\partial \log(P_i)} = \frac{\partial X}{\partial P_i} \frac{P_i}{X}$$

where s_i is the local sensitivity value, which equals the derivative of output X with respect to parameter P_i multiplied by the ratio P_i/X . It is the relative change in the cell proliferation induced by a small relative change in parameter i . s_i is approximated by the second-order central finite difference. Each parameter is individually varied by $\pm 5\%$ of its value. Therefore,

$$\begin{aligned}s_i &\approx \frac{X(P_i + 5\% \times P_i) - X(P_i - 5\% \times P_i)}{10\% \times P_i} \frac{P_i}{X(P_i)} \\&= \frac{X(P_i + 5\% \times P_i) - X(P_i - 5\% \times P_i)}{10\% \times X(P_i)}\end{aligned}$$

All parameter sets in the parameter cohort are used for the local sensitivity analysis, and the final sensitivity value is the averaged value over the parameter cohort.

4 Notes

4.1 Results

After calibrating the model using experimental data, we show that the model can not only recapitulate the experimental data on which it was trained (monotherapy data) but can also predict the responses to combination therapies on which it was not trained. Figure 3a shows the experimental and simulation results over 7 days in response to $-E2$. The increase in ER after $-E2$ treatment is because the half-life of E2:ER is lower than that of unbound ER. Thus, $-E2$ treatment stabilizes the ER, causing its level to increase [40]. The jump in total ER level at day 3 is due to changing the medium on this day, which decreases the concentration of E2 in medium. The initial decrease in the c-Myc level after $-E2$ treatment is because E2:ER is a transcription factor for c-Myc. c-Myc also decreases at day 3 and continues to decrease due to changes in the E2 concentration on day 3 and the gradual decrease of RB1-pp, which can drive the production of c-Myc. RB1-pp steadily decreases because of the decrease of cyclinD1:Cdk4/6 and cyclinE:Cdk2 kinase activity. The small decrease in cyclinD1 is because E2:ER is a transcription factor for cyclinD1. The decrease in total RB1 is captured through the presumed transcriptional effect of RB1-pp on RB1.

Figure 3b shows the experimental and simulation results over 7 days in response to $+E2 + ICI$ (500 nM). The decrease in the ER level after $+E2 + ICI$ (500 nM) treatment is because ICI binds to ER and enhances ER degradation. The reduction of E2:ER, due to both ICI displacing E2 and enhancing its degradation, causes changes in c-Myc, cyclinD1, RB1-pp, and total RB1 similar to $-E2$ treatment. Figure 3c shows the experimental and simulation results over 7 days in response to $+E2 + palbo$ (1 μ M) treatment. Palbociclib inhibits Cdk4/6 kinase activity and leads to reduced RB1-pp, which in turn causes the c-Myc level to decrease.

Figure 4 shows the experimental and simulated cell proliferation results for the $+E2$, $-E2$, $+E2 + ICI$ (500 nM), and $+E2 + palbo$ (1 μ M) cases. It validates that proliferation can be adequately modeled using the RB1-pp level. To further validate the model, its ability to predict the effect of combination therapies on protein levels and cell proliferation is tested. Figure 5a shows the

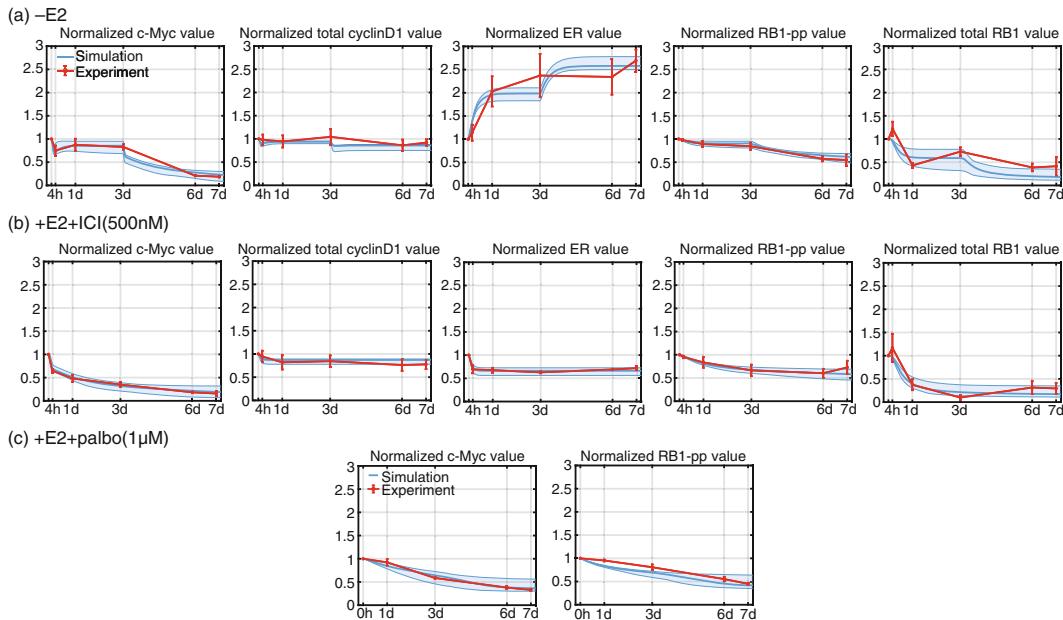


Fig. 3 Simulation and experimental results for protein-level changes under different conditions (h, hour; d, day). Red represents the experiment results (mean \pm s.e., $n = 3$), and cyan represents the simulation results. E2 concentration is 10 nM. The solid line represents the simulation results with lowest cost value. The shaded region contains 98% of the cohort simulations. (Reproduced from ref. 9 with permission from the Royal Society)

model prediction of $-E2 + ICI$ (500 nM) treatment compared to the experimental results. The combination of reducing the supply of both E2 and ER causes larger changes in the protein levels than either $-E2$ or $+E2 + ICI$ (500 nM) treatment alone. Figure 5b shows the model prediction of $-E2 + palbo$ (1 μ M) compared to the experimental results. The combination of $-E2$ and palbociclib inhibits cyclinD1:Cdk4/6 by both reducing the cyclinD1 level and inactivating Cdk4/6. It causes larger reductions of c-Myc and RB1-pp than either $-E2$ or $+E2 + palbo$ (1 μ M) treatment alone. The predictions match the experimental results for both the $-E2 + ICI$ (500 nM) and $-E2 + palbo$ (1 μ M) treatments quite well. Figure 5c shows that the model can also predict the proliferation changes in response to the two $-E2 + ICI$ (500 nM) and $-E2 + palbo$ (1 μ M) combination therapies. The low proliferation follows from the fact that RB1-pp decreases more in response to the combination of $-E2$ and ICI or palbociclib than to $-E2$, ICI, or palbociclib treatment alone.

4.2 Discussion

The wiring diagram of a mechanistic mathematical model should incorporate the targets of the proposed treatments and the major signaling pathways affected by these targets. The structure of our model is based on signaling pathways related to the G1-S transition

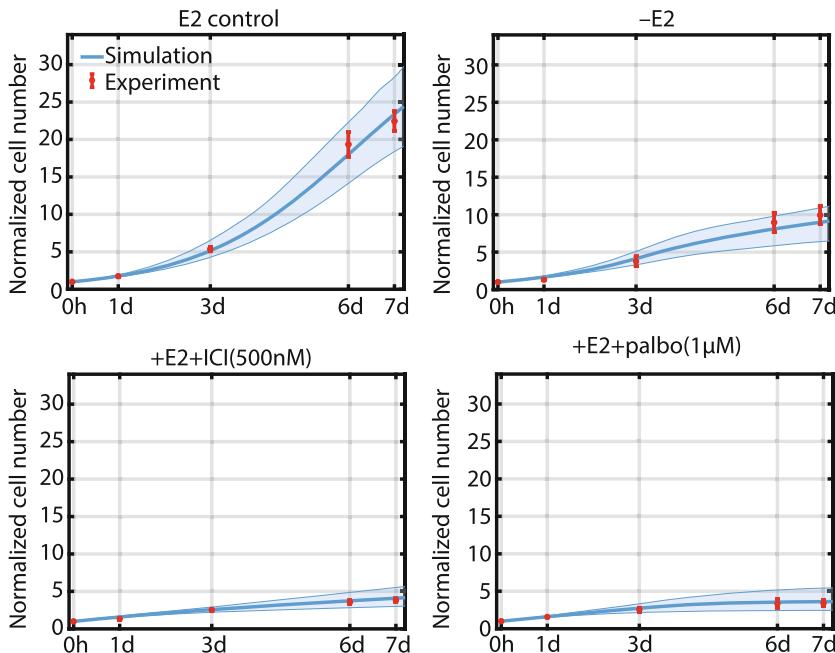


Fig. 4 Simulation and experimental results for normalized cell numbers under different treatment conditions. Red represents the experimental counts (mean \pm s.e., $n = 3$), and cyan represents the simulation results. E2 concentration is 10 nM. The meaning of the solid line and the shaded region are the same as Fig. 3. (Reproduced from ref. 9 with permission from the Royal Society)

of the cell cycle as shown in Figs. 1 and 2. This choice is because the most important E2:ER response genes regulate progression through the G1-S checkpoint of the cell cycle [21]. Inhibition of ER signaling halts cell proliferation by arresting the cell cycle in the G1 phase [41]. The two endocrine therapies we considered, $-E2$ and $+E2 + ICI$, impede the G1 to S transition and arrest the cells in a state with characteristics of quiescence [42]. Therefore, when constructing this mechanistic model, we kept the most important signaling networks impacted by our treatments and ignored others of less importance. Thus, we kept the classical ER signaling pathway but ignored the following three nonclassical pathways: (1) E2:ER can bind to other transcription factors by protein-protein interaction in the nucleus [17]. (2) Growth factors, such as insulin-like growth factor (IGF) and epidermal growth factor (EGF), can activate RTKs, which in turn can activate extracellular signaling regulated kinase (ERK) and protein kinase B (AKT). After that, ER itself can be phosphorylated by these serine/threonine kinases leading to estrogen-independent activation of ER [17]. (3) Estrogen can also bind to membrane-bound ER, resulting in assembly of a protein complex that activates protein kinase cascades leading to transcription factor activation [21]. All of these nonclassical ER signaling mechanisms were ignored as we think the major treatment effects depend on the classical ER signaling mechanism.

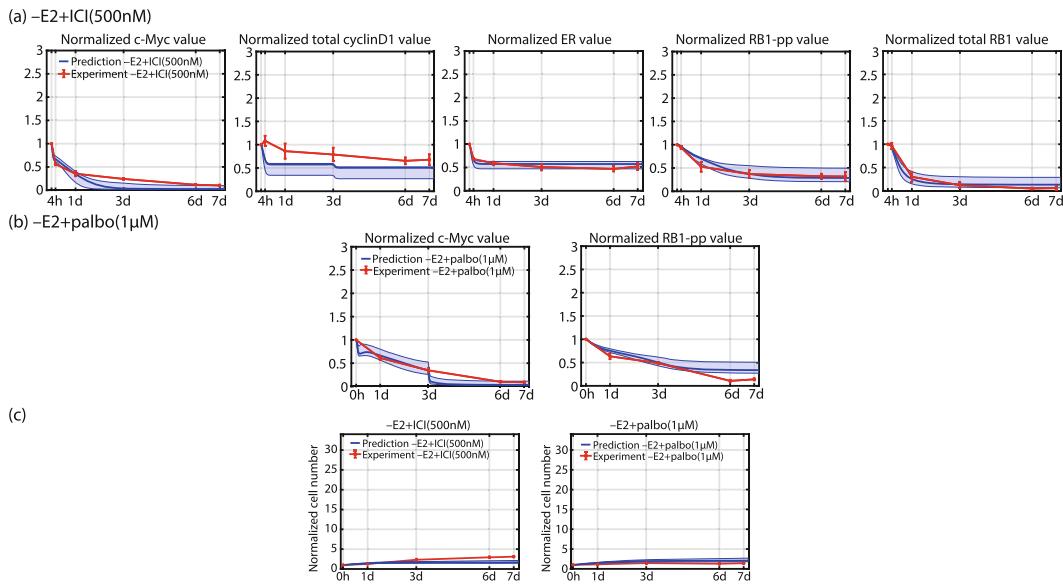


Fig. 5 Prediction and experimental results for combination therapies. Red represents the experimental results ($\text{mean} \pm \text{s.e.}, n = 3$), and blue represents the prediction results. The meaning of the solid line and the shaded region are the same as Fig. 3. (a) Prediction of protein-level changes in response to combination $-E2 + ICI$ (500 nM) treatment. (b) Prediction of c-Myc and RB1-pp levels in response to $-E2 + \text{palbo}$ (1 μM) treatment. (c) Prediction of proliferation in response to $-E2 + ICI$ (500 nM) and $-E2 + \text{palbo}$ (1 μM) treatment. (Reproduced from ref. 9 with permission from the Royal Society)

In classical ER signaling, the most important effects are enhanced transcriptional expression of regulatory proteins c-Myc and cyclinD1 [27]. Ectopic expression of either c-Myc or cyclinD1 is sufficient to induce S phase entry in MCF-7 cells previously arrested in G1 phase by ICI treatment [27]. Therefore, c-Myc and cyclinD1 are key proteins that should be included in the model and measured experimentally for model parameter calibration.

In addition to c-Myc and cyclinD1, ER is another important protein needed to be included in the model. The half-lives of E2-bound ER and unbound ER are different. $-E2$ treatment will create more unbound ER, which has a longer half-life compared with E2-bound ER, so the total ER level will increase. On the other hand, when ICI binds to ER, it causes enhanced degradation, so the total ER level will decrease. Therefore, ER is included in the model, and the total ER level is measured experimentally for model parameter calibration. The endocrine therapies $-E2$ and $+E2 + ICI$ both cause decreases in c-Myc and cyclinD1, which in turn cause decreases in RB1-pp and total RB1, which are also measured experimentally as they are critical to the G1-S transition.

Proliferation is the most important endpoint for treatment decisions, so it is important for the model to be able to simulate proliferation changes with different treatments. Since the major

treatment effect is G1 arrest and a cell passing the G1-S transition will finish its cell cycle and divide, E2F, which is the key and last driver of the G1-S transition, would be the best candidate to determine the proliferation rate. The E2F transcription factor plays a central role in regulating the expression of genes involved in the G1-S transition and DNA synthesis [37]. Key E2F target genes regulating the G1-S transition include cyclinD1, cyclinE, cyclinA, and Cdc25A [37]. E2F target genes related to DNA synthesis include DNA polymerase α , dihydrofolate reductase (DHFR), thymidine kinase, Cdc6, and minichromosome maintenance (MCM) proteins [37]. However, measuring the transcriptional activity of E2F or the free E2F level is quite difficult.

In mammalian cells, six E2F family members have been recognized (E2F1–6) [37]. There is a clear divergence of function for E2F family members: E2F1–3 function mainly as transcriptional activators and E2F4–5 function mainly as repressors [43]. The role of E2F6 is different from the other E2F members and has not been thoroughly investigated [43]. E2F1–5 can bind to different pocket protein members (RB1, p107, and p130) and show different functionality in G1 and the G1-S transition [43]. Due to the complexity of the different possible combinations of E2F family members and pocket proteins, it is hard to use one protein level to represent E2F transcriptional activity. Therefore, we decided to use the phosphorylation status of RB1 to represent E2F transcriptional activity.

As mentioned above, RB1 is one of three members in the pocket protein family, and E2F is regulated principally through its temporal association with RB1. E2F1–3 functions mainly as transcriptional activators and binds only to RB1 [44]. CyclinD1:Cdk4/6 phosphorylates RB1 to a hypophosphorylated form (RB1-p), and then cyclinE:Cdk2 phosphorylates RB1-p to a hyperphosphorylated form (RB1-pp) [27]. RB1-pp is fully inactivated and releases E2F transcription factors that transactivate the genes required for G1 to S transition and DNA synthesis [28]. It is possible to use a single specific phosphorylation site to reflect the hyperphosphorylated state of RB1, which we then use to reflect the E2F transcriptional activity. There are over 15 phosphorylation sites found on RB1, including T5, S249, S252, T356, T373, S567, S608, S612, S780, S788, S795, S807, S811, T821, and T826 [44]. Based on the literature [44, 45], in late G1, cyclinE:Cdk2 phosphorylates RB1 on S612 and T373 relieving RB1's repression of E2F activity. We tested these two specific phosphorylation sites for whether they reflect the hypothesized decrease in RB1-pp level after –E2 and +E2 + ICI treatments. According to our experimental results, we chose RB1 phosphorylated on S612 to represent the hyperphosphorylated status of RB1 and thus the transcriptional activity of E2F.

The idea of using one specific phosphorylation site to represent the hyperphosphorylated status of RB1 is not uncommon. In [46], they demonstrate that RB1 exists mainly in unphosphorylated, monophosphorylated (hypophosphorylated), or hyperphosphorylated form. There is no evidence of partially phosphorylated RB1 under unperturbed conditions. In [46], they also suggested that measuring a specific phosphorylation site on RB1 can be exploited to infer the hyperphosphorylated status of RB1. They reasoned that most of the hyperphosphorylated RB1 has each specific phosphorylation site phosphorylated on most copies of RB1. The monophosphorylated (hypophosphorylated) RB1, however, has the same site phosphorylated on only a small fraction of the RB1. Based on their experimental results [46], they used S807/S811 to reflect the hyperphosphorylation status of RB1. Although their specific phosphorylation site is different from ours, the idea of using an antibody against a specific phosphorylation site on RB1 to reflect its hyperphosphorylation status is the same. The best specific phosphorylation site might well be cell and treatment dependent.

Because we used RB1 phosphorylated on Ser612 to represent the hyperphosphorylated RB1 and the free E2F, we also measured total RB1 level to check whether it changes under treatment. In total, we measured five key proteins: total ER, c-Myc, cyclinD1, RB1-phosphorylated on Ser612, and total RB1. As shown in Fig. 2, we included the E2:ER transcriptional activity on cyclinD1 and c-Myc. Although c-Myc has been reported to transcriptionally upregulate cyclinD1 [47], when tested on MCF-7 cells, induction of c-Myc fails to increase cyclinD1 expression, and induction of cyclinD1 has no effect on c-Myc expression. Therefore, the increased transcription of cyclinD1 and c-Myc is treated as a direct effect of E2:ER in the model. The model also includes one of the major effects of c-Myc, the transcriptional repression of the cyclin-dependent kinase inhibitor p21, to link c-Myc to its downstream effects. We then included p21 binding to the cyclinD1:Cdk4/6 and cyclinE:Cdk2 complexes to inhibit their kinase activity [29, 31].

We also added the most important signaling events that control the RB1-pp level: the kinase activity of cyclinD1:Cdk4/6, which phosphorylates unphosphorylated RB1 to create RB1-p, and the kinase activity of cyclinE:Cdk2, which phosphorylates RB1-p to RB1-pp. We also included a transcriptional activity of RB1-pp on RB1 itself, reflecting the transcriptional activity of free E2F on RB1 [33]. Adding this pathway allows the model to recapitulate the experimental changes in total RB1 level in response to endocrine therapies. As the RB1-pp level decreases after treatment, the total RB1 level will also decrease. The last signaling pathway included is the transcriptional activity of RB1-pp on c-Myc, which reflects the transcriptional activity of free E2F on c-Myc [32]. From the experimental results, we see that, unlike the changes of cyclinD1 in response to treatment where the level initially drops and then

remains constant, the c-Myc level drops at the beginning of the treatment and keeps decreasing afterward. This behavior implies that there exists a feedback mechanism downstream of c-Myc signaling that controls the expression of c-Myc. The E2F transcriptional activity on c-Myc is the best candidate among the G1 signaling pathways, and adding it to the model allows the model to recapitulate the c-Myc level changes in response to different treatments.

Another simplification for the model is that we did not model Cdk4/6 explicitly but assume all cyclinD1 not bound to p21 is bound to Cdk4/6 and is active. This choice reduces the number of species modeled so as to be more in line with the number of species measured. It also decreases the number of parameters to be estimated. For the same reasons, there are signaling mechanisms shown in Fig. 1 that are not included in Fig. 2. For example, cyclinE is a well-established target gene of E2F, forming a positive feedback loop, number [24] in Fig. 1; c-Myc can activate cyclinE:Cdk2 kinase activity via Cdc25A, number [17] in Fig. 1; E2F is a target gene of itself, number [23]; other pocket proteins (p107, p130) can bind and inhibit E2F activity, number [21] and [22] in Fig. 1; cyclinD1:Cdk4/6 phosphorylates other pocket proteins (p107, p130) and enhances their degradation, number [19] and [20] in Fig. 1. Had we been unable to reasonably fit the experimental data with our simplified model; we would have considered adding one or more relevant pathways back into the model.

As the final point for constructing the signaling wiring diagram of the model, it is important to note that the changes in protein levels or proliferation are the result of thousands of signaling interactions among DNA, RNA, proteins, and metabolites. It is currently impossible to model this level of complexity, so we simplified the model and included only those mechanisms necessary to capture the key effects of the therapies. While it is enticing to include all the well-known mechanisms around the G1 to S transition, the limited data we are able to obtain does not warrant the increase in parameters, which would be practically unidentifiable.

After the model was able to recapitulate the key protein and proliferation changes in response to endocrine therapy, it became important to add the capability to recapitulate the treatment effects of Cdk4/6 inhibitors, as these inhibitors represent an appealing therapeutic strategy for treatment of ER+/HER2- breast cancer [25]. To date, there are three Cdk4/6 inhibitors approved by the US FDA (Food and Drug Administration): palbociclib (PD0332991; Ibrance, Pfizer, United States), ribociclib (LEE011; Kisqali, Novartis, Switzerland), and abemaciclib (LY2835219, Verzenio, Lilly, United States). Adding a Cdk4/6 inhibitor into the model involves issues of ease of incorporation as well as usefulness of the specific inhibitor. Palbociclib and ribociclib have similar potencies: the reported IC₅₀s for palbociclib are 11 nM

and 15 nM, and for ribociclib are 10 nM and 39 nM for Cdk4 and Cdk6, respectively [48]. Considering the average potency of Cdk4/6 inhibition, palbociclib is a little better than ribociclib. Abemaciclib has the most potent IC₅₀s at 2 nM and 10 nM, respectively [48]. However, the transcriptional and proteomic changes caused by abemaciclib differ significantly from palbociclib and ribociclib, and abemaciclib is the least selective of the three [49]. Abemaciclib has additional activities against cyclinE:Cdk2, cyclinB1:Cdk1, cyclinH:Cdk7, cyclinK:Cdk9, CAMKIIα/β/γ (Ca²⁺/calmodulin-dependent protein kinase II α/β/γ), GSK-3α/β (glycogen synthase kinase 3α/β), DYRK (dual specificity tyrosine phosphorylation regulated kinase)/HIPK (homeodomain interacting protein kinase), AURKA/B (Aurora kinase A/B), and PLK1 (polo-like kinase 1) [49]. Most importantly, palbociclib only causes cell cycle arrest in the G1 phase, with little or no apoptosis at lower doses, whereas abemaciclib can arrest the cell in both the G1 and G2 phases of the cell cycle and also elicit apoptosis at doses greater than 0.3 μM [49]. Adding abemaciclib to the model likely involves a major extension of the model, so we settled on adding palbociclib. We hypothesized that if the current model was accurately capturing the key driving mechanisms, adding a new drug affecting these mechanisms should require calibrating only a few new parameters associated with the new drug's interactions with Cdk4/6. Such a calibration should not require measuring the responses of all the proteins in the model. So, we only measured RB1-pp, since it controls proliferation, and c-Myc, since we expect it to gradually decrease after palbociclib treatment due to feedback from RB1-pp. Using these two measurements, it was straightforward to get the gross responses to palbociclib treatment correct by calibrating only the new parameters associated with palbociclib and leaving the previously calibrated parameters alone.

One question worthy of explanation is the experimentally measured decrease in total RB1 level after -E2 or +E2 + ICI treatment. This decrease in total RB1 has been observed by others in response to E2 deprivation [50]. Less total RB1 might be expected to create more free E2F and increase the G1-S transition and proliferation rather than decrease proliferation as is observed experimentally. However, because E2F is a transcription factor of RB1, the decreased transcriptional activity of E2F resulting from the decreased level of RB1-pp can reduce the total RB1 level. Second, the decreased E2F transcriptional activity can also be from the increase of another pocket protein, p130. As cyclinD1:Cdk4/6 can phosphorylate p130 and induce its degradation, the decrease of cyclinD1:Cdk4/6 kinase activity after endocrine therapy can cause the increase of p130 [33, 35]. In [50], after the treatment of MCF-7 cells with ICI, p130 increased fourfold and formed a p130:E2F4 complex and accumulated hyperphosphorylated E2F4, which is a marker of cellular quiescence. Both E2F4

and E2F5 can form a complex with p130 and act as repressors of E2F transcription [43]. And the decrease of transcriptional activity of E2F will be further reinforced as E2F is also a transcriptional factor for itself [33]. Once the E2F transcriptional activity decreases, it will keep lowering the transcription of itself. Therefore, the decrease of total RB1 level does not conflict with the decrease of E2F level and its transcriptional activity.

Another point that needs to be mentioned is the E2 deprivation procedure, which is conducted by exchanging medium with 10 nM of E2 added for medium with no added E2. Both media contain 10% charcoal stripped calf serum (CCS), which contributes very little E2 as the E2 level in CCS is less than 4pM [51]. After the exchange, however, the E2 in the cell, which is at a significantly higher concentration than that in the +E2 medium, can diffuse back into the medium and cause an increase in the E2 concentration. So, each time the -E2 medium is replenished, the residual E2 level will decrease further. In our 7-day experiments, the medium is changed on day 0 and day 3. And we incorporate two additional model parameters, one for the E2 concentration after day 0 and another for the concentration after day 3. These parameters are used to account for the fact that the E2 concentration decreases with additional changes in the medium.

Another point that needs to be emphasized concerns the prediction results for -E2 + ICI treatment. While predicting that a combination therapy will have greater effect than either therapy by itself is not surprising, there is one surprising prediction made by the model, namely, that ER for the combination decreases below the level for ICI treatment alone. Since -E2 alone causes a significant increase in ER and ICI alone causes a significant decrease, the ER level for the combination therapy is not obvious. Only a quantitative model can resolve the outcome. The reason the model predicts a stronger decrease in ER level is that the scarcity of E2 allows more ICI to bind to ER, causing greater degradation than occurs when ICI competes with E2 for binding to ER.

The last points that need to be mentioned concern the cohort of parameter sets we used and the local sensitivity of the parameters. We identify a total of 400 parameter sets that reasonably fit the experimental data and refer to this set as a parameter cohort. Each parameter set in the cohort has a cost value less than 300 as shown in Fig. 6a (right), and also each single parameter in the cohort has a reasonable coefficient of variation as shown in Fig. 6a (left), which means each single parameter spreads over a reasonable range for the parameter. When using the model for prediction, we simulated all 400 parameter sets. The idea is that if all of the parameter sets that are well justified by the data give a comparable prediction (little spread), the prediction is justified based on the experimental data used to calibrate the model. Figure 6b shows the local sensitivity of cell number to each of the parameters. This computation is used to

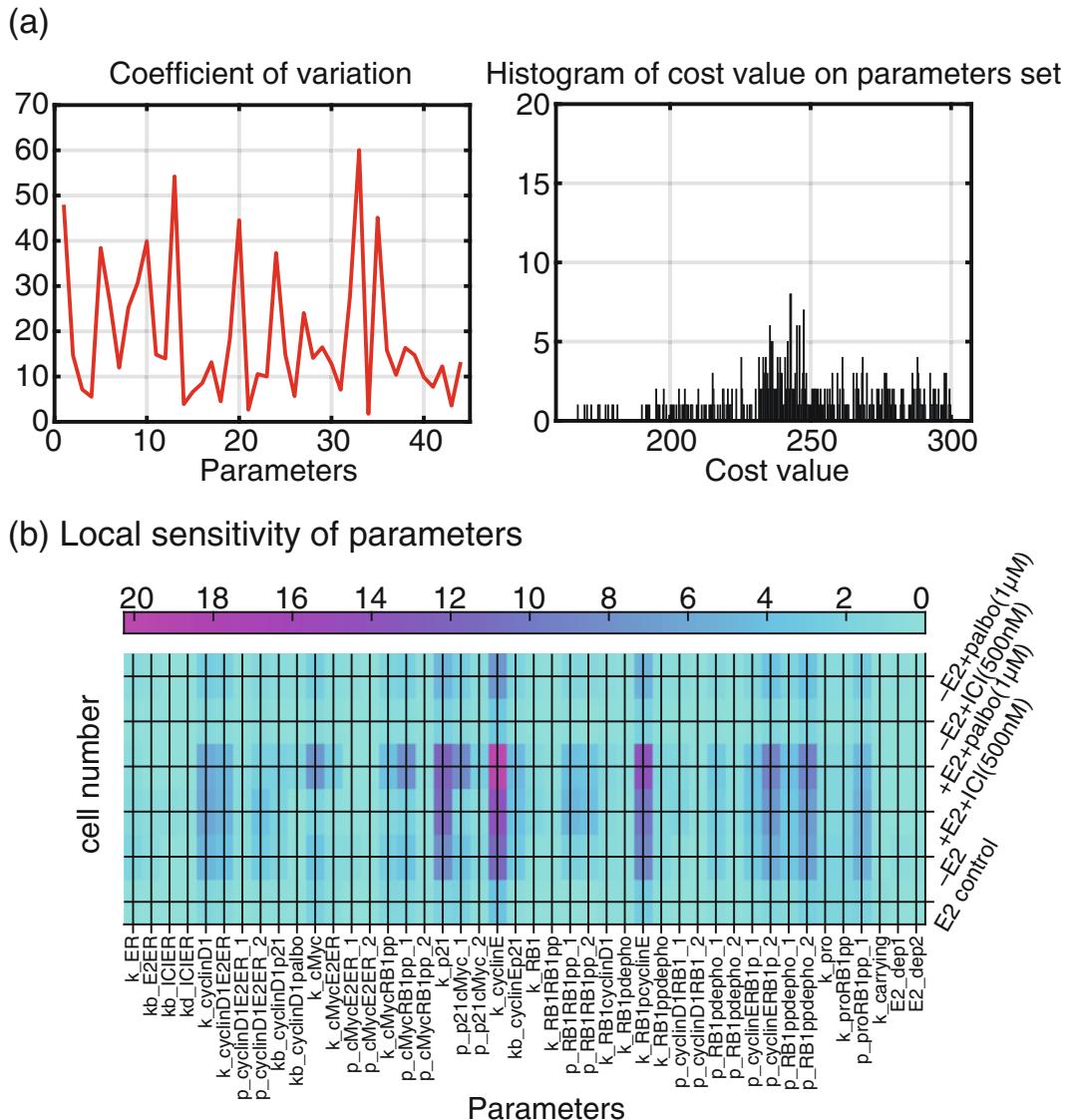


Fig. 6 (a) Coefficient of variation and histogram of cost values for parameter sets in the simulation cohort. (b) Local sensitivity of the effect on proliferation level of each model parameter. Each parameter is changed $\pm 5\%$, and the sensitivity at day 7 for each treatment is plotted. (Reproduced from ref. 9 with permission from the Royal Society)

check if the output of the model is very sensitive to the parameters. We expect that the response of a population of at least tens of thousands of asynchronous cells should not be highly sensitive to a specific signaling interaction, so the sensitivity value of a specific parameter should not be large. The plot of Fig. 6b is consistent with this idea, and the overall sensitivity values of these parameters are not high.

In summary, the aim of this chapter is to show the procedure for building a mechanistic ODE model to capture the responses of breast cancer cells to clinically used treatments for ER+ breast cancer, which is the most common type of breast cancer and accounts for about 70% of breast cancer tumors. As a mechanistic model, it can be extended in a straightforward manner to test the effect of any drug with known molecular targets in the signaling pathways being modeled. For example, we were able to add the Cdk4/6 inhibitor palbociclib into the model merely by adjusting a few new parameters related to the new interactions. This was possible because the relevant targets were already included when initially creating the model. Our strategy was to model the kinetics of signaling networks targeted by endocrine therapies and Cdk4/6 inhibitors. The calibrated mathematical model can be used to predict protein and proliferation changes in response to novel therapeutic protocols, such as combination therapies or sequential therapies. Consequently, a mathematical model based on the well-established first-line treatments for ER+ breast cancer provides the possibility of suggesting new protocols that may benefit the clinical treatment of this most common subtype of breast cancer.

Acknowledgments

This work was partly supported by Public Health Service grant R01-CA201092 to W.T.B and A.N.S.-H. Technical services were provided by shared resources at Georgetown University Medical Center, including the Tissue Culture Core Shared Resource, that were funded through Public Health Service award 1P30-CA-51008 (Lombardi Comprehensive Cancer Center Support Grant).

References

1. Citron ML, Berry DA, Cirrincione C, Hudis C, Winer EP, Gralishar WJ et al (2003) Randomized trial of dose-dense versus conventionally scheduled and sequential versus concurrent combination chemotherapy as postoperative adjuvant treatment of node-positive primary breast cancer: first report of Intergroup Trial C9741/Cancer and Leukemia Group B Trial 9741. *J Clin Oncol Off J Am Soc Clin Oncol* 21(8):1431–1439
2. Norton L, Simon R (1977) Tumor size, sensitivity to therapy, and design of treatment schedules. *Cancer Treat Rep* 61(7):1307–1317
3. Michor F, Beal K (2015) Improving cancer treatment via mathematical modeling: surmounting the challenges is worth the effort. *Cell* 163(5):1059–1063
4. Zhang J, Cunningham JJ, Brown JS, Gatenby RA (2017) Integrating evolutionary dynamics into treatment of metastatic castrate-resistant prostate cancer. *Nat Commun* 8(1):1816
5. Lalonde RL, Kowalski KG, Hutmacher MM, Ewy W, Nichols DJ, Milligan PA et al (2007) Model-based drug development. *Clin Pharmacol Ther* 82(1):21–32
6. Visser SAG, De Alwis DP, Kerbusch T, Stone JA, Allerheiligen SRB (2014) Implementation of quantitative and systems pharmacology in large pharma. *CPT Pharmacometrics Syst Pharmacol* 3(10):1–10
7. Darwich AS, Ogungbenro K, Vinks AA, Powell JR, Reny JL, Marsousi N et al (2017) Why has model-informed precision dosing not yet become common clinical reality? lessons from

- the past and a roadmap for the future. *Clin Pharmacol Ther* 101(5):646–656
8. Kirouac DC (2016) Using systems pharmacology to advance oncology drug development. In: Mager D, Kimko H (eds) Systems pharmacology and pharmacodynamics, AAPS advances in the pharmaceutical sciences series, vol 23. Springer, Cham
 9. Hryniuk W (2001) Dosage parameters in chemotherapy of breast cancer. *Breast Dis* 14:21–30
 10. Lake DE, Hudis CA (2004) High-dose chemotherapy in breast cancer. *Drugs* 64(17): 1851–1860
 11. He W, Demas DM, Conde IP, Shahjahan-Haq AN, Baumann WT (2020) Mathematical modelling of breast cancer cells in response to endocrine therapy and Cdk4/6 inhibition. *J R Soc Interface* 17(169):20200339
 12. Shin SY, Müller AK, Verma N, Lev S, Nguyen LK (2018) Systems modelling of the EGFR-PYK2-c-Met interaction network predicts and prioritizes synergistic drug combinations for triple-negative breast cancer. *PLoS Comput Biol* 14(6):1–30
 13. Kirouac DC, Du JY, Lahdenranta J, Overland R, Yarar D, Paragas V et al (2013) Computational modeling of ERBB2-amplified breast cancer identifies combined ERBB2/3 blockade as superior to the combination of MEK and AKT Inhibitors. *Sci Signal* 6(288): ra68–ra68
 14. Kirouac DC, Du J, Lahdenranta J, Onsum MD, Nielsen UB, Schoeberl B et al (2016) HER2+ cancer cell dependence on PI3K vs. MAPK signaling axes is determined by expression of EGFR, ERBB3 and CDKN1B. *PLoS Comput Biol* 12(4): e1004827
 15. Kirouac DC, Schaefer G, Chan J, Merchant M, Orr C, Huang S-MA et al (2017) Clinical responses to ERK inhibition in BRAF (V600E)-mutant colorectal cancer predicted using a computational model. *NPJ Syst Biol Appl* 3:14
 16. Spitali A, Mazzola P, Soldini D, Mazzucchelli L, Bordoni A (2009) Breast cancer classification according to immunohistochemical markers: Clinicopathologic features and short-term survival analysis in a population-based study from the South of Switzerland. *Ann Oncol* 20(4):628–635
 17. Björnström L, Sjöberg M (2005) Mechanisms of estrogen receptor signaling: convergence of genomic and nongenomic actions on target genes. *Mol Endocrinol* 19(4):833–842
 18. Farzaneh S, Zarghi A (2016) Estrogen receptor ligands: a review (2013–2015). *Sci Pharm* 84(3):409–427
 19. McDonnell DP, Norris JD (2002) Connection and regulation of the human estrogen receptor. *Science* 296(5573):1642–1644
 20. Vrtačník P, Ostanek B, Mencej-Bedrač S, Marc J (2014) The many faces of estrogen signaling. *Biochem Med* 24(3):329–242
 21. Musgrove EA, Sutherland RL (2009) Biological determinants of endocrine resistance in breast cancer. *Nat Rev Cancer* 9(9):631–643
 22. Chia YH, Ellis MJ, Ma CX (2010) Neoadjuvant endocrine therapy in primary breast cancer: indications and use as a research tool. *Br J Cancer* 103(6):759–764
 23. Tremont A, Lu J, Cole JT (2017) Endocrine therapy for early breast cancer: updated review. *Ochsner J* 17:405–411
 24. Ma CX, Reinert T, Chmielewska I, Ellis MJ (2015) Mechanisms of aromatase inhibitor resistance. *Nat Rev Cancer* 15(5):261–275
 25. Xi J, Ma CX (2020) Sequencing endocrine therapy for metastatic breast cancer: what do we do after disease progression on a CDK4/6 inhibitor? *Curr Oncol Rep* 22(6):57
 26. Thürlimann B, Keshaviah A, Coates AS, Mouridsen H, Mauriac L, Forbes JF et al (2005) A comparison of letrozole and tamoxifen in postmenopausal women with early breast cancer. *N Engl J Med* 353(26):2747–2757
 27. Prall OWJ, Rogan EM, Musgrove EA, Watts CKW, Sutherland RL (1998) c-Myc or cyclin D1 mimics estrogen effects on cyclin E-Cdk2 activation and cell cycle reentry. *Mol Cell Biol* 18(8):4499–4508
 28. Bretones G, Delgado MD, León J (2015) Myc and cell cycle control. *Biochim Biophys Acta – Gene Regul Mech* 1849(5):506–516
 29. Musgrove EA, Caldon CE, Barraclough J, Stone A, Sutherland RL (2011) Cyclin D as a therapeutic target in cancer. *Nat Rev Cancer* 11(8):558–572
 30. Sherr CJ (1995) D-type cyclins. *Trends Biochem Sci* 20(5):187–190
 31. Sherr CJ, Roberts JM (1995) Inhibitors of mammalian G1 cyclin-dependent kinases. *Genes Dev* 9(10):1149–1163
 32. Álvaro-Blanco J, Martínez-Gac L, Calonge E, Rodríguez-Martínez M, Molina-Privado I, Redondo JM et al (2009) A novel factor distinct from E2F mediates C-MYC promoter activation through its E2F element during exit from quiescence. *Carcinogenesis* 30(3): 440–448

33. Yao G, Tan C, West M, Nevins JR, You L (2011) Origin of bistability underlying mammalian cell cycle entry. *Mol Syst Biol* 7(485): 1–10
34. Leng X, Noble M, Adams PD, Qin J, Harper JW (2002) Reversal of growth suppression by p107 via direct phosphorylation by cyclinD1/cyclin-dependent kinase 4. *Mol Cell Biol* 22(7):2242–2254
35. Tedesco D, Lukas J, Reed SI (2002) The pRb-related protein p130 is regulated by phosphorylation-dependent proteolysis via the protein-ubiquitin ligase SCF(Skp2). *Genes Dev* 16(22):2946–2957
36. Morris L, Allen KE, La Thangue NB (2002) Regulation of E2F transcription by cyclinE-Cdk2 kinase mediated through p300/CBP co-activators. *Nat Cell Biol* 2(4):232–239
37. Stevens C, La Thangue NB (2003) E2F and cell cycle control: a double-edged sword. *Arch Biochem Biophys* 412(2):157–169
38. Zi Z (2011) Sensitivity analysis approaches applied to systems biology models. *IET Syst Biol* 5(6):336–346
39. Nagaraja S, Wallqvist A, Reifman J, Mitrophanov AY (2014) Computational approach to characterize causative factors and molecular indicators of chronic wound inflammation. *J Immunol* 192(4):1824–1834
40. Wijayaratne AL, McDonnell DP (2001) The human estrogen receptor- α is a ubiquitinated protein whose stability is affected differentially by agonists, antagonists, and selective estrogen receptor modulators. *J Biol Chem* 276(38): 35684–35692
41. Riggins RB, Bouton AH, Liu MC, Clarke R (2005) Antiestrogens, aromatase inhibitors, and apoptosis in breast cancer. *Vitam Horm* 71:201–237
42. Doisneau-Sixou SF, Sergio CM, Carroll JS, Hui R, Musgrove EA, Sutherland RL (2003) Estrogen and antiestrogen regulation of cell cycle progression in breast cancer cells. *Endocr Relat Cancer* 10(2):179–186
43. Cam H, Dynlach BD (2003) Emerging roles for E2F: beyond the G1/S transition and DNA replication. *Cancer Cell* 3(4):311–316
44. MacDonald JI, Dick FA (2013) Posttranslational modifications of the retinoblastoma tumor suppressor protein as determinants of function. *Genes Cancer* 3(11–12):619–633
45. Lents NH, Gorges LL, Baldassare JJ (2006) Reverse mutational analysis reveals threonine-373 as a potentially sufficient phosphorylation site for inactivation of the retinoblastoma tumor suppressor protein (pRB). *Cell Cycle* 5(15):1699–1707
46. Chung M, Liu C, Yang HW, Köberlin MS, Cappell SD, Meyer T (2019) Transient hysteresis in CDK4/6 activity underlies passage of the restriction point in G1. *Mol Cell* 76(4): 562–573.e4
47. Dakss JI, Lu RY, Facchini LM, Marhin WW, Penn LJ (1994) Myc induces cyclin D1 expression in the absence of de novo protein synthesis and links mitogen-stimulated signal transduction to the cell cycle. *Oncogene* 9(12): 3635–3645
48. Marra A, Curigliano G (2019) Are all cyclin-dependent kinases 4/6 inhibitors created equal? *npj Breast Cancer* 5(1):27
49. Hafner M, Mills CE, Subramanian K, Chen C, Chung M, Boswell SA et al (2018) Multiomics profiling establishes the polypharmacology of FDA-approved CDK4/6 inhibitors and the potential for differential clinical activity. *Cell Chemical Biology* 26(8):1067–1080.e8
50. Carroll JS, Prall OWJ, Musgrove EA, Sutherland RL (2000) A pure estrogen antagonist inhibits cyclin E-cdk2 activity in MCF-7 breast cancer cells and induces accumulation of p130-E2F4 complexes characteristic of quiescence. *J Biol Chem* 275(49):38221–38229
51. Lewis JS, Osipo C, Meeke K, Jordan VC (2005) Estrogen-induced apoptosis in a breast cancer model resistant to long-term estrogen withdrawal. *J Steroid Biochem Mol Biol* 94(1–3):131–141



Chapter 17

SynDISCO: A Mechanistic Modeling-Based Framework for Predictive Prioritization of Synergistic Drug Combinations Targeting Cell Signalling Networks

Sung-Young Shin and Lan K. Nguyen

Abstract

The widespread development of resistance to cancer monotherapies has prompted the need to identify combinatorial treatment approaches that circumvent drug resistance and achieve more durable clinical benefit. However, given the vast space of possible combinations of existing drugs, the inaccessibility of drug screens to candidate targets with no available drugs, and the significant heterogeneity of cancers, exhaustive experimental testing of combination treatments remains highly impractical. There is thus an urgent need to develop computational approaches that complement experimental efforts and aid the identification and prioritization of effective drug combinations. Here, we provide a practical guide to SynDISCO, a computational framework that leverages mechanistic ODE modeling to predict and prioritize synergistic combination treatments directed at signaling networks. We demonstrate the key steps of SynDISCO and its application to the EGFR-MET signaling network in triple negative breast cancer as an illustrative example. SynDISCO is, however, a network- and cancer-independent framework, and given a suitable ODE model of the network of interest, it could be leveraged to discover cancer-specific combination treatments.

Key words SynDISCO, Drug combination, Drug synergy, ODE model, Mechanistic modeling, Computational simulation, Signaling network

1 Introduction

The inevitable development of resistance to anticancer monotherapies has highlighted the need to identify combinations of drug agents as a more effective approach to cancer treatment [1, 2]. Indeed, combination therapies are being actively explored across all cancer indications [3]. Yet, given the vast space of possible target combinations, we are faced with daunting challenges of how to predict and prioritize the best possible combinations of targets (and drugs targeting them) for clinical validation. Although experimental screenings of combinations of existing drug agents are becoming increasingly feasible [3], the large number of available drugs and the marked heterogeneity of cancers render exhaustive

experimental screens costly and impractical. Moreover, experimental screens can only incorporate molecular targets that already have existing drug(s) but cannot examine candidate targets for which no drugs are available. Given the still rather sizable number of the latter, this presents a significant missed opportunity. There is therefore an urgent need to develop new computational approaches capable of interrogating a large number of drug combinations efficiently in an unbiased manner, from which their antitumor effects can be systematically predicted and prioritized.

Signaling pathways play a critical role in the governing of cancer-relevant cellular processes, including cell growth, death, migration, and differentiation [4, 5]. Reflecting this, signaling protein kinases represent a major class of cancer therapeutic targets. Following the FDA approval of the first kinase inhibitor imatinib in 2001, 20 years later in 2021, 62 therapeutic agents targeting about two dozen different protein kinases have been approved by the FDA, most of which are prescribed for the treatment of neoplasms [6]. However, the number of kinases having approved drugs is dwarfed compared to the total number of kinases encoded in the human proteome (518 [6]), many of which have been implicated in tumorigenesis, leaving significant room for further therapeutic discovery.

However, the translation of kinases from promising candidates to clinically actionable targets has been slower than expected. Part of the reason is due to the enormous complexity of cellular signaling networks. These networks usually contain intricate post-translational modifications and complex intertwined feedback loops and elaborate pathway crosstalk, which together render it highly challenging to analyze and predict cellular response to drug treatment [7, 8]. These complexities also complicate the search for potential synergistic therapeutic targets within signaling networks. To address these issues, mechanistic and dynamic mathematical models, such as Boolean network models [9, 10], dynamic Bayesian networks [11], fuzzy-logic models [12], and ordinary differential equation (ODE) network models [5, 13–16] have been developed for a variety of signaling networks. In addition to translating complex biochemical relations into precise mathematical terms, these models also provide powerful quantitative platforms for analyzing and predicting drug response, as well as inferring causal mechanisms that underscore nontrivial behaviors, all of which are critical in facilitating the identification of novel and effective drug combinations.

In this chapter, we present a practical guide to SynDISCO, a computational framework that leverages mechanistic ODE modeling to predict and prioritize synergistic combination treatments directed at signaling networks. We first demonstrate the key modules and steps of SynDISCO and then illustrate its application through a case study involving the EGFR-MET crosstalk network

in triple negative breast cancer. SynDISCO is, however, independent of the signaling network or cancer type, and as long as an appropriate ODE model of the network is available, it could be integrated on top of the model to discover cancer-specific combination treatments.

2 Description of SynDISCO

The entire SynDISCO pipeline consists of three modules, depicted in Fig. 1. The first module involves construction of a predictive mathematical model in ODE format that quantitatively describes a signaling network/pathway of interest. This module, however, can be skipped if such a suitable model already exists. Building upon the ODE model from module 1, the second module involves computational simulation and evaluation of a specified number of pairwise drug combinations directed at the network nodes (model species), both in terms of inhibitory effect and drug synergy. The drug combinations are then ranked and prioritized according to the levels of computed synergistic effect. Once the combinations are ranked, the top candidates will be tested experimentally, and the results will be cross-validated with model predictions in the last module, module 3. While it is highly desirable for module 3 to take place, in practice this is optional depending on the experimental capabilities of the investigator team. In such a case, the end result of SynDISCO would be a set of predicted promising drug combinations that can be followed up experimentally by other researchers. Below, we describe the steps within each module in more details, with an emphasis on module 2 due to its central importance within the whole pipeline.

2.1 Module 1: ODE Model Construction

Once a signaling network of interest is settled, construction of an ODE network model can begin. This typically involves stepwise establishment of a network wiring diagram, followed by a reaction schematic diagram, then formulation of differential equations from the model reactions, and finally by model calibration against known experimental data [17, 18].

Building a biologically accurate and evidence-based wiring diagram that specifies the network components and how they interact is the first and critical step in kinetic model development. A good starting point for this would be to consult signaling pathway databases [19], for example, Reactomes [20], KEGG [21], Signor [22], or WikiPathways [23]. However, since the wiring of the same signaling network may differ subtly between different tumor types (or even cell lines of the same tumor) while these databases typically do not contain context-specific interactions, one often has to refine the diagram using knowledge of interactions specific to the biological context of interest. These could come from in-house

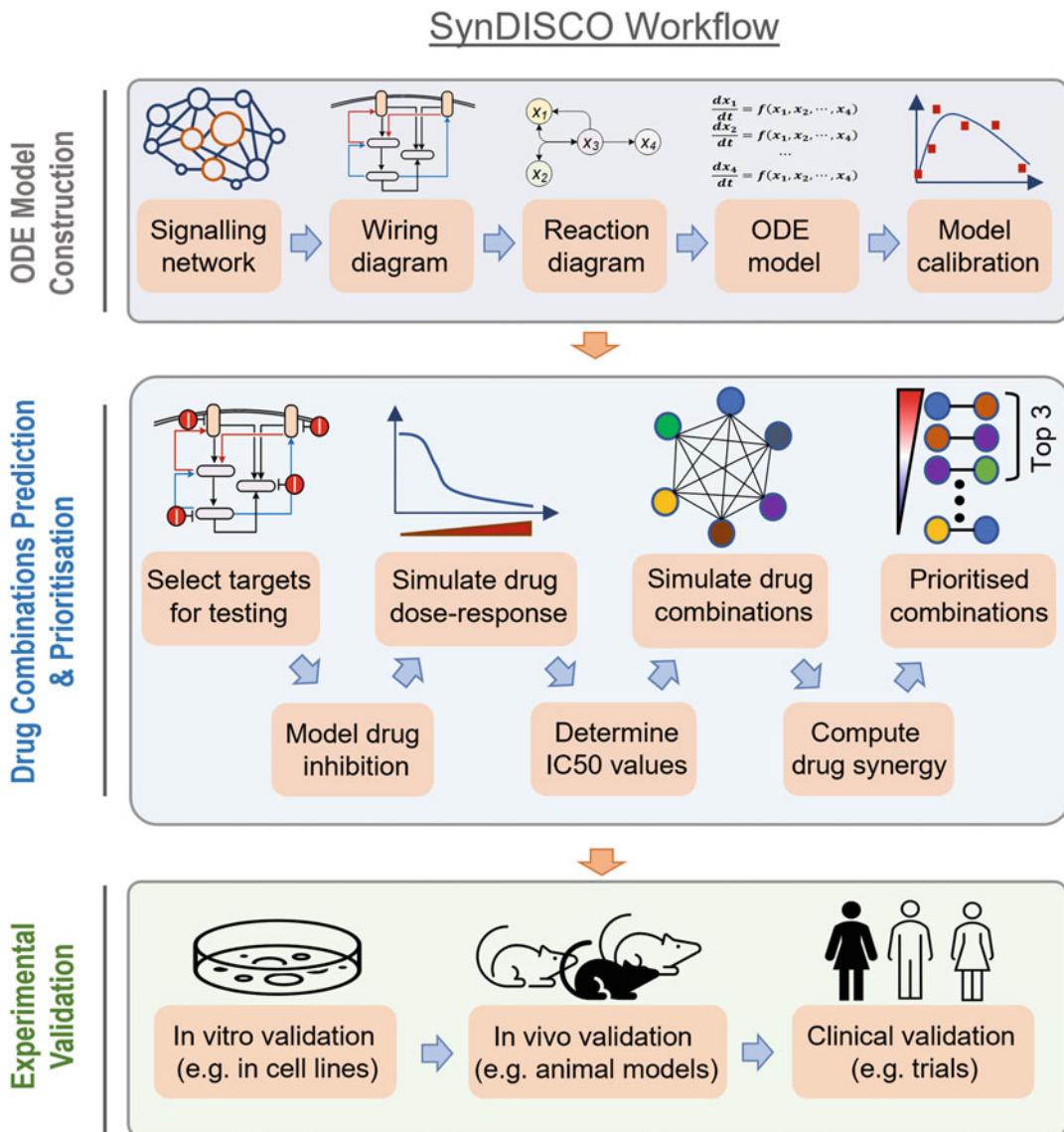


Fig. 1 A workflow of SynDISCO. The pipeline contains three modules: ODE model construction (module 1), drug combinations prediction and prioritization (module 2) and experimental validation (module 3)

data, up-to-date reviews, or previous models developed for the same network (e.g., from Biomodels [24]). More often than not, conflicting information about specific links may arise, at which point one has to try to clarify and resolve, for instance, through obtaining additional experimental data. In the case this is not possible and assumptions have to be made, they should be clearly stated.

Upon completion of the wiring diagram, it should be made further precise by converting it into a reaction schematic diagram

[5, 13]. Here, the interactions between network nodes are described by biochemical reactions with clear specification of the substrates, products, and catalyzing enzyme species. As an example, a positive regulatory link between a kinase (K) and its substrate (S) in the wiring diagram would be represented in the reaction diagram by a reaction: $S \rightarrow$ phosphorylated S (pS), catalyzed by K. The advantage of this step is that once it has been done, the ODEs can be straightforwardly translated from the reaction diagram without any ambiguity. In other words, the reaction diagram provides a one-to-one visual mapping to the ODEs that is convenient for model checking, modification, and troubleshooting. Next, the rate equations for each reaction are constructed based on kinetic laws, for example, mass-action kinetics for association/dissociation of molecules, Michaelis-Menten kinetics for (de)-phosphorylation and (de)ubiquitination, and Hill kinetics for transcriptional regulation [17]. A set of ODEs can then be built based on the rate equations and by considering appropriate production and elimination terms for each node.

The final step in module 1 is to calibrate the ODE model against known experimental data, which is critical in conferring the model with identity and predictive power. Building a model without this step is a job half done. Model calibration involves computational estimation of unknown model parameters using optimization algorithms with the goal to minimize the discrepancies between model simulation and experimental data. A variety of techniques are available for model parameter estimation, and we refer readers to an excellent recent review by Villaverde et al. [25] for detailed information regarding what best to and not to do. The ideal end product here is a well-calibrated ODE model of the signaling network under study.

2.2 Predictive Prioritization of Drug Combinations

Using the predictive ODE model from module 1 as input, module 2 aims to predict and prioritize various pair-wise drug combinations targeting the network nodes based on the levels of synergistic effect displayed by them. This module involves multiple steps described below.

Selection of targets for virtual screening Prior to simulation, a set of drug targets need to be specified. This involves specification of the pairs of targets, that is, suitable nodes within the modeled network that will be co-inhibited. It is important to note that the choice of targets for computational evaluation does not necessarily require them to have actual existing drugs, meaning suitable targets without any targeting drugs can be considered (*see Note 1*).

Modeling drug inhibition of target nodes Following the selection of target nodes, we assume that each of them in theory can be

inhibited by a corresponding small-molecule inhibitor. Hence, hereafter we will refer to target combinations and drug combinations interchangeably. The action of the inhibitor on its target needs to be explicitly incorporated into the existing ODE equations by modeling the target-inhibitor interactions.

In this regard, two common modes of target inhibition can be considered: competitive and allosteric inhibition. Consider a simple catalytic reaction where a kinase binds the substrate (S) to form a complex (ES) before producing the product (P) (Fig. 2). In competitive inhibition, an inhibitor (I) binds to the kinase and form a competitive, inhibitory complex (EI) (Fig. 2a). This type of inhibition can be completely overcome by very high concentration of the substrate S . Thus, the rate of the kinase-catalyzed reaction becomes saturated at a certain point (same V_m), whereas the Michaelis constant (K_m) is increased by a factor of $(1 + [I]/K_i)$, where $[I]$ is the drug concentration and K_i is the dissociation constant (Fig. 2c).

On the other hand, in allosteric inhibition, the kinase has an additional binding site other than the site for the substrate [26], where an inhibitor binds, forming both a binary (ES) and ternary complex (ESI) (Fig. 2b). Unlike competitive inhibition, in allosteric inhibition V_m changes by a factor of $(1 + [I]/K_i)$ while K_m is not affected (Fig. 2b, d).

Simulating drug dose-response curves Having incorporated the actions of the inhibitors, one can simulate the effect of drug treatment under various conditions. It is important to make sure the model is under the steady state before application of the drugs (Fig. 3a), in order to better mimic experimental scenarios (where cells are often grown in cultured medium before drug treatment). However, in general specific model simulations should be guided by the exact experimental setup (see Note 2). Next, to determine the dynamic responses to a range of drug concentrations for subsequent analyses, dose-response simulation should be performed (Fig. 3b). The most obvious response readout would be the activity of the corresponding drug target but can also be a (theoretical) function representing cell growth or viability (see Note 3). Dose-response curves are established by gradually increasing the concentration of the drugs and simulating the levels of the response readouts, often at a time point where steady state is reached after drug treatment.

Determination of drugs' IC₅₀ values The half-maximal inhibitory concentration (IC₅₀) is a commonly used value for quantifying the potency of a drug and is often employed in drug combination studies. The IC₅₀ is also used as a reference value of a drug concentration because drugs have different sensitivity for

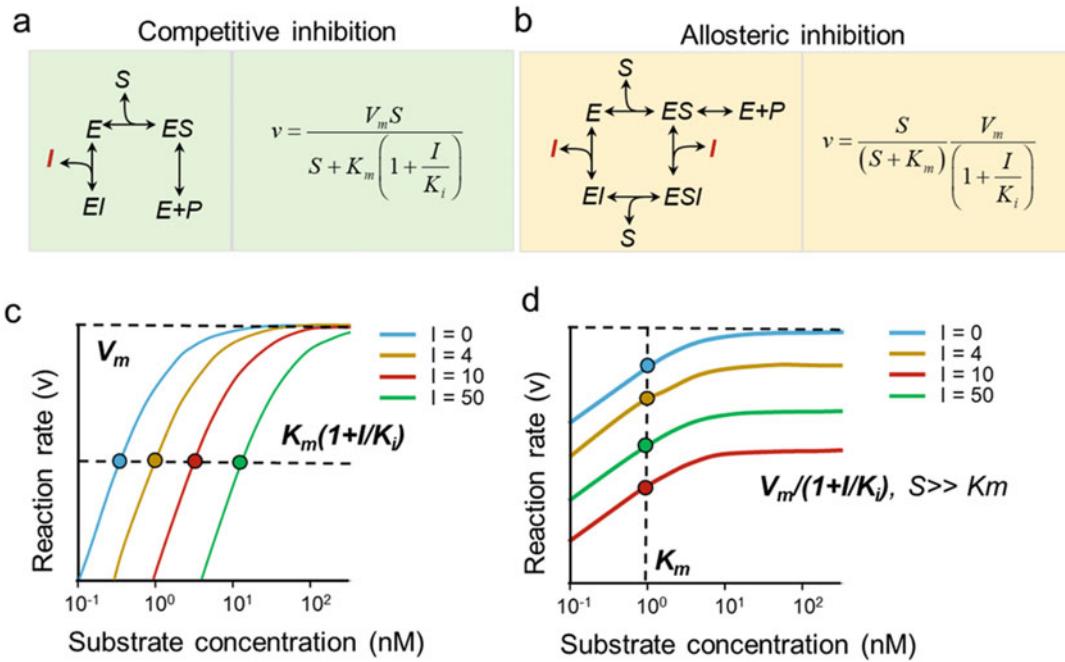


Fig. 2 Mathematical formulation of different modes of drug-target interaction. **(a)** Competitive inhibition. **(b)** Allosteric inhibition. E, enzyme; S, substrate; I, inhibitor; P, product. **(c, d)** Substrate–rate curve for competitive inhibition **(c)** and allosteric inhibition **(d)**

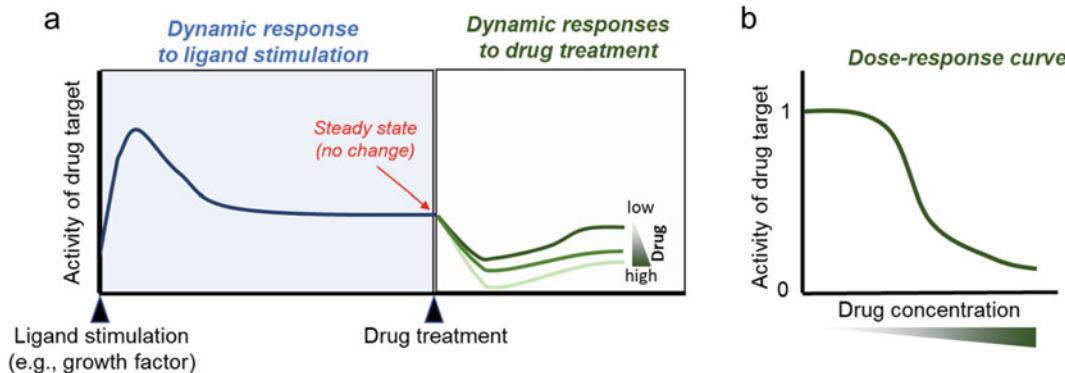


Fig. 3 An illustration of a time-dependent drug response simulation. It consists of two-step simulations: ligand stimulation and drug treatment **(a)**. Dose-response curve measured at a specific time point (e.g., 24 h after a drug treatment) **(b)**

different cells. Thus, in an experimental design, one uses a fold of IC₅₀ concentration (e.g., IC₂₅ or IC₅₀) to treat cells rather than using absolute drug concentration. Here, the theoretical IC₅₀ value for each drug inhibitor will be computed based on the dose-response curves from the previous step by fitting these curves to the four-parameter dose-response function [27] depicted in Fig. 4.

Four-parameter dose-response function

$$\text{Response} = \text{Bottom} + \frac{(\text{Top} - \text{Bottom})}{\left(1 + \left(\frac{\text{IC50}}{X}\right)^{\text{HillSlope}}\right)}$$

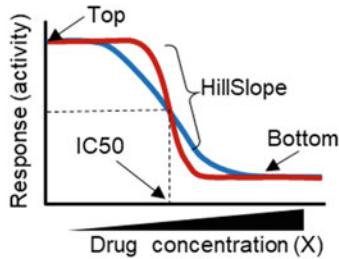


Fig. 4 The four-parameter dose-response model for estimation of IC50. Top is the maximal value of response curve, and bottom is the minimal value (i.e., maximally inhibited response). Top and bottom are plateaus of the curve in the units of the Y-axis. HillSlope denotes the steepness of the sigmoidal curve. When HillSlope = 1, the dose response is “standard”; when HillSlope <1, the curve become “shallower”; when HillSlope >1, the curve become “steeper”

Simulating the effect of drug pairs Having determined the suitable dose for each drug (e.g., IC50), here the effect of the pair-wise drug combinations can be systematically simulated and compared. Because in most scenarios we want to inhibit the growth/viability of the cancer cells, thus the effect of the drug combinations on a molecular readout that best reflect cell growth/viability can be considered. Alternatively, as discussed above and in Note 3, one can use a heuristic function representing cell viability as the readout to judge the effect of drug treatment. In any cases, for each drug combination consisting of drug A and B, the effects of vehicle treatment (no drug), drug A alone, drug B alone, and their combination are simulated.

A key way in which SynDISCO assesses drug combinations is to quantify possible synergy (or lack thereof) between the individual drugs. To this end, different approaches can be used to compute drug synergism, depending on the specific context and preference. These include the Coefficient of Drug Interaction (CDI) model [28, 29], the Bliss Independence (BI) model [30, 31], the Chou-Talalay model [32], highest single agent (HSA) model [33], and zero interaction potency (ZIP) model [34, 35]. More details about these techniques are given in Notes 4 and 5. Regardless of the chosen approach, a synergy score is derived for each drug combination. For example, using BI, based on the single-drug and combined drug effects previously simulated, a synergy score for each drug pair is computed where the synergy score < 1, = 1 or >1 indicates that the drugs are synergistic, additive, or antagonistic, respectively. Importantly, a smaller BI score indicates stronger synergism, while a larger BI score indicates stronger antagonism. Taking advantage of this quantitative property, SynDISCO then ranks all the tested combinations in order from most synergistic to most antagonistic, enabling prioritization of the combinations (Fig. 5).

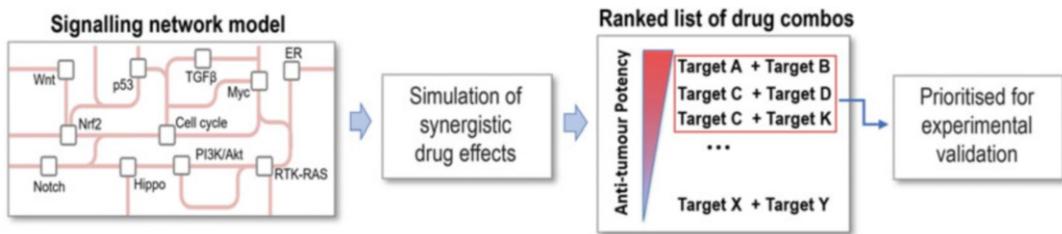


Fig. 5 A workflow of identification and prioritization of combinatorial drugs

2.3 Experimental Validation of Top Candidates

By this stage, one should have obtained a set of highly synergistic drug combinations predicted by modeling that are ready for experimental validation. As mentioned earlier, although this module is optional depending on whether the investigators have the capacity to perform experiments, it is highly recommended as it is the only way to find out if the model predictions are correct or not. More importantly, any discrepancies between the experimental data and prediction will help pinpoint gaps in the model, allowing it to be further refined and improved.

One of the most common ways to test candidate cancer drug combinations is to perform experiments in cancer cell lines cultured in the laboratory. For example, the effect of single-drug and combination drug treatments, as well as vehicle control (e.g., DMSO) on cell viability or colony formation ability can be done using MTS/MTT assays and/or colony formation assay, respectively. The MTS assay is normally used to assess cell proliferation, cell viability, and cytotoxicity based on the reduction of the MTS (3-(4,5-dimethylthiazol-2-yl)-5-(3-carboxymethoxyphenyl)-2-(4-sulfophenyl)-2H-tetrazolium) compound by viable mammalian cells [36]. The MTT (3-[4,5-dimethylthiazol-2-yl]-2,5 diphenyl tetrazolium bromide) assay is a colorimetric assay for assessing cell metabolic activity by measuring NAD(P)H-dependent cellular oxidoreductase, which reflects the number of viable cells present [37]. On the other hands, colony formation assay (or clonogenic assay) is an in vitro cell survival assay based on the ability of a single cell to grow into a colony [38]. Other methods such as the Incucyte or xCelligence systems can be also used to visualize and quantify living cell behavior over time by automatically gathering and analyzing images [39, 40]. Promising candidates validated in cell lines can then be further tested in more advanced models such as organoids or mouse xenograft cancer models.

3 Application of SynDISCO to the EGFR-PYK2 Signaling Network in TNBC

In this section, we provide a concrete application by illustrating how SynDISCO is used to identify effective drug combinations directed at the EGFR-MET signaling network in triple negative

breast cancer (TNBC), an aggressive subtype of breast cancer [13]. The epidermal growth factor receptor (EGFR) and hepatocyte growth factor receptor (HGFR or MET) are frequently over-expressed in TNBC, making them promising therapeutic targets [41–43]. However, single-drug inhibition of these kinase receptors failed to induce durable clinical benefit, primarily due to emergence of drug resistance [44, 45]. Identification of more effective combinatorial therapies that circumvent resistance is therefore of urgent importance for TNBC.

3.1 Construction of the EGFR-MET Network Model

Building reaction diagram Given our purpose here is to predict effective drug combinations targeting the EGFR-MET crosstalk signaling network in TNBC, our first step is thus to establish an ODE model of this network in the context of TNBC (Fig. 6a).

ODE model formation From the reaction diagram, a set of ODEs was formulated using a combination of kinetic laws. Specifically, we employed Michaelis-Menten (MM) kinetics for (de)-phosphorylation and (de)ubiquitination reactions, mass-action kinetics for association/dissociation reactions, and Hill kinetics for transcription reactions. In the end, the model consists of 13 ODEs and 25 reactions, which are given in Appendices A and B.

Model calibration As the last step of module 1, we calibrated the model using a dataset of kinetic, time-resolved data under different perturbation conditions, obtained from the TNBC cell line MDA-MB-468. This includes the time-dependent changes in the phosphorylated and total levels of key network components (EGFR, PYK2, MET, and STAT3) in response to EGF stimulation. A genetic algorithm-based global optimization procedure implemented in Matlab was used for model calibration and estimation of unknown model parameters [46, 47]. As result of this process, a set of best-fitted parameter values was obtained (Appendix C). Model simulations using the best-fitted parameter set recapitulate the kinetic profiles of the observed data, as shown in Fig. 6b. The calibrated model now can be used for the next steps in module 2.

3.2 Drug Response Simulation of the EGFR-MET Model

Selecting Targets

We select EGFR, PYK2, STAT3, and MET as possible targets for exploration because they are known pro-growth kinases and are frequently altered in TNBC [13]. About half of TNBC overexpress EGFR [48]. Proteomics analysis revealed that PYK2 is active mainly in basal-like breast carcinomas among breast cancer subtypes, and activated PYK2 correlates with high levels of EGFR [42]. Moreover, STAT3 frequently is activated in TNBC cells [49], while MET is highly expressed in ~52% of TNBC [50].

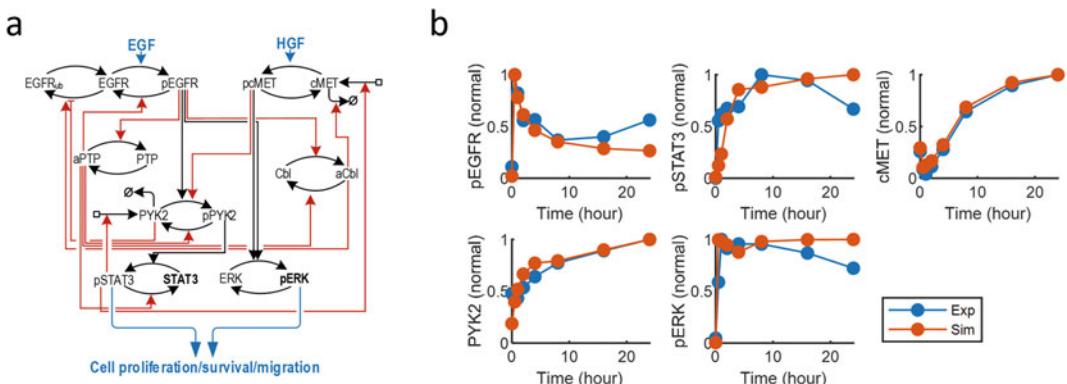


Fig. 6 An EGFR-MET network model and model calibration [13]. (a) A schematic diagram of EGFR-MET network model. (b) The model calibration and fitting to experimental data. Red line: simulated data of the best-fitted model. Blue line: experimentally observed time-course data

Modeling Drug Inhibition

We assumed in the model that the target EGFR, PYK2, MET, and STAT3 can be inhibited by their respective small-molecular inhibitors: Gefitinib, PF396, EMD-1214063 (EMD), and Stattic, respectively. We assumed Gefitinib effectively blocks the phosphorylation of major EGFR tyrosine sites [51]. PF396 inhibits the kinase activity of PYK2 by preventing the transfer of a phosphate group to a target protein from ATP [52]. EMD is a small molecule that inhibits c-Met phosphorylation of target substrates [53, 54]. In particular, we assumed that Gefitinib, PF396, and EMD allosterically inhibit their targets since there is no evidence that they compete for the binding sites with the activating kinases [13]. On the other hand, Stattic was modeled as a mass action interaction with a target molecule due to its a reversible drug-target binding reaction [55].

Simulating Drug Dose-Response and Determining IC50 Values

Here, we assumed the phosphorylated levels of ERK and STAT3 (pERK and pSTAT3), the two major oncogenic signaling effectors downstream of EGFR and MET as the models' co-outputs that are used as surrogate markers of cell growth in order to assess the effect of drug treatments. Drug potency is therefore determined by the degree to which the drug inhibits these readouts.

As an example, Fig. 7a displays the simulated dose-response curves for each drug, using pSTAT3 as a readout. These curves were then fitted to the Hill-type function discussed in Sect. 2. The estimated values of IC50 along with other function parameters for each drug are displayed in Fig. 7b. The estimated IC50 values provide the drug-specific doses as basis for simulation of drug combinations in the next section. Similar steps should also be done for pERK as the readout in order to obtain drug-specific doses specific for this readout (not shown).

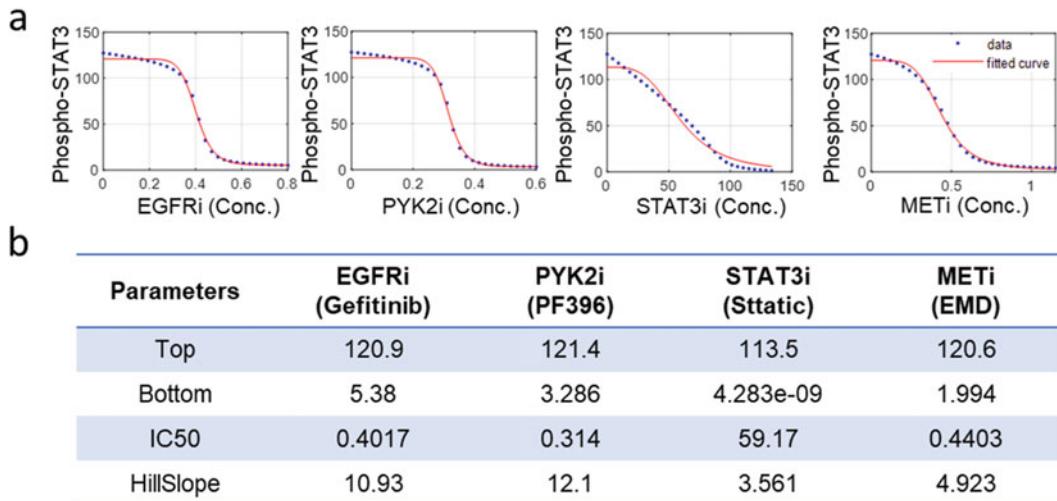


Fig. 7 Estimation of IC₅₀ using the four-parameter dose-response model. (a) Dose-response curves of pSTAT3. (b) The estimated parameter values of the dose-response function

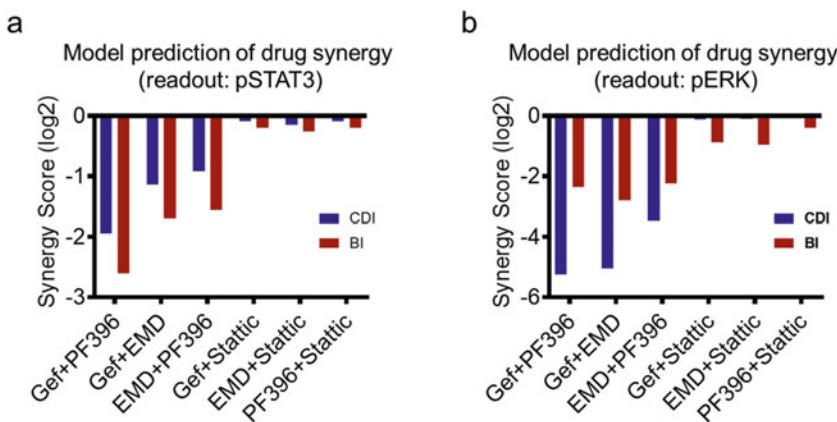


Fig. 8 Ranking synergy scores of drug combinations. pSTAT3 (a) and pERK (b) were used as a readout for calculation of the synergy score. IC₂₅ concentrations were used for each drug (note that similar results were found using IC₅₀ concentrations)

Simulating Drug Combination Effect and Drug Synergy

With four drugs, we have a total of six different pair-wise combinations for evaluation. For each drug pair, we simulated the steady-state response of the readouts, that is, pERK and pSTAT3, to the single-drug treatments, the combined treatment at the pre-determined dose (e.g., IC₂₅ or IC₅₀) of each drug, and vehicle (no drugs) treatment. The drug synergy score for each drug pair can then be computed at this dose combination, using different synergy scoring methods such as CDI or BI (see Note 4). Figure 8a displays the computed synergy scores for the six drug combinations using pSTAT3 as readout, ordered from the most synergistic to

most antagonistic combinations based on the values of the score. This ranked list allowed us to identify the most synergistic combinations which will be prioritized for experimental validation. Specifically, the model predicted that among the six combinations, dual inhibition of EGFR-PYK2 (using Gefitinib + PF396) and EGFR-MET (using Gefitinib + EMD) represents the most synergistic combinations, a prediction consistently made by both CDI and BI (Fig. 8a). Importantly, we also obtained similar ranking of the combinations when pERK was used as the readout (Fig. 8b). This led us to hypothesize that co-targeting EGFR-PYK2 or EGFR-c-Met will yield synergistic effects in inhibiting TNBC growth.

The concordance in terms of drug combination ranking between the readouts makes it easier to prioritize the top combinations; however, it is possible that the predicted ranking may differ between different signaling readouts. In this case, one may consider using a single composite function of multiple signaling readouts as an indicator of cell viability (*see Note 3*).

Once highly synergistic drug combinations are predicted, further analyses are recommended to obtain deeper insights into the mechanistic basis of how synergy is gained. For example, one can interrogate the time-course response of the network in single vs. combined drug treatments. A common reason for synergy is that single-drug treatments tend to trigger unwanted rebound in oncogenic activity of specific network nodes, and combined drug treatments can eliminate such rebound and enable more durable blockade of oncogenic signals [2, 8].

Simulation at Varying Dose Combinations

The simulations above have primarily focused on testing the drug combinations by combining the drugs at specific doses (e.g., IC₅₀ or IC₂₅). However, a strength of our model-based approach is that it allows one to simulate drug combinations at myriad dose combinations within specified dose ranges, thus enabling a broader view of combinatorial drug effects. Using the top-predicted EGFR-PYK2 combination as an illustrative example, Fig. 9a displays the simulated effects of dual inhibition of EGFR and PYK2 on the levels of pSTAT3 at increasing combined doses of the individual drugs. Follow on from this, Fig. 9b displays the landscape of synergy score (CDI) computed at the corresponding dose combinations. These simulations revealed that the synergy score can vary significantly depending on the combined drug doses. Co-targeting EGFR-PYK2 displayed strongest synergism for the CDI score at around the IC₅₀ of the drugs and for the BI score about IC₂₅ (Fig. 9c) but exhibited antagonism at high drug concentrations. Although the synergy score landscape differs slightly between different drug synergy models because of different model assumptions, the above observation was supported by all the models. It is a good practice to perform different drug synergy models for cross-comparison (*see Note 5*).

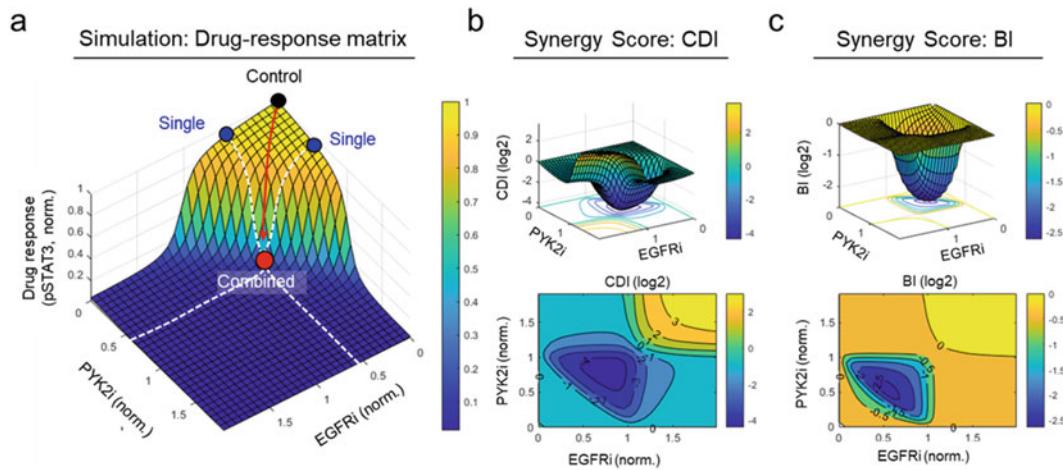


Fig. 9 Simulation of pair-wise drug combination and computing drug synergism. (a) Drug response of pSTAT3 to EGFR and PYK2 inhibitors. Activation state of STAT3 was slightly suppressed upon single-drug treatments (blue dots), while the combined treatment significantly suppressed these activation states (red dot). (b) Synergy score of CDI (b) and BI (c). Note the drug concentration was normalized to the IC₅₀ value of each drug. “1” indicates the IC₅₀ value

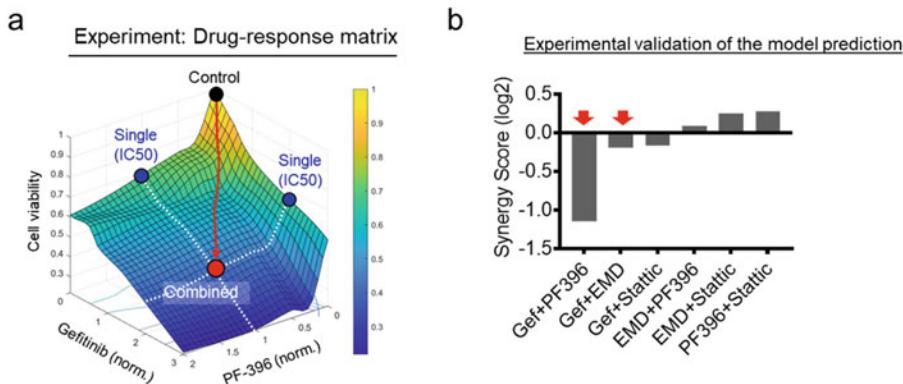


Fig. 10 Experimental validation of pair-wise drug combination. (a) Drug response of cell viability to Gefitinib and PF-396. (b) Ranking synergy score of drug combinations. IC₅₀ concentrations were used for each drug

3.3 Experimental Validation of Top-Predicted Drug Combinations

It is desirable to verify model predictions with follow-up experiments. To try experimentally to validate the predictions in module 2, TNBC cells MDA-MB-468 were subject to increasing levels of the inhibitors Gefitinib, PF396, EMD, and Stattic in six pair-wise combination schemes, and the effects of the single- and combined-drug treatments on cell viability were measured using the MTT assay [13]. As an example, Fig. 10a presents the data for

co-targeting EGFR and PYK2 using Gefitinib + PF396 (*see Note 5* for more information on experimental setup of drug treatment).

To analyze the synergism of all six drug combinations, we calculated the synergy scores using the obtained data based on the Chou-Talalay method (Fig. 10b). Consistent with model prediction, the data-based ranking of the combinations indicated that dual blockade of EGFR-PYK2 using Gefitinib + PF396, respectively, was indeed the most synergistic combination, followed by EGFR-MET co-targeting by Gefitinib + EMD. In addition, the ranked order of the six drug combinations was largely in concordance with model prediction. Importantly, co-targeting of EGFR and PYK2 was also shown to significantly reduce the growth of basal-like TNBC cells in animal models, and it was found that this combination was more potent than dual inhibition of EGFR and Met [42]. Together, these experimental studies support the model predictions.

4 Summary

Resistance to anticancer treatments, particularly single-agent therapies, remains a major clinical barrier to a cure in most cancer types. This highlights the critical need to predictively identify combination therapies of clinical potential in an unbiased, scalable, and cheap manner. These capabilities would enable the most promising candidates to be tested experimentally in a selective rather than brute-force strategy, thereby saving cost and accelerating the bench-to-bedside translation for cancer drugs. While computation modeling and simulation have been an integral part of the cancer systems biology field [5, 13, 16, 47], modeling-based approaches that predict synergistic drug combinations remain limited. In this chapter, we have provided a pedagogical guide to SynDISCO, a holistic framework developed precisely for this purpose. We have demonstrated the applicability of SynDISCO through identification of synergistic drug combinations against triple negative breast cancer. We hope it will serve as a useful tool for discovery of potentially effective drug combinations in other cancer settings.

5 Notes

1. Selection of Potential Targets

While experimental screens for drug combinations are limited by the availability of the drugs, this is not the case for SynDISCO-based screening. In principle, any network nodes can be included in the pool of potential targets for combination exploration. If it turns out that a node (with no targeted drugs) is robustly predicted to be a promising co-target in specific

combinations, this may serve as a motivation to develop new compounds against that target. This is useful because the complexity of signaling network wirings may mask the therapeutic vulnerabilities of certain network nodes, and their hidden therapeutic potential can be revealed by computational simulations. On the flip side, in SynDISCO, the exploration of drug targets is constrained by the scope and size of the input ODE models since one cannot investigate targets that are not included in the model. Thus, the plan regarding which targets to be explored should be considered early and factored into the model construction step in module 1 to determine the model scope.

While the EGFR-MET example above represents a proof-of-principle case study with a small number of drug combinations, this number can be scaled up significantly given a larger ODE model. In other studies, we have used SynDISCO to evaluate tens of targets and hundreds of possible combinations without issues [56]. Generally, there is a trade-off between the number of network nodes and the computational cost required to evaluate all possible target/drug pairs, and one needs to consider this balance, taking into account available computing capabilities.

2. Dynamic Simulations

The drug dose-response curves in module 2 can be generated by measuring the readouts (i.e., responses) at different time points post drug treatment. A good approach is to get a sense of how long it would take for the network to settle into steady state following drug perturbations by performing long-term time-course simulations. For example, if the system takes 24 h to reach steady state, this time point could be used for the dose-response simulations (Fig. 3b). Sometimes, one may be interested in interrogating the acute effect of the drugs, and much earlier time points (e.g., 1 h) could be used instead.

3. Heuristic Function Representing Cell Viability

Because the outputs of ODE models of signaling networks are signaling and not phenotypic readouts (e.g., cell viability), the effects of drug treatment on phenotypic responses cannot be directly derived. Rather, these are usually indirectly determined. One approach is to assume key oncogenic downstream model outputs (e.g., pSTAT3 and pERK, as in Sect. 3) as proxies of cell phenotypic responses (often cell viability). However, in the case where cell viability is best represented by multiple model outputs and/or when there are significant inconsistencies in the results between different outputs, then a composite function of these outputs could be formulated. A simple approach is to assume that the selected model outputs contribute linearly to the cell viability function as follows [57]:

$$\text{Cell Viability Function} = \sum_1^n \text{pro } W_i \tilde{n} \text{pro } O_i - \sum_1^m \text{anti } W_i \tilde{n} \text{anti } O_i,$$

where $\text{pro } O_i$ represents the n contributing pro-growth model outputs, while $\text{anti } O_i$ represents the m contributing anti-growth model outputs (e.g., proapoptotic nodes), and $\text{pro } W_i$ and $\text{anti } W_i$ are the corresponding weights (normalized, between 0 and 1). Determining the weights of the outputs is a nontrivial task and may require further data and/or assumptions. For example, these weights can be estimated as part of the model calibration procedure given suitable experimental data that links cell viability to signaling readouts [57]. In the absence of such data, a reasonable starting point may be to assume equivalent weights (i.e., equal 1).

4. Methods to Compute Drug Synergy

There are a number of different methods for calculating drug synergy, underscored by different mathematical theories. These include the Coefficient of Drug Interaction (CDI) model [28, 29], the Bliss Independence (BI) model [30, 31], the Chou-Talalay model [32], highest single agent (HSA) model [33], and zero interaction potency (ZIP) model [34, 35]. Here, we provide a brief description of the two common and simpler methods, CDI, BI, as a guide to the topic. However, we refer readers to the cited references for detailed technical explanation of each method.

Coefficient of Drug Interaction: CDI is a simple model to evaluate the inhibitory effect of the drug combination [28, 29]. The synergy score is calculated as follows:

$$\text{Synergy Score} = \frac{E([A]_X, [B]_Y)}{E([A]_X) \tilde{n} E([B]_Y)}$$

where $E([A]_X, [B]_Y)$ denotes the combined drug response of drug A and B at X and Y concentration. $E([A]_X)$ and $E([B]_Y)$ are the single drug response at X and Y concentrations, respectively. Note the drug response should be normalized to its untreated control for the synergy calculation for using the BI model, and thus 1 indicate no drug treatment (ND) or control; 0 indicates the complete inhibition. SynergyScore <1, = 1 or >1 indicates that the drugs are synergistic, additive, or antagonistic, respectively. For instance, if compared to vehicle treatment [1], the tumor growth is decreased to 0.5 and 0.5 by the single-drug treatments, and to 0.1 by the combined treatment, then the Synergy-Score = $0.1/(0.5 \times 0.5) = 0.4$, indicating this drug combination has a synergistic effect (Fig. 11a).

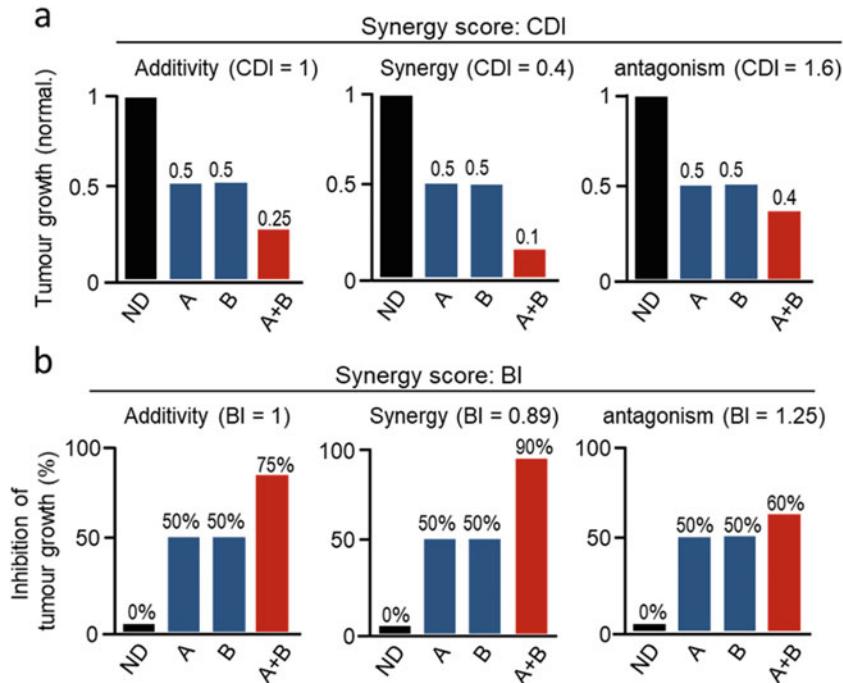


Fig. 11 Calculation of drug synergy score. For CDI (a), drug response (e.g., tumor growth) is normalized to its untreated control. Thus, “1” indicates no drug treatment or control. “0” indicates complete inhibition. For BI (b), the drug effect indicates a percentage inhibition of the response. Thus, 0% denotes an untreated control. 100% indicates complete inhibition

Bliss Independence: BI is another statistical model to assess the combination efficacy of two drugs based on the assumption that the two drugs act through independent mechanisms and thus the individual drugs do not directly interfere with each other [30, 31]. The predicted effect of drug A and B at drug concentrations of X and Y satisfies the following equation:

$$E_{AB}^{\text{Pred}} = E([A]_X) + E([B]_Y) - E([A]_X) \cdot E([B]_Y)$$

where E_{AB}^{Pred} denotes the predicted combined effect of drug A and B at X and Y concentration. $E([A]_X)$ and $E([B]_Y)$ are the single drug effect at X and Y concentrations, respectively. Note that the drug response is indicated by a percent inhibition compared to an untreated control and thus 0 indicate no drug effect (0% inhibition) or control; 1 indicates the complete inhibition (100%). The BI score is calculated as follows:

$$\text{SynergyScore} = \frac{E_{AB}^{\text{Pred}}}{E_{AB}^{\text{Obs}}}$$

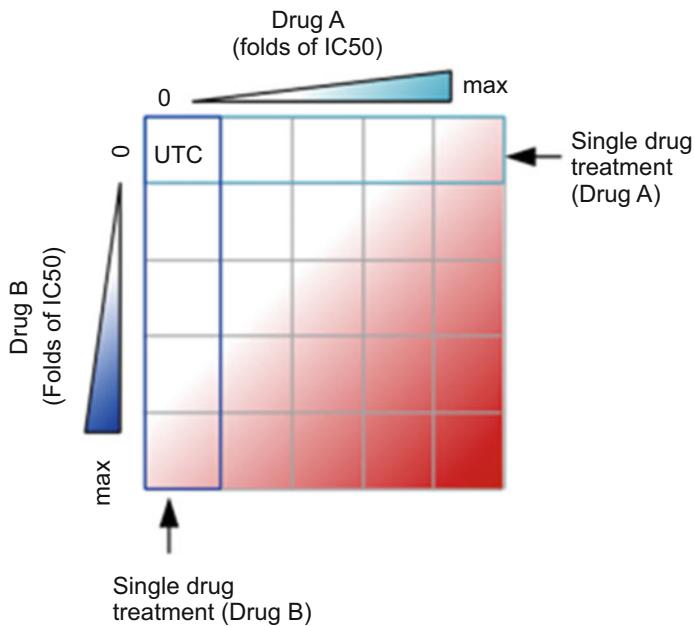


Fig. 12 A dose-response matrix. The first row and column indicate effects of a single drug treatment. The first element of the matrix corresponds to the untreated control (UTC). max: the maximal tolerance concentration

where E_{AB}^{Obs} denotes the observed combined effect of drug A and B at concentration X and Y , respectively. For example, if drug A and B suppressed tumor growth 50% and 50% compared to the control group, then the predicted combined effect would be 75% ($= 0.5 + 0.5 - 0.5 \times 0.5$). If the actual inhibition of tumor growth when combined is greater than 75% (e.g., for 90%, SynergyScore = 0.89), then the compounds would be synergistic. If the tumor inhibition is less than 75% (e.g., for 60%, SynergyScore = 1.25), then the compounds would be defined as antagonistic (Fig. 11b).

5. Drug Combination Setup and Drug Synergy Tools

Because the synergy between two drugs can depend strongly on the specific doses at which they are combined, in order to gain a broad understanding of the synergy landscape, it is good to perform drug combination experiments (computationally or experimentally) at various dose combinations in a “dose-matrix” setup. Shown in Fig. 12, a typical setup involves combination of gradually increasing doses of two drugs A and B, around their respective IC₅₀ values, including single-drug treatments. For example, the drugs could be combined at five doses: 1/5-, 1/2-, one-, two-, and fivefold of the IC₅₀ concentration (Fig. 12). The first row and column of the matrix correspond to the single-drug treatments, and the first element of the matrix corresponds to vehicle control (UTC).

In addition to the CDI and BI methods for computing drug synergy, there are other methods, including Chou-Talalay [32], HSA [33], Loewe [58], and ZIP [34, 35]. A number of software tools are available that allow one to compute drug synergy using the data obtained from the dose-matrix experiment setup above. For instance, SynergyFinder [35] is a web-based application that allows interactive analysis and visualization of drug synergy using a variety of methods, such as HSA [33], Loewe [58], BI [30, 31], and ZIP [34, 35]. A stand-alone implementation of SynergyFinder as an R package is also available [59]. Combenefit is a stand-alone software for Windows [60], which provides calculation of drug synergy using the Loewe, BI, and HAS methods. In addition, CompuSyn is another computer program to determine drug synergy using the median-effect principle of the mass-action law and the combination index theorem [32, 61] as part of the Chou-Talalay method (<https://www.combosyn.com/>).

Acknowledgments

This work was supported by a Victorian Cancer Agency Mid-Career Research Fellowship (MCRF18026), an Investigator Initiated Research Scheme grant from National Breast Cancer Foundation (IIRS-20-094), and a Metcalf Venture Grant by Cancer Council Victoria, Australia, awarded to L.K.N.

Appendix A. Ordinary Differential Equations of the EGFR-MET Model

The reaction rates are given in Appendix B.

Left-hand sides	Right-hand sides	Initial conditions (nM)
$d[p\text{EGFR}]/dt$	$v_1 - v_2$	0.109
$d[\text{EGFRub}]/dt$	$v_3 - v_4$	6.940
$d[\text{PYK2m}]/dt$	$v_5 - v_6$	0.622
$d[\text{PYK2}]/dt$	$v_7 - v_8 - v_9 + v_{10}$	9.299
$d[p\text{PYK2}]/dt$	$v_9 - v_{10}$	2.510
$d[p\text{STAT3}]/dt$	$v_{11} - v_{12}$	1.178
$d[c\text{METm}]/dt$	$v_{13} - v_{14}$	0.023
$d[c\text{MET}]/dt$	$v_{15} - v_{16} - v_{17} + v_{18}$	4.672
$d[pc\text{MET}]/dt$	$v_{17} - v_{18}$	0.502

(continued)

Left-hand sides	Right-hand sides	Initial conditions (nM)
$d[pCbl]/dt$	$v19 - v20$	10.476
$d[aPTP]/dt$	$v21 - v22$	0.494
$d[pERK]/dt$	$v23 - v24$	0.669
$d[STAT3uStatic]/dt$	$v25$	0.00

Appendix B. Reactions and Reaction Rates of the EGFR-MET Network Model

	Reaction	Reaction rates
v1	$EGFR \rightarrow pEGFR$	$kc1 * (EGF/(1 + Gefitinib/Ki1) + caEGF) * EGFR/(Km1 + EGFR)$
v2	$pEGFR \rightarrow EGFR$	$(Vmax2 + kc2 * aPTP) * pEGFR/(Km2 + pEGFR)$
v3	$EGFR \rightarrow EGFRub$	$(Vmax3 + kc3 * pCbl) * EGFR/(Km3 + EGFR) * Ki3a/(Ki3a + PYK2tot/(1 + PF396/Ki3b))$
v4	$EGFRub \rightarrow EGFR$	$Vmax4 * EGFRub/(Km4 + EGFRub)$
v5	$\emptyset \rightarrow PYK2m$	$Vs5 + Vmax5 * pSTAT3/(Km5 + pSTAT3)$
v6	$PYK2m \rightarrow \emptyset$	$kdeg6 * PYK2m$
v7	$PYK2m \rightarrow PYK2$	$Vmax7 * PYK2m/(Km7 + PYK2m)$
v8	$PYK2 \rightarrow \emptyset$	$kdeg8 * PYK2$
v9	$PYK2 \rightarrow pPYK2$	$(kc9a * pEGFR + kc9b * pcMET/(1 + EMD/Ki9)) * PYK2/(Km9 + PYK2)$
v10	$pPYK2 \rightarrow PYK2$	$(Vmax10 + kc10 * aPTP) * pPYK2/(Km10 + pPYK2)$
v11	$STAT3 \rightarrow pSTAT3$	$kc11 * (pPYK2/(1 + PF396/Ki3b)) * STAT3/(Km11 + STAT3)$
v12	$pSTAT3 \rightarrow STAT3$	$(Vmax12 + kc12 * aPTP) * pSTAT3/(Km12 + pSTAT3)$
v13	$\emptyset \rightarrow cMETm$	$Vs13 + Vmax13 * pSTAT3/(Km13 + pSTAT3)$
v14	$cMETm \rightarrow \emptyset$	$kdeg14 * cMETm$
v15	$cMETm \rightarrow cMET$	$Vmax15 * cMETm/(Km15 + cMETm)$
v16	$cMET \rightarrow \emptyset$	$(kdeg16 + kc16 * pCbl) * cMET/(Km16 + cMET)$

(continued)

Reaction	Reaction rates
v17 cMET → pcMET	$(kc17 * HGF + caHGF) * cMET / (Km17 + cMET)$
v18 pcMET → cMET	$Vmax18 * pcMET / (Km18 + pcMET)$
v19 Cbl → pCbl	$kc19 * pEGFR * Cbl / (Km19 + Cbl)$
v20 pCbl → Cbl	$(Vmax20 + kc20 * aPTP) * pCbl / (Km20 + pCbl)$
v21 PTP → aPTP	$kc21 * pEGFR * PTP / (Km21 + PTP)$
v22 aPTP → PTP	$Vmax22 * aPTP / (Km22 + aPTP)$
v23 ERK → pERK	$(kc23a * pcMET / (1 + EMD / Ki23) + kc23b * pEGFR) * ERK / (Km23 + ERK)$
v24 pERK → ERK	$Vmax24 * pERK / (Km24 + pERK)$
v25 STAT3 + Stattic ↔ STAT3uStattic	$ka25 * STAT3 * Stattic - kd25 * STAT3uStattic$

Appendix C. Best-Fitted Parameter Values Used for Simulations

Parameter	Value	Unit	Parameter	Value	Unit
kc1	413.0475	min^{-1}	Km17	9.817479	nM
Km1	248.8857	nM	Vmax18	0.060674	nM min^{-1}
Ki1	1.0000	μM	Km18	9.954054	nM
kc2	1406.048	min^{-1}	kdeg16	24.49063	min^{-1}
Km2	3.801894	nM	kc16	1.174898	min^{-1}
Vmax3	0.000104	nM min^{-1}	Km16	528.4453	nM
Km3	2.285599	nM	kc19	52.72299	min^{-1}
Ki3a	0.08356	nM	Km19	13.30454	nM
Ki3b	1.0000	nM	kc20	35.64511	min^{-1}
Vmax4	11.11732	nM min^{-1}	Km20	24.32204	nM
kc3	10.78947	min^{-1}	kc21	0.003972	min^{-1}
Km4	90.78205	nM	Km21	52.72299	nM
Vs5	26.54606	nM min^{-1}	Vmax22	0.034914	nM min^{-1}
Vmax5	34.04082	nM min^{-1}	Km22	46.45153	nM
Km5	4.74242	nM	Vmax2	112.2018	nM min^{-1}
kdeg6	53.57967	min^{-1}	Vmax12	7.638358	nM min^{-1}

(continued)

Parameter	Value	Unit	Parameter	Value	Unit
Vmax7	3.349654	nM min ⁻¹	Vmax20	0.048306	nM min ⁻¹
Km7	3.334264	nM	kc23a	7.03E + 09	min ⁻¹
kc9a	0.463447	min ⁻¹	kc23b	8.43E + 08	min ⁻¹
kc9b	0.988553	min ⁻¹	Km23	2.831392	nM
Km9	34.91403	nM	Vmax24	4.4E + 09	nM min ⁻¹
Vmax10	0.530884	nM min ⁻¹	Km24	0.156675	nM
Km10	9.141132	nM	kc10	0.006109	min ⁻¹
kdeg8	0.056624	min ⁻¹	Ki9	1.65577	nM
kc11	0.321366	min ⁻¹	Ki23	13.48963	nM
Km11	20.6063	nM	ka25	127.3503	μM ⁻¹ min ⁻¹
kc12	0.00029	min ⁻¹	kd25	11.74898	min ⁻¹
Km12	11.58777	nM	caEGF	0.089125	nM
Vs13	0.093756	nM min ⁻¹	caHGF	0.009036	nM
Vmax13	0.354813	nM min ⁻¹	EGFRtot	398.1072	nM
Km13	38.72576	nM	STAT3tot	144.2115	nM
kdeg14	4.560369	min ⁻¹	Cbltot	174.9847	nM
Vmax15	91.41132	nM min ⁻¹	PTPtot	296.4831	nM
Km15	6.456542	nM	ERKtot	166.7247	nM
kc17	0.000811	min ⁻¹			

References

1. Labrie M, Brugge JS, Mills GB, Zervantakis IK (2022) Therapy resistance: opportunities created by adaptive responses to targeted therapies in cancer. *Nat Rev Cancer* 22(6):323–339
2. Cremer CG, Nguyen LK (2019) Network rewiring, adaptive resistance and combating strategies in breast cancer. *Cancer Drug Resist* 2(4):1106–1126
3. Jaaks P et al (2022) Effective drug combinations in breast, colon and pancreatic cancer cells. *Nature* 603(7899):166–173
4. Kolch W, Halasz M, Granovskaya M, Kholodenko BN (2015) The dynamic control of signal transduction networks in cancer cells. *Nat Rev Cancer* 15(9):515–527
5. Romano D et al (2014) Protein interaction switches coordinate Raf-1 and MST2/Hippo signalling. *Nat Cell Biol* 16(7):673–684
6. Roskoski R Jr (2021) Properties of FDA-approved small molecule protein kinase inhibitors: a 2021 update. *Pharmacol Res* 165:105463
7. Ghomlaghi M, Hart A, Hoang N, Shin S, Nguyen LK (2021) Feedback, crosstalk and competition: ingredients for emergent non-linear behaviour in the PI3K/mTOR signalling network. *Int J Mol Sci* 22(13):6944
8. Nguyen LK, Kholodenko BN (2016) Feedback regulation in cell signalling: lessons for cancer therapeutics. *Semin Cell Dev Biol* 50:85–94
9. Montagud A et al (2022) Patient-specific Boolean models of signalling networks guide personalised treatments. *elife* 11:e72626
10. Frank TD, Cavadas MAS, Nguyen LK, Cheong A (2016) Non-linear dynamics in transcriptional regulation: biological logic gates. In: Carballido-Landeira J, Escribano B (eds)

- Nonlinear dynamics in biological systems. Springer International Publishing, Cham, pp 43–62
11. Toni T, Stumpf MPH (2009) Simulation-based model selection for dynamical systems in systems and population biology. *Bioinformatics* 26(1):104–110
 12. Aldridge BB, Saez-Rodriguez J, Muhlich JL, Sorger PK, Lauffenburger DA (2009) Fuzzy logic analysis of kinase pathway crosstalk in TNF/EGF/insulin-induced signaling. *PLoS Comput Biol* 5(4):e1000340
 13. Shin S-Y, Müller A-K, Verma N, Lev S, Nguyen LK (2018) Systems modelling of the EGFR-PYK2-c-Met interaction network predicts and prioritizes synergistic drug combinations for triple-negative breast cancer. *PLoS Comput Biol* 14(6):e1006192
 14. Nguyen LK et al (2013) A dynamic model of the hypoxia-inducible factor 1α (HIF-1α) network. *J Cell Sci* 126(Pt 6):1454–1463
 15. Bouhaddou M et al (2018) A mechanistic pan-cancer pathway model informed by multi-omics data interprets stochastic cell fate responses to drugs and mitogens. *PLoS Comput Biol* 14(3):e1005985
 16. Ghomlaghi M, Yang G, Shin S-Y, James DE, Nguyen LK (2021) Dynamic modelling of the PI3K/MTOR signalling network uncovers biphasic dependence of mTORC1 activity on the mTORC2 subunit SIN1. *PLoS Comput Biol* 17(9):e1008513
 17. Shin SY, Nguyen LK (2017) Dissecting cell-fate determination through integrated mathematical modeling of the ERK/MAPK signaling pathway. *Methods Mol Biol* 1487:409–432
 18. Tyson JJ et al (2011) Dynamic modelling of oestrogen signalling and cell fate in breast cancer cells. *Nat Rev Cancer* 11(7):523–532
 19. Chowdhury S, Sarkar RR (2015) Comparison of human cell signaling pathway databases—evolution, drawbacks and challenges. *Database* 2015:bau126
 20. Gillespie M et al (2021) The reactome pathway knowledgebase 2022. *Nucleic Acids Res* 50(D1):D687–D692
 21. Kanehisa M, Goto S (2000) KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res* 28(1):27–30
 22. Licata L et al (2020) SIGNOR 2.0, the SIGNALING Network Open Resource 2.0: 2019 update. *Nucleic Acids Res* 48(D1):D504–d510
 23. Martens M et al (2021) WikiPathways: connecting communities. *Nucleic Acids Res* 49(D1):D613–d621
 24. Malik-Sheriff RS et al (2020) BioModels-15 years of sharing computational models in life science. *Nucleic Acids Res* 48(D1):D407–d415
 25. Villaverde AF, Pathirana D, Fröhlich F, Hasenauer J, Banga JR (2022) A protocol for dynamic model calibration. *Brief Bioinform* 23(1):bbab387
 26. Wentur CJ, Gentry PR, Mathews TP, Lindsay CW (2014) Drugs for allosteric sites on receptors. *Annu Rev Pharmacol Toxicol* 54: 165–184
 27. Seabaugh JL (2011) Guidelines for accurate EC50/IC50 estimation. *Pharm Stat* 10(2): 128–134
 28. Li X et al (2016) β-elemene sensitizes hepatocellular carcinoma cells to oxaliplatin by preventing oxaliplatin-induced degradation of copper transporter 1. *Sci Rep* 6:21010
 29. Liu F, Shang Y, Chen S-z (2014) Chloroquine potentiates the anti-cancer effect of lidamycin on non-small cell lung cancer cells in vitro. *Acta Pharmacol Sin* 35(5):645–652
 30. Keith CT, Borisy AA, Stockwell BR (2005) Multicomponent therapeutics for networked systems. *Nat Rev Drug Discov* 4(1):71–78
 31. Zhao W et al (2014) A new bliss Independence model to analyze drug combination data. *J Biomol Screen* 19(5):817–821
 32. Chou TC (2010) Drug combination studies and their synergy quantification using the Chou-Talalay method. *Cancer Res* 70(2): 440–446
 33. Berenbaum MC (1989) What is synergy? *Pharmacol Rev* 41(2):93–141
 34. Yadav B, Wennerberg K, Aittokallio T, Tang J (2015) Searching for drug synergy in complex dose-response landscapes using an interaction potency model. *Comput Struct Biotechnol J* 13:504–513
 35. Ianevski A, Giri AK, Aittokallio T (2020) SynergyFinder 2.0: visual analytics of multi-drug combination synergies. *Nucleic Acids Res* 48(W1):W488–W493
 36. Arab-Bafrani Z, Shahbazi-Gahrouei D, Abbasian M, Fesharaki M (2016) Multiple MTS assay as the alternative method to determine survival fraction of the irradiated HT-29 colon cancer cells. *J Med Signals Sens* 6(2): 112–116
 37. Sellés Vidal L, Kelly CL, Mordaka PM, Heap JT (2018) Review of NAD(P)H-dependent oxidoreductases: properties, engineering and application. *Biochim Biophys Acta Proteins Proteom* 1866(2):327–347
 38. Franken NAP, Rodermond HM, Stap J, Haveman J, van Bree C (2006) Clonogenic

- assay of cells in vitro. *Nat Protoc* 1(5): 2315–2319
39. Hanson KM, Finkelstein JN (2019) An accessible and high-throughput strategy of continuously monitoring apoptosis by fluorescent detection of caspase activation. *Anal Biochem* 564–565:96–101
 40. Kho D et al (2015) Application of xCELLigence RTCA biosensor technology for revealing the profile and window of drug responsiveness in real time. *Biosensors* 5(2): 199–222
 41. Nakai K, Hung MC, Yamaguchi H (2016) A perspective on anti-EGFR therapies targeting triple-negative breast cancer. *Am J Cancer Res* 6(8):1609–1623
 42. Verma N et al (2017) Targeting of PYK2 synergizes with EGFR antagonists in basal-like TNBC and circumvents HER3-associated resistance via the NEDD4–NDRG1 axis. *Cancer Res* 77(1):86–99
 43. Ho-Yen CM, Jones JL, Kermorgant S (2015) The clinical and functional significance of c-Met in breast cancer: a review. *Breast Cancer Res* 17(1):52
 44. Linklater ES et al (2016) Targeting MET and EGFR crosstalk signalling in triple-negative breast cancers. *Oncotarget* 7(43): 69903–69915
 45. Qi J et al (2011) Multiple mutations and bypass mechanisms can contribute to development of acquired resistance to MET inhibitors. *Cancer Res* 71(3):1081–1091
 46. Shin S-Y et al (2014) The switching role of β-adrenergic receptor signalling in cell survival or death decision of cardiomyocytes. *Nat Commun* 5(1):5777
 47. Shin D et al (2014) The hidden switches underlying ROR α -mediated circuits that critically regulate uncontrolled cell proliferation. *J Mol Cell Biol* 6(4):338–348
 48. Masuda H et al (2012) Role of epidermal growth factor receptor in breast cancer. *Breast Cancer Res Treat* 136(2):331–345
 49. Zhu Z et al (2021) Eucannabinolide, a novel sesquiterpene lactone, suppresses the growth, metastasis and BCSCS-like traits of TNBC via inactivation of STAT3. *Neoplasia* 23(1):36–48
 50. Chaudhary SS et al (2020) Chapter 11 – c-Met as a potential therapeutic target in triple negative breast cancer. In: Gupta SP (ed) *Cancer-leading proteases*. Academic, San Diego, California, United States, pp 295–326
 51. Pedersen MW, Pedersen N, Ottesen LH, Poulsen HS (2005) Differential response to gefitinib of cells expressing normal EGFR and the mutant EGFRvIII. *Br J Cancer* 93(8):915–923
 52. Han S et al (2009) Structural characterization of proline-rich tyrosine kinase 2 (PYK2) reveals a unique (DFG-out) conformation and enables inhibitor design. *J Biol Chem* 284(19): 13193–13201
 53. Dussault I, Bellon SF (2009) From concept to reality: the long road to c-Met and RON receptor tyrosine kinase inhibitors for the treatment of cancer. *Anti Cancer Agents Med Chem* 9(2): 221–229
 54. Allen JV et al (2011) The discovery of benzamilides as c-Met receptor tyrosine kinase inhibitors by a directed screening approach. *Bioorg Med Chem Lett* 21(18):5224–5229
 55. Schust J, Sperl B, Hollis A, Mayer TU, Berg T (2006) Stattic: a small-molecule inhibitor of STAT3 activation and dimerization. *Chem Biol* 13(11):1235–1242
 56. Shin S-Y et al (2021) Integrative modelling of signalling network dynamics identifies cell type-selective therapeutic strategies for FGFR4-driven cancers. *bioRxiv*:2021.2011.2003.467180
 57. Frohlich F et al (2018) Efficient parameter estimation enables the prediction of drug response using a mechanistic pan-cancer pathway model. *Cell Syst* 7(6):567–579 e566
 58. Loewe S (1953) The problem of synergism and antagonism of combined drugs. *Arzneimittelforschung* 3(6):285–290
 59. Zheng S et al (2022) SynergyFinder plus: toward better interpretation and annotation of drug combination screening datasets. *J Pharmacokinet Pharmacodyn* 20(3):587–596. <https://doi.org/10.1016/j.jpb.2022.01.004>
 60. Di Veroli GY et al (2016) Combbenefit: an interactive platform for the analysis and visualization of drug combinations. *Bioinformatics* 32(18):2866–2868
 61. Chou TC (2006) Theoretical basis, experimental design, and computerized simulation of synergism and antagonism in drug combination studies. *Pharmacol Rev* 58(3):621–681

INDEX

B

- Biochemical networks
complexity 3–31
Bioinformatics 109
Biological processes
cellular differentiation 218–221, 230
diffusion coefficient 193, 194, 210
DNA double strand breaks 268, 271, 281
genotoxic stress 268, 271
glucose-insulin interaction 88–94
ligand 201
paradoxical activation 66–68
receptor 201, 286
stress response 218–219, 221
ultradian endocrine model 88–90, 94
Biological robustness 217, 227, 229–230

C

- Cancer
breast cancer 286, 309, 337–353, 359, 366
Cdk4/6 inhibition 350
endocrine therapy 339, 349
ER+ breast cancer 339, 353
palbociclib 339, 340, 344, 349, 350

- Cell-cell communication
co-culture signalling 285–312
co-culturing 285–312
receiver cells 285, 287, 289,
290, 296, 298, 300, 301, 304, 306, 310, 311
sender cells 304–306, 310, 312
sender-receiver system 286

- Cell fate 52, 153, 163, 217,
219, 221, 222, 226, 228, 230, 234, 244, 267,
285, 286, 315–325

- Cell lines
DT40 cells 257, 262, 264
embryonic stem cell (ESC) 157
293FT cells 290, 296
HMT-3522 cell 290, 297, 299, 309
MCF-7 cells 338–340, 346, 350
NMP cells 155
S1 cells 290, 295
T4-2 cells 290, 295
- Cell signalling v, vi, 192, 285, 287,
300, 306, 357–379

- Cell signal transduction 4, 29, 59–82
Cellular behaviour 222, 234–246
Cellular decision making 218, 229
Cellular heterogeneity
cell-to-cell variability 218, 219, 222, 224
extrinsic noise 154, 156–159, 161–163
fluctuation 154, 157, 161, 163,
218, 221, 224–226, 229–234, 244
gene expression heterogeneity 231–233,
244–246
intrinsic noise 161, 162
protein expression heterogeneity 173–175
- Chemical reaction networks 361
Combination therapy 351
Community-based metabolite potential
(CMP) 145, 146
- Computational biology 134
Computational modelling v, vi, 60,
109, 123, 135, 286, 330–332
- Computational random mutagenesis 329–333
- CRISPR-Cas9 293
- CRISPR gene editing 295, 307
- CRISPR gene tagging 293
- Cross-pathway links 140

D

- Data analysis 135, 142–146,
256–258, 261–264, 331
- Data visualisation
parallel coordinate plots 35, 42
parallel coordinates 36, 55, 57
- Design principle 3–31, 229–230
- Diagrammatic method 5, 6, 11–14, 19–29
- Dose-response curves
IC₅₀ values 362, 367, 375
- Drug combinations 357–379
- Drug resistance 182, 218, 220, 366
- Drug screening 288
- Drug synergy
Bliss Independence (BI) 364, 373
Chou-Talay model 373
Coefficient of Drug Interaction (CDI) 364, 373
CompuSyn 376
highest single agent (HSA) 364, 373
synergy score 364, 368–371, 373, 374

- Drug synergy (*cont.*)
 SynergyFinder 376
 zero interaction potency (ZIP) 364, 373
- Dynamical systems 90
- F**
- FAIR principles 133, 136
 Feedback 5, 7, 15, 19, 23, 42,
 43, 51–55, 60, 67, 128, 176, 177, 191, 221, 229,
 231, 235, 236, 239–241, 243, 245, 268, 318,
 319, 325, 349, 350
- Feedback loops 4, 8, 11–16, 19, 24, 26,
 28, 42, 51, 52, 172, 173, 226, 242, 349, 358
- Feedback regulation 19, 180
 Fluorescence lifetime imaging microscopy
 (FLIM) 287
- Fluorophore 237, 287, 288, 316
- Förster resonance energy transfer (FRET) 286,
 287, 307, 309
- FRET reporters 287
- G**
- Genetically-encoded reporters 286, 287,
 291, 293, 307
- Goodwin system 42–51
- I**
- In silico drug screening
 SynDISCO 357–379
 treatment prediction 229, 264, 281, 344, 351
 treatment prioritisation 357–379
- Intercellular communication 285
- K**
- Kinase activity 240, 287, 295,
 340, 343, 348–350, 367
- Kinase translocation reporters (KTR) 287, 295
- L**
- Live-cell imaging 221, 222, 234,
 237, 241, 245, 296
- M**
- Mathematical modelling
 conservation law 38, 56, 57, 75
 dynamic modelling 116
 energy-based modelling 60, 61
 mechanistic modelling 218, 233,
 245, 357–379
 metabolic reprogramming 139, 149
 model parameterization 233, 268
 model reduction 38, 56, 244
- model selection 61, 77, 270
 network modelling v, 34, 110, 358
 ODE modelling 60, 61, 169, 170
 ordinary differential equations
 (ODE) 34, 169, 170, 192, 342, 359
 particle-based modelling 192
 population-average models 226–229,
 236, 239–242
 rule-based modelling 59–82
 stochastic modelling 224, 226,
 231–232, 244, 245
 subpopulation-specific modelling 272, 276–280
 thermodynamic constraints 60
 thermodynamic models 62–68
- Meta-dynamic network (MDN) modelling
 model instances 170, 171, 175, 183
 qualitative behaviour descriptors 171
- Microbiome 140, 142, 144, 148, 149
- MicroRNA 147, 161
 miR-296 162, 163
 miRNA-target gene regulatory
 interactions 146
- Microscopy
 fluorescence microscopy 262, 296
 image acquisition 257
 image analysis 237, 258, 302, 307
 image processing 293, 301, 302, 307
 partial least squares regression (PLSR) 312
 reporter cells 297, 299, 309
 spinning disk confocal microscope 264
- MIMOSA 145, 146
- Mixed feedback 51–55
- Model calibration
 experimental validation 366
 fisher information matrix (FIM) 99, 101–104
 model fitting 232
 model identifiability 61, 271
 model training 71
 parameter estimation v, 361
 parameter inference 279
 practical identifiability 88, 99,
 101, 102, 104
 structural identifiability 88, 90–94, 96
- Model construction
 Hill kinetics 361, 366
 mass action kinetics 239, 361
 Michaelis-Menten kinetics 169, 361, 366
 network wiring 359, 372
 schematic diagram 43, 51, 359, 360, 367
- Model repositories
 biomodels 132, 133
- Model reproducibility
 model versioning 133, 134
 version control 131, 132

- M**
- Model simulation 123, 225
 deterministic simulation 123, 225
 experimental validation 227, 370
FLAME-accelerated Signalling Tool
 (FaST) 191–211
FLAME GPU 198–201
Gillespie's stochastic simulation
 algorithm (SSA) 155
GPU 191–211
 GPU-based parallelisation 192
 high-dimensional parameter space 35
 high-dimensional visualisation 33–57
 hyperspace 34, 55
 model prediction 126, 359
 Monte Carlo sampling 36, 126–127, 229
 numerical simulation 227
 parameter variation 178
 random parameter sampling 332
 sensitivity analysis 61, 75, 76, 192, 228
 stochastic simulation 122, 123, 153–164, 226
- M**ulti-omics 139–149
- N**
- Negative feedback 4, 5, 8, 19,
 24, 42–51, 176, 177, 180, 229, 235, 236, 239,
 241, 243, 245, 268
- Network topology 5, 7, 8, 148,
 168, 171, 173, 182, 183, 264
- Neural networks
 deep neural networks 94
 systems-biology informed neural networks
 (SBINN) 88, 94–99, 101, 103, 104
 topologically important nodes (TINs) 146
- Next generation sequencing
 ATAC-seq 255, 264
 ChIP-seq 255, 256, 264
 RNA-seq 264
- O**
- Operational taxonomic unit (OTU) 144–146
- Optogenetics
 LED 316, 322
 optoSOS 315–318, 320, 321, 324, 325
 photoswitching 321, 322
- P**
- Pathway crosstalk v, 358
- Phosphorylation cascade 51–55
- PICRUSt 145
- Positive feedback 19, 42, 51, 254, 349
- Protein dynamics 167, 178, 183, 227
- Protein-protein interaction 140, 141,
 143, 147, 330, 333
- Q**
- QIIME2 144, 145
- Quantitative modelling 222, 232–234, 241–244
- R**
- RightField 117, 124
- Robust perfect adaptation (RPA) 3–30
- S**
- SEEK platform 117, 124
- Signalling crosstalk 172, 267–270
- Signalling dynamics 178, 238
- Signalling interaction v, 278–282, 288, 310
- Signalling outputs 183
- Signalling pathways
 cell cycle 223, 231, 236, 237
 c-Myc 340, 343, 344, 346, 348–350
 DNA synthesis 347
 E2F 340, 341, 347–351
 EGFR-MET signalling crosstalk 365
 EGFR signalling 67
 ERK signalling 171, 173, 177, 300
 G1-S transition 340, 341, 344,
 346, 347, 350
 Hippo-ERK crosstalk 171–173, 181, 183
 Hippo signalling 173
 LAT51/2 172
 MAPK signalling 66, 171,
 178, 221, 222, 226
 MET signalling 366
 MST1/2 172, 178
 NFkB signalling pathway 222
 p53 signaling 271, 272, 283
 PYK2 signalling 366, 367, 369–371, 377
 RAF inhibition 62–68
 RAF signalling 67
 RAS mutant 60, 66, 329–333
 RAS signalling 330
 RB1 340, 343, 346–348, 350, 351
 RB1 phosphorylation 340, 347, 348
 SMAD signalling 234–242, 244–246
 STAT3 signalling 366, 367, 377, 378
 TGF β signalling 217–246
 YAP 172, 176
- Single cells
 single cell analysis 218
 single-cell experiments 227, 237, 243
- Software, tools and databases
 AMICI 61, 74, 75, 78
 Antimony 37, 38, 43, 117, 120, 121
 BioCyc 113
 BioModels 110, 112, 122, 132, 133, 360
 BioNetGen 60, 117

- Software (*cont.*)
- Bitbucket 132
 - BRENDA 113
 - ChEBI 113, 114, 119
 - COMBINE archive 112, 121, 129–131
 - DeepXDE 95, 96, 104
 - DYVIPAC 34, 35, 44–46, 53
 - FAIR 133, 136
 - GitHub 88, 111, 112, 118, 120, 132, 135
 - ImageJ 257, 258, 260–262
 - IntiQuan Tools 170
 - IQM software package 174
 - Julia programming 91, 101, 104
 - Jupyter notebook 34–36, 39–41, 55, 61, 111, 117, 119–121, 123, 126–130
 - Kappa 60, 254
 - KEGG pathway 113, 114, 146
 - libRoadRunner 36, 37, 52, 57, 121–123
 - MATLAB 117, 170, 174, 192, 195, 277, 288, 293, 303, 331, 342, 366
 - Matplotlib 37
 - Numpy 37
 - pandas 37, 45, 46, 49, 50, 54
 - pyDYVIPAC 33–57
 - PyPESTO 61, 74–82
 - PySB 61–68, 117
 - Python 34–37, 45, 55, 61, 74, 95, 104, 111, 115, 117, 119–121, 126, 128, 129, 131, 196, 264, 288, 302
 - RCSB PDB 113, 114
 - Reactome 113, 114, 142
 - Rhea 113, 114
 - R package 258, 262, 376
 - RStudio 257, 261
 - SABIO-RK 113, 114
 - SED-ML 121–123, 129–131
 - systems biology graphical notation (SBGN) 117, 118
- Systems Biology Markup Language (SBML) 35–38, 40, 43, 55, 60, 61, 73–74, 112, 116–122, 124, 126, 128–134
- Tellurium 37, 117, 121, 122, 124, 126, 129–131
- TensorFlow 95, 96
- UniProt 113, 115, 119
- Visual Studios 193, 196, 198, 199, 211
- XDAT 35–37, 40, 45–50, 55, 57
- XML 38, 43, 45, 52, 197, 199–201, 211
- Zenodo 132
- Systems biology vi, 36, 60, 87–104, 111, 116, 117, 119, 135, 226, 233, 371
- Systems dynamics analysis
- bistability 35, 42, 52, 54, 55
 - bistable response 33
 - local stability analysis 35, 36, 41, 55
 - oscillation 45
 - switch-like activation 253
- T**
- Targeted drugs 371
- Targeted therapy 339
- Target inhibition
- allosteric inhibition 362, 363
 - competitive inhibition 362, 363
- Transcription
- burst-like shuttling 238–239, 244
 - Nanog transcription 154–157, 160–164
 - Nanog transcriptional regulatory network 155–157
 - SMAD transcription factors 244
 - transcriptional burst 231
 - transcriptional regulation 254, 361
 - transcription factor 59, 140, 142, 146, 147, 154, 155, 219, 221, 234, 235, 244, 254–257, 262, 264, 340, 343, 345, 347, 350