PROBABILISTIC APPROACHES TO MODELING UNCERTAINTY IN BIOLOGICAL PATHWAY DYNAMICS

BENJAMIN MATE GYORI

NATIONAL UNIVERSITY OF SINGAPORE

2014

PROBABILISTIC APPROACHES TO MODELING UNCERTAINTY IN BIOLOGICAL PATHWAY DYNAMICS

BENJAMIN MATE GYORI

(*B.Sc.*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

NUS GRADUATE SCHOOL FOR INTEGRATIVE SCIENCES AND ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2014

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Benjamin Mate Gyori

July 10, 2014

Acknowledgement

First and foremost, I would like to thank my supervisor David Hsu. He was a mentor whose enthusiasm and curiosity in computer science research inspired and motivated me. I greatly appreciate his support and guidance through these years. I owe much gratitude to P.S. Thiagarajan for involving me in a series of exciting projects, connecting me with collaborators and giving me valuable advice. I would like to thank both professors for offering me generous support during the last months of my candidacy.

I would like to thank Jeremy Gunawardena and Peter Sorger, who invited me to the Department of Systems Biology at Harvard Medical School. I am grateful for the amazing people I had a chance to work and interact with at Harvard, including Tathagata Dasgupta, Mingsheng Zhang, Sudhakaran Prabakaran, Mohan Malleshiah, Marc Hafner, Mohammad Fallahi-Sichani, Somponnat Sampattavanich and Daniel Gibson.

I would also like to thank Gireedhar Venkatachalam and Marie-Veronique Clement for our fruitful collaboration. Special thanks to my friend and collaborator Daniel Paulin, with whom it was a great pleasure to think and work together.

My research at the National University of Singapore was made possible by the scholarship provided by the NUS Graduate School for Integrative Sciences and Engineering. From the NGS department I would specifically like to thank Irene Chua and Ho Wei Min for all their help. The interdisciplinary mindset encompassed by this department has enabled me to venture into the domain of computational systems biology.

I appreciate the support from my peers in the Computational Biology Lab at NUS, including Liu Bing and Sucheendra Palaniappan, who taught me a lot about systems biology and whom I still have the pleasure to work with; Tsung-Han Chiang, who first welcomed me to the lab and helped me settle in; Wang Yue, a good friend who brightened my days; and also Chuan Hock Koh, Jing Quan Lim, Wilson Goh, Hufeng Zhou, Ratul Saha, R. Ramanathan and Soumya Paul.

I also thank my former supervisors Ferenc Vajda and Andras Recski at the Budapest University of Technology and Economics, and Tobias Gindele at the Karlsruhe Institute of Technology for their guidance during my early days in research.

I am dedicating this thesis to my family for all their care and encouragement. This would surely not have been possible without their support. Last but not least I would like to express my gratitude to Claire Lee for her love and support during my PhD years.

Contents

Summary

The behavior of biological cells is governed by a multitude of pathways which coordinate processes including metabolism, gene regulation and signaling. The list of elements and connections between them are often identified, but less is known about the temporal dynamics of pathways. Many of the important functions including the cell cycle, cell proliferation and programmed cell death can only be understood through dynamics. Due to the size and complexity of the networks and their non-linear dynamics, quantitative models are essential in representing pathways and making predictions. When modeling pathway dynamics, one has to capture and make predictions with respect to several sources of uncertainty including molecular noise, cell-to-cell variability, and the fact that typically only noisy and partial measurements are available.

The first part of this thesis focuses on parameter uncertainty in ordinary differential equation models. Due to the sparsity of measurement data, model parameters are commonly under-constrained, and choosing a single best estimate is often inadequate for further analysis. We pose the parameter estimation problem as that of Bayesian inference, where the uncertainty of the parameter values is characterized by a posterior probability distribution. This allows us to consider a variety of possible behaviors consistent with data when making predictions. Particle filters can sequentially approximate posterior probability distributions, however, they suffer from practical issues such as sample impoverishment. We provide an enhanced particle filter that improves sample diversity while preserving the parameter posterior. Our case studies show that using a given number of samples, the proposed method is better than previously used particle filters in making accurate predictions under model uncertainty.

It is important to know that the qualitative and quantitative properties of pathway models hold under model uncertainty. Using statistical model checking (SMC) it is possible to verify whether a system meets a behavior specified in temporal logic with at least a given probability. Standard SMC approaches rely on simulating independent realizations of the dynamics, but this is not possible when dealing with a Bayesian posterior distribution. We propose a method for performing model checking in this setting based on a sequence of dependent samples obtained from a Markov chain. Case studies on the JAK-STAT pathway, and a large model of extrinsic apoptosis demonstrate the practical usefulness of the approach.

If elements of interest don't directly interact with each other in a pathway, building mechanistic ODE models is not a realistic option. Probabilistic graphical models can represent influences among elements of interest and capture the uncertainty arising

from unmodeled components. We propose a method for learning the parameters of a dynamic Bayesian network (DBN) model. Linear programming is used to fit variables to experimental data while satisfying monotonicity constraints corresponding to activation and repression. The method scales well for large pathways due to the local nature of parametrization. Having learned a DBN model, we use probabilistic inference to make predictions about dynamics. We are able to monitor if a specified behavior is met using model checking, and this allows us to identify combinations of perturbations that result in desired behavior. Our method is then used to model novel experimental data for the phosphorylation of 12 key proteins involved in liver cancer progression on 4 relevant cell lines. We are able to predict the response of diseased cells to perturbation combinations and identify ones that modify the dynamics of certain proteins to mimic their dynamics in healthy cells.

List of Figures

List of Tables

Chapter 1

Introduction

Biology studies life from the level of molecules up to whole organisms and beyond. More than a century of research on the cell, the basic unit of life, has shed light on many of the fundamental processes governing living organisms.

Much of the recent progress has been driven by novel experimental technologies. Methods such as the polymerase chain reaction, microarray technology, flow cytometry and fluorescence microscopy have all contributed significantly to our understanding of cellular components. These technologies have enabled the collection of vast amounts of data and induced a change towards a systems approach in biology.

Classical approaches in biology have focused on the precise characterization of individual components. One problem with this approach is that the same molecular entity may be simultaneously involved in several higher level functional roles through *interactions* with other elements. Therefore it is unlikely that higher level cellular processes can be understood only through studying elements in isolation. This, coupled with the availability of experimental technologies to measure several components simultaneously, has lead to the emergence of the *systems* approach in biological research.

Systems biology concentrates on the network level understanding of cellular components including genes, RNA molecules and proteins [1]. Networks of interacting components which are responsible for some cellular function are often called *pathways*. While the connectivity structure of several canonical and disease specific pathways has been studied, less is understood about the temporal *dynamics* of the associated processes. There are many examples, where a list of components or even the structure of interactions between them is not enough to explain important cellular processes. For instance,

upon DNA damage, the decision between cell survival and cell death depends on the pulsating or prolonged activation of the protein p53 [2].

Due to the size and complexity of the pathways and the non-linearity of the dynamics, computational models are essential for the understanding of biology at the systems level. Models have predictive power and offer a coherent basis for depositing and sharing biological knowledge. They can also be used to generate hypotheses and design useful experiments, thereby reducing the need for costly and time consuming wet-lab experiments.

The ability to predict behavior under targeted perturbations using computational models could have an enormous impact. The cost of developing new drugs is growing dramatically and many proposed compounds fail at later stages of approval, often because they do not work as expected. Modeling could be the missing link from traditional drug design to a pathway-level, systemic understanding of drug effects. This could make developing new drugs cheaper and more reliable, and would help to identify which treatment is most likely to result in a good outcome for patients with a specific instance of a disease.

To achieve these goals, new computational methods are needed to efficiently construct and use quantitative pathway models. The research described in this thesis is meant to contribute to this goal.

1.1 Context and motivation

One of the key challenges faced by modeling efforts is to capture uncertainty in biological systems. It is increasingly accepted that noise and variability is an inherent and fundamental aspect of biological systems rather than an additive nuisance [3]. In addition to this, we are often limited to partial, inaccurate and often indirect observation about biological systems. These effects result in *uncertainty* in model based predictions.

Quantitative computational models of pathway dynamics play an increasingly important role in modern biology. Biological pathways are often modeled using ordinary differential equations (ODEs) [4]. The initial conditions and kinetic rate constants (together called model parameters) are commonly unknown and therefore the model is subject to considerable uncertainty. A standard approach involves using an optimiza-

tion procedure to find a single nominal set of parameter values. Models along with the nominal parameter set are often published and deposited in repositories such as the BioModels database [5]. However, this approach has important limitations because there are often several points or regions of parameter space which explain the experimental data equally well. These parameter values could otherwise correspond to very different model behaviors.

One explanation for the under-constrained nature of ODE model parameters is that parameters are often functionally related and there is a large amount of parametric redundancy due to the evolved nature of the underlying networks. Further, the system can only be observed partially and at a low time resolution. Observations are invariably subject to noise due to cellular variability and the measurement process itself. These all contribute to pathway model parameters being unidentifiable [6]. There is also evidence that even large amounts of ideal time-series data can leave parameters poorly constrained [7, 8]. These factors lead to model uncertainty (Figure 1.1), and one primary motivation of our work is to develop methods to deal with this.

By adopting a probabilistic framework and posing the ODE parameter estimation problem as one of Bayesian inference, we can embrace model uncertainty by (i) explicitly modeling it and (ii) making predictions with respect to it. Prior knowledge can also be exploited in a straightforward manner [9]. However, designing efficient inference methods is a major challenge in the context of pathway models with high-dimensional parameter spaces motivating novel computational methods.

When the modeling goal is to capture overall characteristics in signaling for a certain cell type or in a given disease condition, it is useful to only measure and model a limited but representative subset of elements. The existence of missing components results in a special instance of model uncertainty, and detailed kinetic models (such as ones based on ODEs) are of limited usefulness in this context. Graphical models can capture indirect effects between elements, and account for missing components and other sources of uncertainty through assuming probabilistic relationships between them. The use of graphical models in systems biology (including Bayesian networks and dynamic Bayesian networks) has mostly been limited to structure learning, both in the context of gene regulatory networks [10] and signaling pathways [11, 12]. There is great potential in using dynamic Bayesian networks as *predictive* dynamical models.

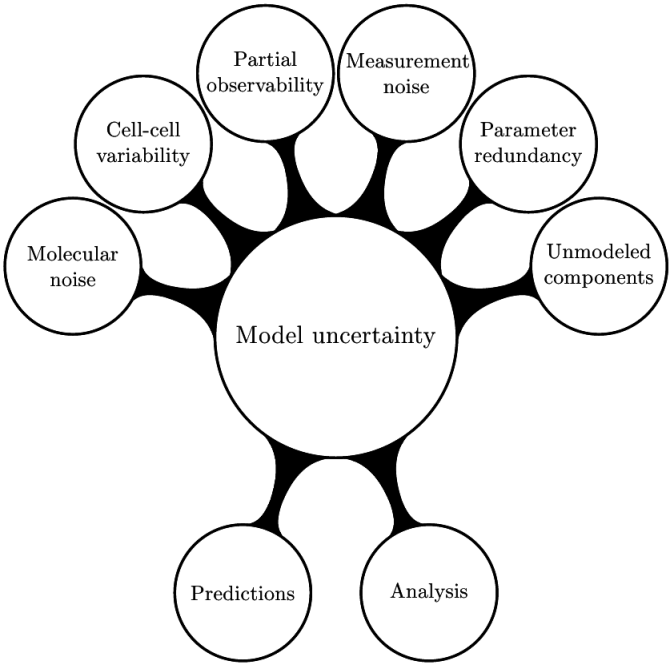


Figure 1.1: Sources of uncertainty in biological pathway models.

1.2 Research contributions

1.2.1 Efficient Bayesian inference of pathway parameters

Dynamical pathway models contain a number of parameters including kinetic rate constants and initial conditions. Since these parameters generally cannot be measured directly, their values have to be inferred from noisy measurement data. Optimization based parameter estimation approaches cannot account for overall parameter uncertainty. Conversely, in a Bayesian probabilistic framework the quantification of the parameter uncertainty becomes possible. However, the reconstruction of the Bayesian posterior distribution is a highly challenging task.

We propose an enhanced particle filtering method to address some of the practical issues encountered in this process (Chapter 3). Particle filters propagate parameter samples forward in time and assimilate experimental data sequentially as weights on the particles. In order to concentrate samples in high-probability areas, resampling is done, but this often leads to sample impoverishment [13]. The solution proposed here involves designing particle transitions on the parameter space using Markov kernels. Applying the Markov transition kernel on a (possibly collapsed) set of samples introduces diversity and results in a more faithful posterior representation. The quality of the posterior is assessed through the accuracy of predictions made using it, and is compared against

other, previously proposed particle filters. The methods are evaluated on a model of the JAK-STAT signaling pathway, and show that kernel-enhanced filters can reach high accuracy with significantly reduced sample size.

1.2.2 Verification of pathway dynamics under Bayesian uncertainty

Model checking is a widely used technique for automatically verifying properties of biological pathways. ODE models with a component of uncertainty are difficult to verify using model checking due to the continuity of the state space and the fact that their solutions are not available in closed form. This has motivated the use of statistical model checking techniques, which rely on sampling independent realizations of the dynamics. The assumption that samples need to be independent has thus far prevented the use of statistical model checking schemes on Bayesian parameter posteriors, since in this case, independent sampling is not possible.

We propose a novel methodology and the theoretical foundations for performing statistical model checking on ODE models characterized by a Bayesian parameter posterior (Chapter 4). The key idea is to construct a Markov chain on the parameter space of the model, which produces a sequence of dependent parameter samples from the posterior. Each sample corresponds to a realization of the system, which is then verified using a model checker. Due to the dependency of samples, it is challenging to decide how many samples are needed to complete the model checking task with a given precision. In our previous work [14], we proposed practically applicable sample size bounds for Markov chain Monte Carlo estimates. Here we derive a form of these bounds applicable to statistical model checking. This allows us to design a fix sample size and an adaptive sample size (sequential) algorithm for performing statistical model checking. We first verify properties on a model of the JAK-STAT signaling pathway. We then consider the EARM model of apoptosis with 71 unknown parameters and very limited experimental data, and show that some important qualitative properties of the model are preserved, while others cannot be verified to hold with high probability due to substantial parameter uncertainty.

1.2.3 Learning dynamic Bayesian network models of pathway dynamics

Probabilistic graphical models provide a succinct representation of stochastic pathway dynamics. They are especially well suited in case an exact physical interaction between elements does not exist or is unknown. Dynamic Bayesian networks (DBNs) have the capability of dealing with temporal data and (in contrast with static Bayesian networks) can model feedback loops. Previous research in using dynamic Bayesian networks in biology has concentrated on inferring the structure of pathways. Less attention has been given to learning and predicting dynamics. Learning pathway dynamics using discrete DBN models has been proposed before but it requires an existing ODE model to fill conditional probability parameters [15]. Here we propose a method to learn the DBN parameters directly from experimental data. We incorporate prior knowledge about the nature of interactions (activation or inhibition) in the form of constraints. We then solve a series of linear programming problems, one for each time point, to learn the conditional probability parameters from data. The method is scalable in the sense that the size of the optimization problem is locally exponential but scales linearly with the total number of nodes. The learned DBN model can be used to make predictions under previously unseen conditions. We learn DBN models based on experimental data collected for 4 cell lines covering stages from healthy to late stage liver cancer. Using approximate inference on the learned DBN models, we can predict time course behavior under various treatments including signaling ligands and small molecule drugs. We are able to find promising combinations of kinase inhibitors that transform some dynamical properties of diseased cells to mimic those of healthy cells.

1.3 Outline

The rest of this thesis is organized as follows. In Chapter 2 we provide an overview of relevant concepts and methods used in modeling biological pathways. This includes modeling formalisms, parameter estimation techniques and model checking methods. Chapter 3 discusses the kernel-enhanced particle filtering method. We show that the method outperforms previously proposed particle filters for the Bayesian inference of pathway parameters. In Chapter 4 we present our method to perform statistical model

checking on biological pathways whose parameters are characterized by a Bayesian posterior distribution. We present both fix sample size and adaptive sample size algorithms and provide sample size bounds for both. Chapter 5 presents a method to learn dynamic Bayesian network models of pathway dynamics. Using inference on the learned model we are able to predict behavior under various stimuli and perturbations. We learn cell type specific models for four cell lines from different stages of liver cancer and obtain insights about their behavior under previously unseen perturbations using the proposed method. Chapter 6 summarizes the contributions of the thesis and discusses promising directions for future research.

Chapter 2

Preliminaries and Background

The immense complexity present in biochemical networks, along with the rapid development of experimental techniques has sparked interest in quantitative modeling approaches in biology. In this chapter we briefly review the biological foundations of pathways and the relevant concepts behind modeling them.

2.1 Biological pathways

2.1.1 Genes to proteins and cellular function

The genetic code is stored in the DNA which is built up of a sequence of nucleotide bases. Through the process of transcription, portions of the DNA sequence called genes are read and copied to a messenger RNA (mRNA) molecule. Transcription starts at a special segment of the DNA called a promoter and ends when a terminator sequence is met. Each mRNA molecule contains one or more protein coding regions which is translated to a sequence of complementary tRNA (transfer RNA) molecules. Finally, the amino acids carried by tRNA are linked to form a protein. The primary structure of proteins is defined by the sequence in which the amino acid molecules are linked. However, it is only after folding into a dedicated three dimensional structure that the protein can properly fulfill its function inside the cell.

Proteins play a principal role in executing the cellular behavior specified by the genetic code. Structural proteins form the cytoskeleton, which maintains the shape and size of the cell. Proteins contain special binding sites which allow them to form complexes with other proteins or bind small molecules. Enzymes catalyze specific chemical

reactions by binding substrate molecules and transforming them into products. Without enzymes, most chemical reactions would occur at a very slow rate, making the cell dysfunctional. Protein molecules are also involved in relaying external or internal signals, essential in reacting to environmental cues. DNA binding proteins, referred to as transcription factors can bind to the promoter region of a gene to influence the speed at which the gene is transcribed.

As we see from these examples, an understanding of how proteins work and interact is of crucial importance towards discovering how cells function.

2.1.2 Pathway types

Cellular behavior is attained through a complex system of chemical reactions. Molecules constantly collide, bind and are transformed into other molecules. We will call molecules of the same type as a molecular *species*. Due to their structure and physical properties, certain species are likely to interact only with a limited set of other species. The interactions between species can be thought of as links in a large network. Current biological knowledge is far from completely mapping out interactions in this network. It is more reasonable to concentrate investigations on sub-networks of restricted scope which can be linked to a specific function. These sub-networks are commonly referred to as pathways.

Biological pathways are generally classified in three distinct groups. In reality, these pathways coexist and interact, however, for purposes of biological understanding, it is useful to discuss them separately. While the main focus of this thesis is signaling pathways, the methods and results will be applicable in a straightforward manner to metabolic and gene regulatory networks as well.

Signal transduction pathways

Signal transduction enables cells to sense environmental cues and respond to them. Signal transduction pathways are activated in response to internal or external stimuli. External signals can reach the cell in the form of molecules but can also be caused by other environmental factors. Signaling molecules, also called ligands can bind to receptors extending from the cell membrane. The receptor changes its spatial structure, thereby setting off a cascade of signal transduction inside the cell. Signaling cascades

typically involve a series of protein modifications such as phosphorylation, dimerization, complex formation and cleavage. Since proteins can act as transcription factors and bind to promoters, if the signal reaches the nucleus, the cell can change its gene expression profile in reaction to the received signal.

Signaling ligands include growth factors such as EGF, TGF, and VEGF, which promote cell cycle progression, cell growth and cell differentiation. Members of the interleukin-1 family regulate inflammatory responses and are important in the immune response of cells. Other important signaling molecules include TNFα, TRAIL and Fas, which induce caspase activation and apoptosis.

The most important process by which signals are propagated in signaling pathways is through phosphorylation. Phosphorylation is a post-translational modification, which happens when a phosphate group is attached to a specific amino acid site (usually serine, tyrosine or threonine) of a protein. Phosphorylation often results in the activation of a protein through a change in its spatial conformation. For instance the tumor suppressor p53 is in an inactive form but is phosphorylated by ATM in response to DNA damage. It is only in its active, phosphorylated state that p53 can fulfill its role as a transcription factor. A typical signal transduction pathway representing externally triggered apoptosis is shown in Figure 2.1.

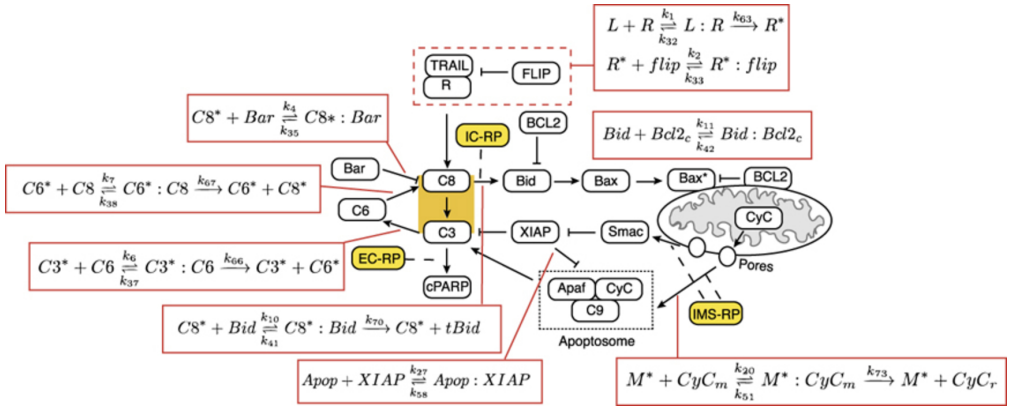


Figure 2.1: Signal transduction pathway governing externally triggered apoptosis, including reaction schemes. Figure is from [16] under the CC BY-NC-SA license.

Gene regulatory pathways

Gene regulatory pathways represent interactions between genes. Genes do not directly interact with each other, however they can influence each other through transcriptional regulation. An example of such a process is a gene which expresses a protein that in

turn binds to another gene's promoter region and changes the speed of transcription. Gene regulatory pathways comprise a network of genetic interactions as direct positive or negative regulation between genes.

The gene regulatory pathway for the circadian oscillator is shown in Figure 2.2. Each node corresponds to a gene and the positive (+), negative (-) and neutral (0) effects are shown along edges.

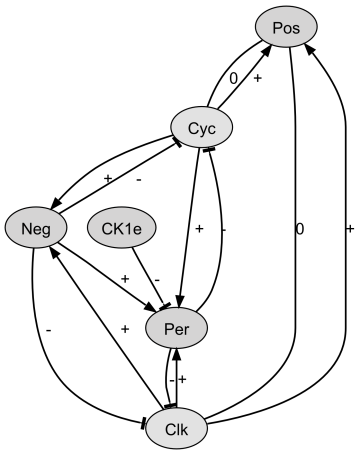


Figure 2.2: Gene regulatory pathway for the circadian oscillator. Figure is from the Science Database of Cell Signaling [17]

Metabolic pathways

Metabolic pathways are networks of reactions that transform metabolites and various other molecules. Cells require energy to function and energy in cells is used to build necessary compounds, maintain structure, and grow. Catabolic processes break down organic matter and store the released energy in form of adenosine triphosphate (ATP) molecules. Anabolic processes use these energy carrying molecules to construct further metabolites or cellular components such as nucleic acids and proteins. Enzymes play a crucial role in metabolic reactions. Enzymes allow certain reactions to happen at a fast rate - and thereby link species in the network - but they are not modified or consumed in the process. Metabolic pathway models therefore often concentrate on links between enzymes and the genes encoding them.

2.2 Modeling formalisms

Biological models have traditionally been represented through informal graphical diagrams. These diagrams can give a qualitative, structural overview of the system. However, diagrams do not specify the concentration of species or the dynamics of different reactions. As the size of a model grows, it is increasingly difficult to understand the complex network of non-linear effects based on informal diagrams alone.

Building quantitative models of pathways are useful in several ways. First of all, quantitative models let us untangle the strength of effects in a network of interactions [18]. They allow a clear and consistent analysis to which extent each component contributes to certain processes. Quantitative models are easily represented and simulated on a computer. In fact, it is the power to execute pathway models and make predictions that truly revolutionizes the way systems are understood [19]. Models also allow us to analyze biological pathways through systems theory and elucidate fundamental properties of biological systems such as modularity or robustness [20, 21].

Several formalisms have been introduced in the pathway modeling context. These can be classified according to many different characteristics, including whether they are mechanistic or abstract, deterministic or stochastic, static or dynamic and qualitative or quantitative. Some of the widely used formalisms are classified in Table 2.1. It is important to note that various extensions to the basic form of these models have been proposed in the literature (for instance *qualitative* differential equations) that make these distinctions less crisp.

	Mech	Abs	Qual	Quant	Det	Stoch	Stat	Dyn
ODE	x			x	x			x
CTMC/SDE	x			x		x		x
Boolean/Logic		x	x		x			x
BN		x		x		x	x	
DBN		x		x		x		x

Table 2.1: Classification of widely used pathway modeling formalisms depending on whether they are mechanistic or abstract, qualitative or quantitative, deterministic or stochastic and static or dynamic.

2.2.1 Mechanistic models

Building mechanistic models of biological pathways relies on chemical reaction kinetics. We first look at the kinetic laws that mechanistic models are built of and then discuss ordinary differential equation based deterministic models and Markov chain based stochastic models.

Reactions kinetics

The general form of a chemical reaction is

$$\sum_{i=1}^{N_r} a_i R_i \rightleftharpoons \sum_{j=1}^{N_p} b_j P_j, \tag{2.1}$$

where R_i are reactants, P_j are products, and a_i, b_j are the stoichiometric coefficients associated with them. In general, chemical reactions are reversible, however one direction may be negligibly slow compared to the other, in which case the reaction is considered irreversible.

The most basic concept in the quantitative modeling of chemical reactions is the law of mass action [22]. According to mass action kinetics, the speed of a reaction is proportional to the concentration of the reactants raised to the power of their stoichiometric coefficients. In what follows, we denote concentration with square brackets, for instance, the concentration of R_i is denoted $[R_i]$. The forward reaction speed of (2.1) is then expressed, according to the law of mass action, as

$$f = k_r \prod_{i=1}^{N_r} [R_i]^{a_i} - k_p \prod_{j=1}^{N_p} [P_j]^{b_j}. \tag{2.2}$$

Here k_r and k_p are kinetic rate constants.

A specific example, often encountered as a component of pathway models is an enzyme-substrate reaction. In this process, a substrate (S) is converted into a product (P) by binding to an enzyme (E) and forming an enzyme substrate complex (ES). The associated reactions can be written as

$$S + E \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES \overset{k_3}{\rightarrow} E + P. \tag{2.3}$$

The assumption in these reactions is that complex formation is a reversible process but product creation and release is irreversible. According to the law of mass action the

reaction rates for this system are

$$\begin{aligned} f_1 =& k_1[S][E] - k_2[ES] \\ f_2 =& k_3[ES], \end{aligned} \tag{2.4}$$

where k_1, k_2 and k_3 are reaction rate parameters.

Mass action kinetics provides a faithful model of the reaction dynamics in case it models elementary, physical interactions (such as binding and release in (2.3)). But it is often only the dynamics of the substrate and the product that is of interest, and this transformation cannot directly be modeled by mass action kinetics. This has resulted in the derivation of kinetic laws that summarize the dynamics of a series of elementary interactions. We now look at two of the most widely used such kinetic laws, the Michaelis-Menten equation and the Hill equation.

Michaelis-Menten kinetics relies on the assumption that the concentration of the substrate is much larger than that of the enzyme, and therefore the enzyme-substrate complex reaches a steady state and is not explicitly modeled. The speed of reaction from substrate to product can be captured by a single reaction rate:

$$f = V_{\max} \frac{[S]}{K + [S]}. \tag{2.5}$$

The parameters $V_{\max}$ and K can be derived from the mass action parameters, and have easily interpretable physical meanings. In addition, they can be measured more easily, therefore Michaelis-Menten kinetics are popular when building quantitative pathway models [23, 24].

The Hill equation can be used to model processes in which a substrate (S) can bind to several different sites of a macromolecule, and bound substrates can influence the rate of new substrates being bound, also called cooperativity [25]. The reaction rate can be written as

$$f = V \frac{[S]^n}{K^n + [S]^n}. \tag{2.6}$$

Here V and K are kinetic rate constants and the parameter n quantifies substrate cooperativity, and can represent positive or negative cooperativity depending on its value.

Several other rate laws have been derived [26] and are used for modeling purposes.

Ordinary differential equation models

Given a model structure and reaction kinetics, it is straightforward to obtain an ordinary differential equations (ODE) model of the dynamics. We construct an equation for each modeled species x_i, $1 \leq i \leq n$, which describes its immediate concentration change at any given time. The reaction rate producing the species will appear with positive sign and the reactions consuming it with negative sign ($r_{i,k} > 0$ and $r_{i,k} < 0$ respectively, for reaction $1 \leq k \leq K$). The differential equation governing x_i is written as

$$\frac{d[x_i]}{dt} = \sum_{k=1}^{K} r_{i,k} f_k, \tag{2.7}$$

where f_k is the kinetic rate of reaction k. The state of the system at time t is described by the vector $\mathbf{x}(t) := (x_1(t), \ldots, x_n(t))$. The kinetic rate constants will be summarized in a vector θ, which we will refer to as model parameters. We also define the vector valued function F, which describes the right hand side of the equations in (2.7) as

$$\dot{\mathbf{x}}(t) = F(\mathbf{x}, \theta). \tag{2.8}$$

Given a value assignment to θ and initial conditions $\mathbf{x}(0)$, the *solution* of the ODE system is the state trajectory $\mathbf{x}(t)$ for some time range $t \in [0, T]$. Additionally, given that the right hand side of the equations are C^1 functions, there is a unique solution to the equations [27]. Analytical solutions only exist for a restricted class of ODE systems, for example ones whose right hand side is linear. In the case of large and non-linear systems typically encountered in the pathway modeling context, closed form solutions will not be available. Therefore, numerical integration methods are used to obtain approximate solutions to the dynamics. Fix step-size solvers such as the fourth order Runge-Kutta method (RK4) are fast and easy to implement [28]. However, due to a fix step-size parameter, they are unsuitable for solving *stiff* problems, which are often encountered in kinetic models due to different time-scales in the system [29]. Simulators for pathway models therefore rely on more sophisticated solver packages which are efficient in a stiff setting, such as LSODA [30] and CVODE [31].

As an example, we show the ODE description of the enzyme kinetic model described by (2.3) in Figure 2.3. The individual equations are obtained by using (2.4) and (2.7). Simulation is performed for $t \in [0, 10]$ with initial conditions $\mathbf{x}(0) = (15, 10, 0, 0)$ and parameters $\theta = (0.1, 0.1, 0.35)$ using the CVODE solver.

$$
\begin{aligned}
\frac{d[S]}{dt} &= k_2[ES] - k_1[E][S] \\
\frac{d[E]}{dt} &= (k_2 + k_3)[ES] - k_1[E][S] \\
\frac{d[ES]}{dt} &= -(k_2 + k_3)[ES] + k_1[E][S] \\
\frac{d[P]}{dt} &= k_3[ES].
\end{aligned}
$$

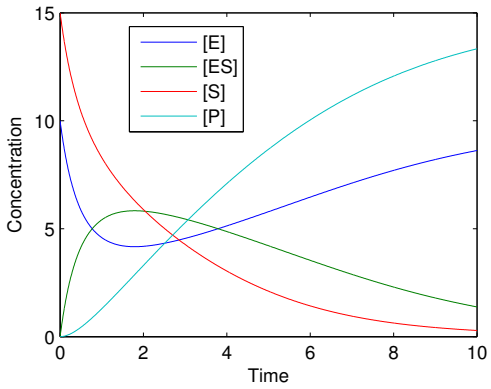


Figure 2.3: ODE equations and time course solutions for a simple enzyme-substrate system.

ODE models are an adequate description of pathway dynamics when the molecule numbers are high enough to be treated as continuous values [32]. ODEs provide a deterministic description of the dynamics. However, by allowing variability in initial conditions and model parameters, it is possible to endow the model with a probability distribution, as we do in Chapters 3 and 4 of this work. In case the quantity of constituents is low, a continuous description of the dynamics may not be suitable. Stochastic models which address this issue are introduced in the following section.

A further assumption in ODE models is that the contents of the cell are well-mixed and the location of the components is not relevant. If representing spatial position is necessary (for instance in pattern formation during development), partial differential equation (PDE) models can be used. Compared to ODE models, PDEs are significantly harder to calibrate and simulate [33].

Stochastic mechanistic models

Chemical reactions inside the cell often happen at low molecule numbers in a stochastic manner. In this case it is reasonable to represent the quantity of species in terms of molecule numbers instead of concentrations [34]. Stochastic models provide a way to describe the discrete change in molecule numbers over time. Reaction events are assumed to be distinct, and each reaction event changes the molecule numbers according to the stoichiometric coefficients. The random occurrence of reaction events in time results in a discrete state space stochastic process governing the species. A rigorous derivation of this stochastic process (also referred to as the chemical master equation) based on statistical physical considerations is described in [35]. The chemical master equation implicitly

defines a continuous time Markov chain (CTMC) which can be exactly simulated using the Gillespie algorithm [36]. The CTMC model also includes transition rate parameters, and in fact, these parameters are even more challenging to learn than ones in ODE models.

There are several ways of relating CTMCs and ODEs. First of all, the expectation of the stochastic Markov process can be modeled using deterministic ODEs [37]. This formally results in the same equations as the deterministic representation of the system, however the species are measured in molecule numbers and the rate constants have different meaning and numeric value. A more appropriate approximation to a CTMC, which retains the stochasticity of the system, is one based on stochastic differential equations (SDE) [38]. SDEs model the change in molecular quantities as diffusion processes. SDEs can speed up the simulation process and are amenable to useful analysis techniques known from other fields, most notably finance [39].

When a pathway contains species, some of which exist at low and others at high molecule numbers, using a purely deterministic or purely stochastic model is impractical. Hybrid simulation methods have been developed to deal with this problem. In this context, species and reactions are partitioned, and a single simulation algorithm is given which contains discrete and continuous state updates [40, 41].

2.2.2 Abstract models

It is often the case that a the species of interest which need to be included in a pathway model do not directly interact with each other. Further, one may be interested in modeling the activity level of each species rather than its molecular amount [42]. In such cases standard kinetic laws are not applicable, and a more abstract description of the influences among species is needed. Conceptually simple methods such as multilinear regression [43] and principal component analysis [44] can reveal influences in a data-driven manner. Models based on logic rules such as Boolean models [45] and fuzzy logic models [46, 47] have also been proposed in this context.

Here we introduce Bayesian and dynamic Bayesian networks, which model influences in a probabilistic framework.

Bayesian networks

Bayesian networks are probabilistic graphical models represented as a directed acyclic graph (DAG) [48, 49]. The graph consists of a finite set of nodes X and a set of edges $E \subset X \times X$. The set of nodes $X = \{X_1, X_2, \ldots, X_n\}$ correspond to random variables and the edges encode the independence structure of the joint distribution of X. Here $X_i \in X$ represents a finite valued random variable taking its value from the set V.

The Markov property induced by the edges states that a node is independent of its non-descendants given its parents. A Bayesian network structure is faithful to the underlying joint distribution when an independence relationship is implied by the Markov property if and only if the same independence relationship exists on the corresponding set of random variables.

The main advantage of the graph representation is that it allows a succinct parametrization of the joint distribution. Namely, it is enough to parametrize the distribution of each node conditioned on its parents. We associate a conditional probability table $\Theta_i = P(X_i|PA(X_i))$ with each node. Here $PA(X_i)$ is the set of parents of X_i defined as $PA(X_i) = \{X_{i_1}, \ldots, X_{i_\ell}\}$ with $(X_{i_k}, X_i) \in E$ for $1 \leq k \leq \ell$. Each entry $\Theta_i(x_i|x_{j_1}, x_{j_2}, \ldots, x_{j_\ell})$ encodes the probability of X_i taking a value $x_i \in V$ given the value assignment $(x_{i_1}, x_{i_2}, \ldots, x_{i_\ell}) \in V^\ell$ to its parents. Using this parametrization, and exploiting the Markov property, we can express the factorized joint distribution as

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|PA(X_i)). \tag{2.9}$$

The conditional probability table entries can be used directly to calculate the probability of a joint assignment.

Dynamic Bayesian networks

Dynamic Bayesian networks (DBNs) represent a set of random variables over time [50, 49]. In the DBN, a set of system variables $X = X_1, X_2, \ldots, X_n$ are modeled at a discrete set of time steps $t \in \{0, 1, \ldots, T\}$. The model consist of a node for each variable at each time point, for instance, X_i^t denotes the random variable representing the value of X_i at time t. Similar to general Bayesian networks, edges encode an independence structure among the set of nodes. However, edges are restricted such that they are (i) directed forward in time and (ii) only span a single time step. With these assumptions, for

$t > 0$, we have the parenthood relationship $PA(X_i^t) \subseteq \{X_1^{t-1}, X_2^{t-1}, \ldots, X_n^{t-1}\}$, and for the initial time point $PA(X_i^0) = \emptyset$. From here the set of edges E is defined as $(X_j^{t-1}, X_i^t) \in E$ if and only if $X_j^{t-1} \in PA(X_i^t)$.

This implies a set of first-order Markov assumptions in time, namely, that given the current state, the next state is independent of the previous state, or

$$(X_1^{t-1}, X_2^{t-1}, \ldots, X_n^{t-1}) \perp\!\!\!\perp (X_1^{t+1}, X_2^{t+1}, \ldots, X_n^{t+1}) | (X_1^t, X_2^t, \ldots, X_n^t). \tag{2.10}$$

The parametrization of a discrete DBN model is through a set of conditional probability tables, one for each variable at each time point. The CPT for variable X_i at time t, denoted Θ_i^t, will contain entries of the form $\Theta_i^t(x_i^t | x_{i_1}^{t-1}, x_{i_2}^{t-1}, \ldots, x_{i_\ell}^{t-1})$, representing the probability of X_i^t taking the value $x_i^t \in V$, given the value assignment $(x_{i_1}^{t-1}, x_{i_2}^{t-1}, \ldots, x_{i_\ell}^{t-1}) \in V^\ell$ to $PA(X_i^t)$.

A BN and DBN model of a small signaling network is shown in 2.4. Each node represents the activity of a molecular species, in this case, a protein. The example illustrates the difference in the way static BNs and DBNs are used in biology. BNs do not have a time component and can only represent static (steady state or equilibrium) influences among molecular species. In contrast, DBNs model the temporal influence among species and can be learned based on time-course experimental data. Another important difference is that BNs require acyclicity and therefore cannot model feedback loops. Due to the fact that DBN variables are present across multiple time steps, the forward directed edges can model feedback loops. For example, the edges RAF $\rightarrow$ ERK and ERK $\rightarrow$ RAF constitute a feedback loop in the DBN in Figure 2.4. Note also, that we have included edges in the DBN from each species to itself in the next time point. This is intended to model forms of persistence, for instance the fact that a protein is more likely to stay active once it has been activated.

2.2.3 Summary

We introduced both mechanistic and abstract pathway modeling formalisms. In the rest of this thesis we will focus on ODE models to represent dynamics based on molecular level interactions in a continuous time, deterministic manner. Our choice of ODEs relies on the assumption that they provide an accurate description of dynamics when molecular quantities are sufficiently high. Conversely, we will use DBNs to represent

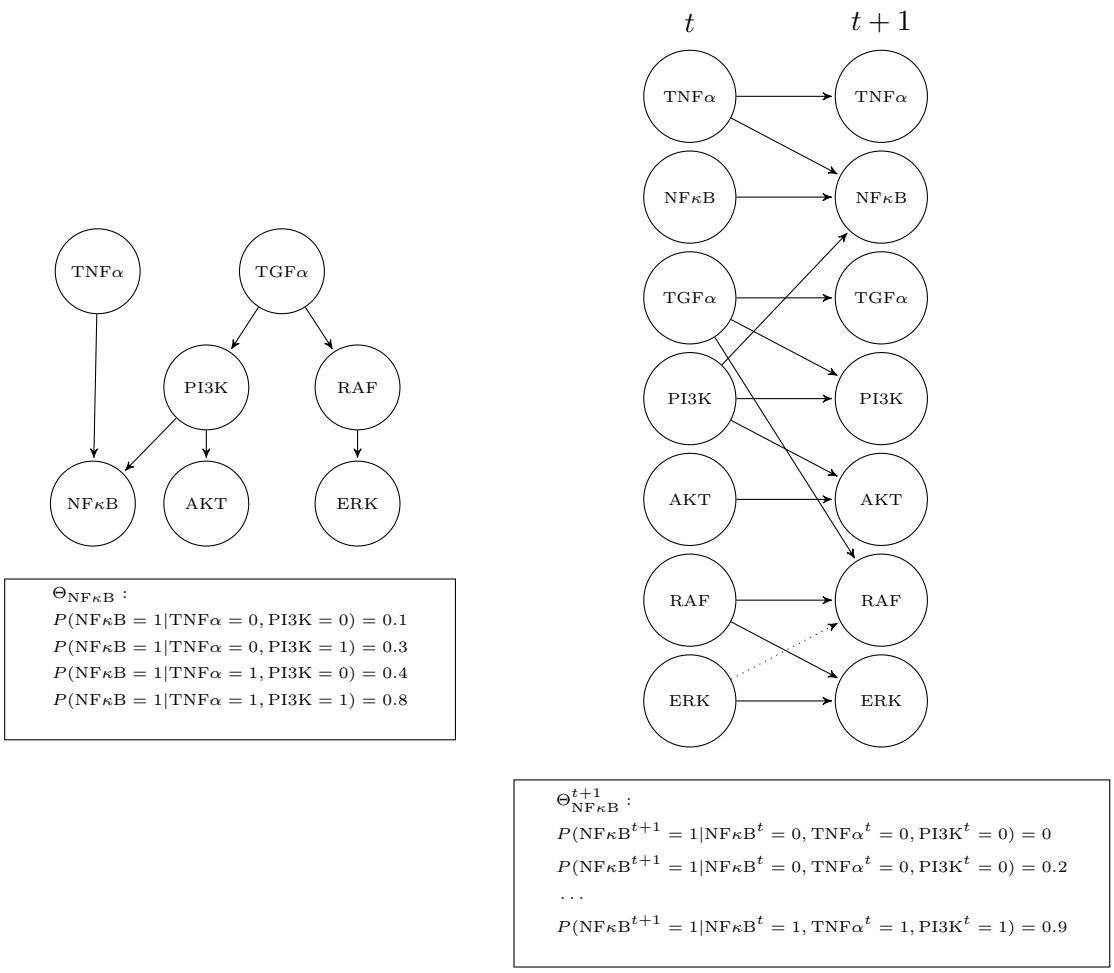


Figure 2.4: Bayesian network and dynamic Bayesian network representation of a small signaling pathway model. The model is adapted and simplified from [51]. The dotted edge from ERK ro RAF in the DBN forms a feedback loop. The same feedback cannot be modeled on the static BN.

abstract, probabilistic interactions between molecular species when the modeling goals require a larger scale but less detailed description.

2.3 Model calibration

Focusing on ODE models, we now discuss how unknown model parameters can be estimated or inferred given experimental data. Dynamical pathway models typically contain a number of unknown kinetic rate parameters. The initial concentration of some species, if they are unknown, can also be considered parameters. Getting quantitatively consistent values for these parameters is a significant challenge in current systems biology efforts and is an active area of research [52].

Some parameters can be measured experimentally. For instance, the parameters of a

reaction with Michaelis-Menten kinetic rate may be measured in vitro. This approach, however, is impractical since experiments are very time consuming and expensive. Resources would be better allocated making measurements on the system instead of its elements in isolation. In addition, reaction rates measured in isolation may not be consistent with those present in the studied system. For the above reasons, the estimation of model parameters is carried out using computational methods.

We introduce two conceptually different ways of formulating the model calibration problem. Parameter *estimation* poses an optimization problem for finding the single best parameter vector. The underlying assumption is that parameters are constants which have an unknown but exact value. Parameter *inference* relies on representing parameters are random variables. The parameters possess a prior probability distribution, which is then updated by experimental data using probabilistic inference. The resulting probability distribution is commonly referred to as the posterior distribution. Note that the latter formalism still maintains that there is an underlying exact parameter value. It is rather our limited knowledge or *belief* about the parameter value which is modeled as a probability distribution.

2.3.1 Parameter estimation

Assume that we are given a set of experimental data Y, which contains measurements for some of the variables at a few discrete time points. Our goal will be to find model parameters $\hat{\theta}$ such that the simulated output of the model provides a good fit to the data. The experimental data is structured as follows. $Y_{i,j}$ denotes the measured value for species i at time point t_j, where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. In practice, data is often available for a set of different experimental conditions and Y can be expanded in the obvious way to show this.

Parameter estimation is formulated as an optimization problem with respect to an objective function J. The objective function takes a vector of proposed model parameters as an argument and quantifies the difference between data and simulated model output. The most commonly used objective function is the weighted sum of squared differences:

$$J(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{i,j}(x_i(t_j)|_\theta - Y_{i,j})^2. \tag{2.13}$$

Here $x_i(t_j)|_\theta$ is the result of simulation when using parameter θ and $w_{i,j}$ is a weight

corresponding to each data point. Weights are used in practice to account for the differences in magnitude of species concentrations.

Given the objective function, parameter estimation is an optimization problem with the goal of finding the least-squares parameter estimate

$$\hat{\theta}_{\mathrm{LS}} = \underset{\theta}{\operatorname{argmin}}\, J(\theta). \tag{2.14}$$

The error function itself is quadratic, but since simulated curves depend on parameters in a highly non-linear way, finding the minimum is a challenging non-linear optimization problem. Parameter estimation methods use a search algorithm to find the optimum in the (usually high-dimensional) space of parameters.

Several methods have been proposed to solve the optimization problem in the context of pathway models [53]. Local methods such as Hooke-Jeeves pattern search [54] or Levenberg-Marquardt method [55] are useful when the optimum is in near the initial point that the search starts from. Often the range of parameters is wide and the parameter space contains numerous local minima. In this setting, global search methods are needed, which implement ways of avoiding local minima. Stochastic ranking evolutionary strategies [56] and genetic algorithms [57] are some of the popular methods that have proved to work well in practice [58].

Global optimization methods often work well in practice but are based on heuristics and are not proven to converge to the global optimum in a finite number of steps. It is not possible to theoretically characterize the set of samples at any specific iteration of the search. Most importantly, these methods only provide a single output to the optimization problem. It is not known whether that is an optimal value and whether there are any other "good" values.

2.3.2 Parameter inference

Parameter inference [59, 60] defines parameters as random variables in a Bayesian probabilistic framework. This is both conceptually and also in methodology, fundamentally different from parameter estimation. Even if the underlying parameter (such as a kinetic rate constant) does have a well defined and exact value, the probabilistic approach allows us to model our belief or uncertainty about its value based on limited data. The parameter vector θ is endowed with a *prior* distribution $p_0(\theta)$, in the simplest case, uni-

form over a bounded interval for each parameter. The experimental data Y is related to the parameters through the *likelihood* function $p(Y|\theta)$, which expresses the probability of observing Y given parameters θ. The form of the likelihood function is assumed to be known, and can be evaluated using simulation.

Our goal is to constrain the distribution of the parameters by conditioning on experimental data. This conditioning is expressed in the *posterior* distribution, which we denote $\pi(\theta|Y)$. Using the Bayes theorem, we can express the posterior as

$$\pi(\theta|Y) = \frac{p(Y|\theta)p_0(\theta)}{p(Y)} = \frac{p(Y|\theta)p_0(\theta)}{\int p(Y|\theta)p_0(\theta)d\theta}. \tag{2.15}$$

Note that in (2.15) the denominator is not a function of θ, and hence $\pi(\theta|Y) \propto p(Y|\theta)p_0(\theta)$. The posterior probability of the parameter is determined by how likely the parameter is inherently (its prior), and its compatibility with the observed data (its likelihood).

The Bayesian framework allows several ways to determine model parameters:

$\underset{\theta}{\operatorname{argmax}}\, p(Y\|\theta)$	Maximum likelihood (ML),
$\underset{\theta}{\operatorname{argmax}}\, \pi(\theta\|Y)$	Maximum aposteriori probability (MAP),
$\pi(\theta\|Y)$	Bayesian posterior.

While ML and MAP estimates recover a single parameter value, the goal of parameter inference is to construct the full Bayesian posterior. Projections of the (usually high-dimensional) parameter posterior can reveal the histograms of individual parameters and the correlation between pairs of parameters. However, recovering the posterior distribution is not the only goal of parameter inference. Namely, it can be useful to evaluate the expected value of a function of θ with respect to the posterior. For instance, the expected value of the function $f(\theta)$ with respect to the posterior is

$$\mathbb{E}_\pi f = \int \pi(\theta|Y)f(\theta)d\theta. \tag{2.16}$$

Several methods have been proposed for calculating integrals of this form including Markov chain Monte Carlo [61], sequential Bayesian filtering (including Kalman filters and particle filters) [62], variational Bayesian methods [63] and approximate Bayesian computation [64]. More details on these methods will be discussed in Chapters 3 and 4.

There is an important connection between parameter estimation and parameter inference. More specifically, there is a correspondence between the sum of squared errors and a special case of the likelihood function. Constructing the likelihood function usu-

ally involves integrating out the possible realizations of the system given a parameter value. In the context of ODE models we can exploit the properties that (i) the system state trajectory is a unique and deterministic function of θ; (ii) Y is available at a finite discrete number of time points; and (iii) the measurements are uncorrelated given the current system state. With these properties, the general form of the likelihood simplifies to $p(Y|\theta) = \prod_{i=1}^{n}\prod_{j=1}^{m} p(Y_{i,j} \mid x_i(t_j)|_\theta)$. Here again $x_i(t_j)|_\theta$ is the result of simulation when using parameters θ. It is often reasonable to assume that $p(\cdot \mid x_i(t_j)|_\theta)$, that is, the distribution of a data sample conditioned on the system state, is Gaussian. It can then be written as $p(\cdot \mid x_i(t_j)|_\theta) = \mathcal{N}(x_i(t_j)|_\theta, \sigma^2_{i,j})$, where $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2. The log-likelihood is then expressed as

$$\log p(Y|\theta) = \sum_{i=1}^{n}\sum_{j=1}^{m} \log p(Y_{i,j} \mid x_i(t_j)|_\theta) = -C\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{(Y_{i,j} - x_i(t_j)|_\theta)^2}{\sigma^2_{i,j}}, \tag{2.17}$$

with C being a positive constant. We can now establish a connection between the maximum-likelihood and the least-squares estimate using (2.13) and (2.14) as

$$\hat{\theta}_{\mathrm{LS}} := \operatorname*{argmin}_\theta J(\theta) = \operatorname*{argmax}_\theta \log p(Y|\theta) =: \hat{\theta}_{\mathrm{ML}}, \tag{2.18}$$

with the choice of $w_{ij} := 1/\sigma^2_{i,j}$.

2.4 Model analysis and verification

It is possible to generate hypotheses and gain insights about biological systems using analysis techniques on the model representing it. In the context of ODE models, steady state analysis concentrates on the behavior of the dynamics in the limit, including stable states and limit cycles [65]. Bifurcation analysis reveals how the steady state landscape changes as a function of changing parameters or initial conditions, and concentrates on abrupt qualitative changes in limit behavior (for instance transition from limit cycle to stable state) during a smooth change in parameters [66]. Sensitivity analysis provides a quantitative measure of how the time-course dynamics or the steady state of the system changes when varying parameters or initial conditions. Both local and global sensitivity analysis methods are widely used and provide insights about variability and robustness [67].

Having constructed and calibrated a quantitative model, it is important to *verify*

that it is consistent with knowledge about the underlying system. Manually verifying properties of interest based on simulation output is difficult and prone to interpretation bias, especially if the pathway being modeled is large. Models are often constructed in an iterative manner, and verification may need to be performed many times during this process. This motivates us to choose a language to make statements about dynamics, and use algorithms to automatically verify whether these properties are met by the model. Temporal logics coupled with model checking algorithms have been applied for this purpose in fields including program analysis [68] and circuit design verification [69], and are recently also adopted for pathway models.

Model checking, in general terms, is used to verify state transition models with respect to properties expressed in formal logic [70]. Model checking has found many applications in systems biology. For instance, it has been used to verify properties of a stochastic model of the mitogen activated protein kinase (MAPK) cascade [71], a model of fibroblast growth factor signaling [72] and to analyze the network controlling the nutritional stress response in E. coli using piecewise linear ODEs [73].

Properties about realizations of dynamical models can be expressed using temporal logic. There are many choices for an appropriate logic depending on the modeling formalism and the goals of verification. Here we focus on linear temporal logic (LTL), which has proven to be particularly useful in systems biology as it can be interpreted on a broad class of model types and can intuitively express properties of interest. An LTL formula is interpreted on a single execution path and is therefore well suited for deterministic ODE models. Using temporal operators in LTL it is possible to make statements about reachability (**F**), stability (**G**) and ordering (**U**) of events, and these statements can be combined using standard logic operators such as AND ($\wedge$), OR ($\vee$), and implies ($\Rightarrow$).

For instance, recall the small enzyme-substrate system from Figure 2.3. The property $\mathbf{FG}[0 \leq [\mathrm{S}] \leq 5]$ is interpreted as: *at some time* the substrate concentration will be in the interval $[0, 5]$, and *from then on* it will *stay* in the same interval. The formula $[5 \leq [\mathrm{E}] \leq 10]\mathbf{U}[5 \leq [\mathrm{ES}] \leq 10]$ states that the enzyme concentration is between $[5, 10]$ *until* the concentration of the enzyme-substrate complex reaches the interval $[5, 10]$. From Figure 2.3, we easily see that both these properties are *true*. We will give a more precise definition of the syntax and semantics of LTL formulas in Chapter 4.

In the context of ODEs, since one must rely on numerical solutions, a bounded LTL (BLTL), interpreted on finite time intervals is usually necessary. Further, when the realizations of the system are stochastic, PBLTL, a probabilistic extension of BLTL can be used. PBLTL is applicable on CTMCs and other stochastic models, but also in a setting where an ODE system is endowed with a probability distribution due to model uncertainty. A formula in PBLTL has the general form $\mathbb{P}_{\geq r}\varphi$, where φ is a BLTL formula, and the intended meaning is, φ holds with *at least probability* r. Solving this probabilistic model checking problem exactly is intractable for large models due to state space explosion. However, statistical model checking provides an approximate solution, and its efficiency does not depend on the size of the state space.

Statistical model checking (SMC) involves repeatedly simulating the system, verifying the property for each realization and deciding whether $P(\mathcal{S} \models \varphi) > r$ that is, whether the dynamical model $\mathcal{S}$ satisfies φ with at least probability r. The decision can be made once sufficiently many samples have been evaluated [74]. The usual formulation of SMC is based on a hypothesis test between H_0: $P(\mathcal{S} \models \varphi) > r + \delta$ and H_1: $P(\mathcal{S} \models \varphi) < r - \delta$ [75]. The parameter δ defines an indifference region around r, in which choosing either H_0 or H_1 is acceptable. The standard SMC scheme relies on the sequential-probability ratio test (SPRT) as a stopping criterion [76]. After drawing m samples, we compute a stopping criterion q_m as

$$q_m = \frac{[r-\delta]^{(\sum_{i=1}^m z_i)}[1-[r-\delta]]^{(m-\sum_{i=1}^m z_i)}}{[r+\delta]^{(\sum_{i=1}^m z_i)}[1-[r+\delta]]^{(m-\sum_{i=1}^m z_i)}} \quad . \tag{2.19}$$

Here z_i is 1 if the ith simulated trajectory satisfies the formula and 0 otherwise. Hypothesis H_1 is accepted if $q_m \geq \frac{1-\beta}{\alpha}$, and hypothesis H_0 is accepted if $q_m \leq \frac{\beta}{1-\alpha}$. If neither is the case then another sample is drawn. The constants α and β are chosen by the user and signify the upper limit on false positive and false negative decisions. Statistical model checking has been applied, for instance, to verify a large stochastic model of T-cell receptor signaling [77].

There are many other works that aim to make model checking methods more applicable for pathway modeling tasks. The method proposed in [78] makes the construction of temporal logic formulas easier for practitioners by allowing biologically relevant, high-level query templates to be pieced together and automatically translated into temporal logic. Model checking has also been used to search for parameters with which the model

fits specified dynamical properties [79, 80]. In [81] we proposed statistical model checking coupled with global optimization to find parameters of ODEs with variability in initial conditions, and used the approach to calibrate large pathway models.

There are a number of tools that are available for model checking biological pathway models including BioCham [82], BioDiVinE [83] and MIRACH [84]. There are also several general purpose tools that have been used for pathway models such as PRISM [85], UPPAAL [86] and Breach [87]. For further details, we refer to a comprehensive review of model checking applied in systems biology in [88].

Chapter 3

Bayesian parameter inference using kernel-enhanced particle filters

3.1 Introduction

Quantitative models are essential for better understanding the dynamics of biological pathways, and ordinary differential equations (ODEs) are the most often used modeling formalism in systems biology. However, calibrating model parameters to be consistent with prior knowledge and experimental data remains a significant challenge. The limited nature of experimental data, coupled with the common unidentifiability of parameters has motivated the representation of the model parameters as a probability distribution, rather than a single value. This allows the representation of a finite or infinite set of model parameters, each possible parameter vector weighted according to its support from prior knowledge and experimental evidence. The advantages of a probabilistic approach include the coherent treatment of measurement noise, prior knowledge and also, possibly, the stochasticity of dynamics, in a single quantitative framework. Further, the uncertainty in the predictions made by the model can be quantified. The main difficulty with this approach is that large-scale Bayesian inference is required for reproducing the posterior distribution of model parameters. Designing efficient parameter inference algorithms for ODE based pathway models is a difficult and open problem [59].

A class of algorithms which can approximate the distribution of parameters is sequen-

tial Bayesian inference, also called Bayesian filtering and data assimilation [62]. These methods were originally developed for inferring the hidden state of a system based on noisy observations. However, by taking parameters as static state variables, they can be extended to recover a distribution over parameters. Sequential inference methods start with a prior distribution and iteratively update it by stepping forward in time and incorporating the measurement data available at the current time point into the distribution. The sequential nature of these methods increases efficiency by breaking up inference into a series of simpler problems, and also enables the use of the methods in settings where observations are received in real time [89].

Particle filters are especially well suited for sequential inference, since they approximate the sequence of distributions by samples (called particles), which has proven numerical advantages in high-dimensional settings [90]. Such high-dimensional settings are very common when dealing with pathway models, as each model parameter corresponds to an additional dimension in the parameter space. The particle filter starts by generating a set of samples according to the prior distribution. These samples are propagated forward in time using the model dynamics. When a new observation is available, the samples are reweighted by the likelihood of the observation. The particles are then resampled proportional to their weights to concentrate them in regions of high probability. This method has been shown to result in samples distributed according to the true posterior distribution [90].

A useful graphical tool called DA1.0 [13] has recently been released, which implements the particle filter algorithm for pathway parameter estimation, following the methods used in [91]. However, the estimation often fails in practice due to particle collapse. Namely, since parameters are static, once they have been sampled initially, their value cannot change. Resampling then results in the multiplication of only a few high-weight particles, leading to a loss of diversity among samples, and, ultimately, to a collapsed representation of the posterior. This phenomenon often appears when using particle filters, and is also called particle degeneracy and sample impoverishment [92].

There have been two approaches proposed in the systems biology literature to deal with this problem. A pragmatic, brute-force approach advocates using very high sample sizes and peta-scale parallel computing to get satisfactory results [93]. However, this approach does not address the root cause of particle collapse, the fact that parameters

are static. Another possible approach to avoid particle collapse is to add random noise to the particles at each step of the filter. This can help in introducing diversity since the resulting particles will be spread to different positions randomly. In the context of ODE pathway models, injecting noise was recently proposed in [94], adopting a method originally proposed in [95]. However, this method disrupts the estimate, since the randomly generated noisy particles are no longer distributed according to the posterior. This can lead to biased and inaccurate estimates.

To overcome these limitations we propose to use an improved particle filter, which relies on applying a Markov transition kernel on the particles at each step [96]. The method works by iterating over the (potentially collapsed) particles and *proposing* a randomly generated new position. Then, the posterior probability of the current and the proposed particle are compared, and the proposal is either *accepted* or rejected according to an acceptance probability. The kernel (the combination of proposal and acceptance) is designed in a way that the new samples are still distributed according to the true posterior distribution.

We use case studies to show that the kernel-enhanced particle filter approach gives more accurate estimates under parameter uncertainty than other methods previously used for ODE based pathway models. A limitation of previous results on particle filters for pathway parameter inference was that their performance was evaluated based on tight convergence to the nominal model parameters. Arguably, this is not an adequate basis of evaluation, since the goal of the inference task is finding the posterior *distribution* representing the uncertainty in parameter values, rather than finding a single best value. We address this by using the accuracy in making predictions according to the parameter posterior as a valid basis of comparison between particle filters. Intuitively, a better particle filter will provide estimates with smaller bias and lower variance with a given sample size.

In our case studies, we first construct a small synthetic example to illustrate the limitations of previously used particle filters in pathway parameter inference, and show that the kernel-enhanced particle filters are more robust due to their ability to recover from particle collapse. Next we use a model of the JAK-STAT signaling pathway (a commonly used benchmark model for Bayesian inference) and evaluate Bayesian predictions about quantities of interest. We show that using an equal number of particles,

predictions are made with much higher accuracy using kernel-enhanced filters than with other particle filters. The significantly increased efficiency holds even when factoring in the additional computational cost of performing kernel steps. Due to their accuracy and efficiency, using kernel-enhanced particle filters will make parameter inference more realistic for ODE based pathway models, and could lead to a wider adoption of Bayesian inference in this context.

In the next section we define the basis of sequential Bayesian estimation and particle filters and discuss how they have previously been used in the pathway modeling context. Section 3.3 introduces the kernel-enhanced particle filter and proposes schemes for implementing it on pathway models. We then evaluate the performance of our particle filters and compare it to previously proposed ones in Section 3.4.

3.2 Background and previous work

In this section we establish the basis of using particle filtering for inferring the distribution of pathway model parameters. We first give a state space formulation of the ODE model, and then introduce sequential inference methods for recovering the parameter posterior. We also introduce previous works using sequential Bayesian inference, and specifically particle filters, in the setting of pathway models.

3.2.1 Pathways as state space models

We introduced ODE models of biological pathways in Section 2.2.1. The basis for performing sequential Bayesian estimation on ODEs is formulating the equations as a discrete time state space model [97]. In a standard ODE model only the state evolution is described. Measurements are regarded as external data, to which some model outputs can be fit through distance measures, such as squared error distance. As opposed to this, the state space model includes both the state evolution and observations as part of a single probabilistic model. Since information about the state is only available through observations, the state transitions are modeled between the available discrete observation times, resulting in a discrete time description of the dynamics. Assume that observations are available at time points t_1, $t_2, \ldots, t_T$. We will use the discrete time index $n \in \{1, 2, \ldots, T\}$ to denote the observation at time t_n as $\mathbf{y}_n := y(t_n)$. The same

discrete indexing is used for the system state, that is, $\mathbf{x}_n := \mathbf{x}(t_n)$.

The state space model is formally stated as

$$\begin{aligned} \mathbf{x}_0 &\sim p_0(\mathbf{x}) \\ \mathbf{x}_n &\sim p(\mathbf{x}|\mathbf{x}_{n-1}, \theta) \\ \mathbf{y}_n &\sim g(\mathbf{y}|\mathbf{x}_n). \end{aligned} \tag{3.1}$$

Here, $\mathbf{x}_n$ and $\mathbf{y}_n$ are random variables representing states and observations, $\theta \in \mathbb{R}^d$ is a vector of d model parameters, p_0 is a prior distribution, $p(\mathbf{x}|\mathbf{x}_{n-1}, \theta)$ is the transition model and $g(\mathbf{y}|\mathbf{x}_n)$ is the observation model.

The state space model will be able to represent a general class of deterministic and stochastic dynamical models including ODEs, DTMCs, CTMCs and SDEs. ODE models constitute a special case, since their dynamics are deterministic, and the transition model $p(\mathbf{x}|\mathbf{x}_{n-1}, \theta)$ needs to be expressed using the ODE equations. We will make use of $\mathcal{F}_\theta$, the *flow* of the ODEs [27] to construct the state transition between successive time points under the parameter vector θ. $\mathcal{F}_\theta$ is a function $\mathbb{R} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$, where d_x is the number of state variables of the ODE, and is defined as

$$\mathcal{F}_\theta(t, \mathbf{x}) = \mathbf{x} + \int_0^t F(\mathbf{x}(\tau), \theta) d\tau, \tag{3.2}$$

where F is the right hand side of the ODE. With this choice, we have

$$\mathbf{x}_n := \mathcal{F}_\theta(t_n - t_{n-1}, \mathbf{x}_{n-1}). \tag{3.3}$$

As a consequence of the discrete-time nature of this model, there are several independence assumptions that hold. First, the state $\mathbf{x}_n$ is independent of all previous states given the state $\mathbf{x}_{n-1}$. The states, therefore, form a Markov chain. Further, the observation $\mathbf{y}_n$ is independent of all other observations and states given the current state $\mathbf{x}_n$. The independence assumptions in this description are intuitively captured by a probabilistic graphical model, in particular, a hidden Markov model. We note that HMMs have traditionally been used in a discrete state space setting, however, here we follow the sequential state estimation literature, where the name is used for both discrete and general state spaces [89]. Algorithms on the HMM allow us to reconstruct the distribution of the hidden sequence of states given observations. Here our goal will be

to use such algorithms to infer the value of the model parameters. This motivates us to formally treat model parameters as part of the system state. We augment the system state $\mathbf{x}$ with θ and introduce the notation $\mathbf{s} = (\mathbf{x}, \theta)$ to refer to the joint state-parameter vector.

Using this formalism, even though the model parameters are static, they are formally endowed with temporal dynamics. We introduce a time index on θ to represent its value at time n as θ_n. We can extend (3.1) to include the process describing the parameter dynamics as

$$\begin{aligned} \theta_0 &\sim p_{0,\theta}(\theta) \\ \theta_n &= \theta_{n-1}. \end{aligned} \tag{3.4}$$

The ODE flow $\mathcal{F}$ and the observation model g can be extended to the joint state $\mathbf{s}$ in the obvious way. This formulation is shown as a graphical model in Figure 3.1.

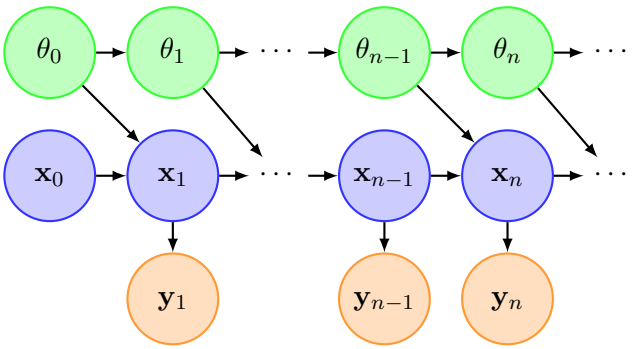


Figure 3.1: State space model with dynamic parameters.

This model structure is crucial since it allows us to infer the value of the parameters sequentially. In the following section, we show how to infer the hidden states and parameters of the system.

3.2.2 Sequential filtering

In this section we introduce filtering, a method to infer the hidden state of the model conditioned on the observations up to time n. We use the notation $\mathbf{y}_{1:n}$ to represent the set of observations from time 1 to time n. The goal of sequential inference is to recover the posterior distribution of the state at time n, written as $\pi(\mathbf{s}_n|\mathbf{y}_{1:n})$, for $n \in \{1, \ldots, T\}$.

The general form of the recursive formulas for filtering [89] are

$$\widehat{\pi}(\mathbf{s}_n|\mathbf{y}_{1:n-1}) = \int p(\mathbf{s}_n|\mathbf{s}_{n-1})\pi(\mathbf{s}_{n-1}|\mathbf{y}_{1:n-1})d\mathbf{s}_{n-1} \tag{3.5}$$
$$\pi(\mathbf{s}_n|\mathbf{y}_{1:n}) = \frac{g(\mathbf{y}_n|\mathbf{s}_n)p(\mathbf{s}_n|\mathbf{y}_{1:n-1})}{\int g(\mathbf{y}_n|\mathbf{s}_n)p(\mathbf{s}_n|\mathbf{y}_{1:n-1})d\mathbf{s}_n}.$$

Here the first equation is the prediction step which constructs the distribution of the state at time n based on the previous posterior and the transition model. The new observation $\mathbf{y}_n$ is only introduced in the second, update step, where the likelihood of the new measurement based on the predicted state is taken into account to construct the posterior at time n.

In general, the equations (3.5) cannot be solved analytically. This is due to the fact that the state transition is usually non-linear and the likelihood may not be Gaussian. Several algorithms have been developed to solve the filtering problem [62]. Each one has different assumptions about the transition and observation model.

The Kalman filter [98] assumes linear dynamics, and a linear observation model with Gaussian noise. In this case the posterior is guaranteed to be Gaussian, and there is a closed form solution to the filtering equations. The extended Kalman filter (EKF) [99] was proposed to deal with non-linear dynamics. It relies on first-order linearization at each discrete time point where an observation is available, and the propagation of the Gaussian approximation via the linearized dynamics. The EKF can work well in practice, but only when the observation frequency is very high, and therefore the estimate converges despite linearization. The unscented Kalman filter (UKF) [100] further relaxes the assumptions by representing the Gaussian posterior using a deterministically placed set of points (called sigma points) around a mean value. These points are propagated using the true system dynamics, and the mean and covariance of the next posterior is calculated from the empirical moments of the points.

Particle filters (PF) assume a fully sample based representation of the posterior, and implement sampling schemes to directly approximate (3.5). Importantly, they work without restrictions on the transition or observation model. Further, (as a general property of Monte Carlo methods) the accuracy of particle filter estimates does not directly depend on the dimensionality of the posterior.

Several sequential Bayesian inference methods have been used in the context of pathway models. The extended Kalman filter [101, 102] and unscented Kalman filter

Name	Transition model/noise	Observation model/noise	Posterior
Kalman Filter (KF)	Linear/Gaussian	Linear/Gaussian	Gaussian
Extended KF (EKF)	Differentiable/Gaussian	Linear/Gaussian	Gaussian
Unscented KF (UKF)	Non-linear/Gaussian	Non-linear/Gaussian	Unimodal
Particle Filter (PF)	Non-linear/Arbitrary	Non-linear/Arbitrary	Non-parametric

Table 3.1: Recursive Bayesian inference methods on hidden Markov Models

[103] have been applied to parameter estimation in non-linear pathway models. The local linearization employed in the EKF can cause the filter to diverge [101], thereby making it necessary to use traditional parameter estimation methods in addition to filtering. While the UKF is more robust to nonlinearity in the transition model, there is still no guarantee against divergence, as shown, for instance, in [94]. In addition, both EKF and UKF produce a unimodal posterior, uniquely characterized by a mean and covariance matrix, which is not more informative than maximum likelihood parameter estimates with local sensitivities. As we will also see from case studies in Section 3.4, the parameter posteriors will often be far from Gaussian. The covariance-based posterior representation of EKF and UKF can also degrade in high dimensions, since it requires a matrix of size d_θ^2 for d_θ parameters. We therefore focus on particle filters for the parameter inference task.

The introduced sequential inference techniques are summarized in Table 3.1. The next section introduces particle filters in more detail and discusses how they have been applied in pathway modeling.

3.2.3 Particle filters

Particle filters approximate a sequence of probability distributions using sampling techniques. Their flexibility, efficiency in high-dimensions, and applicability in on-line temporal settings made particle filters essential in fields including finance, [104] robotics [105] and geophysical science [106].

The main idea behind particle filtering is to represent a sequence of probability distributions by a set of N random samples called particles. Each particle $\mathbf{s}^i$ has an associated weight w^i, $i \in \{1, 2, \ldots, N\}$. These samples can be used as a discrete support to approximate a probability distribution π as

$$\pi(\mathbf{s}) \approx \sum_{i=1}^{N} w_i \delta(\mathbf{s} - \mathbf{s}^i), \tag{3.6}$$

where δ denotes the Dirac-delta function. Perhaps the most important feature of this

representation is that the particles need not conform to a grid or be spaced according to a parametric rule. This allows particles to concentrate in the most important regions of the distribution they represent. Therefore all resources can be allocated to those parts of a distribution that matter most and parts that contribute less to the probability mass are ignored. This sparsity is an especially important trait when representing high-dimensional distributions.

The objective of particle filtering will be to solve the predict-update equations in (3.5) without explicitly having to compute the integrals involved therein. In order to do this, a sampling procedure is given, which ensures that particles in the posterior at time point n will be approximately distributed according to $\pi(\mathbf{s}_n|\mathbf{y}_{1:n})$, that is, the posterior with observations up to n.

The filter starts by sampling N particles from the prior, and the weights are initialized uniformly. The particles are each propagated according to the model dynamics to the next time point, where their weight is updated based on the likelihood of the current observation. As the filter progresses in time, some of the particles can gradually accumulate very low weights. This means that those samples are not in the high-probability regions of the posterior density. Weight degeneracy can be assessed by quantifying how far the distribution of weights is from being uniform. A commonly used measure is the effective sample size N_{eff} [107], calculated as

$$N_{\text{eff}} = \left(\sum_{i=1}^{N} (\bar{w}_n^i)^2 \right)^{-1}, \tag{3.7}$$

where $\bar{w}_n^i := w_n^i / \sum_{j=1}^{N} w_n^j$ is the normalized weight. A low effective sample size means that most of the samples are located in low-probability areas. This motivates us to eliminate the low-weight particles, and instead, allocate more resources to the promising high-weight particles. Resampling achieves just this goal by taking N samples from the multinomial distribution defined by the weights. More precisely, we resample a new set of particles $\mathbf{s}_r^k$, $k \in \{1, \ldots, N\}$ with replacement where the probability of $\mathbf{s}_r^k$ taking the value of $\mathbf{s}^i$ is $P(\mathbf{s}_r^k = \mathbf{s}^i) = p^i := w_n^i / \sum_{j=1}^{N} w_n^j$. Several resampling schemes have been proposed in the literature that have better variance properties than simple multinomial resampling. These include residual, stratified and systematic resampling [92]. We found that the choice of resampling algorithm has little effect on the quality of the results in our case studies, and therefore do not discuss these in more detail.

Algorithm 1 shows the basic, most commonly used form of the particle filter, which is also called the bootstrap filter [108].

Algorithm 1 Basic particle filter

Input: Number of particles N, measurements $Y = \mathbf{y}_{1:T}$, resample threshold κ
Output: Set of particles $\mathbf{s}_T^i, i \in \{1, \ldots, N\}$

Sample initial particles from prior $\mathbf{s}_0^i \sim p_0(\mathbf{s})$, and set $w_0^i := 1/N$, $i \in \{1, \ldots, N\}$
for n=1...T **do**
    for i=1...N **do**
        Propagate particle $\mathbf{s}_n^i := \mathcal{F}(t_n - t_{n-1}, \mathbf{s}_{n-1}^i)$
        Update weights $w_n^i := w_{n-1}^i g(\mathbf{y}_n | \mathbf{s}_n^i)$
    end for
    if $N_{\text{eff}} < \kappa N$ **then**
        Resample according to probabilities $p^i = w_n^i / \sum_{j=1}^N w_n^j$
        Set $w_n^i := 1/N$, $i \in \{1, \ldots, N\}$
    end if
end for

Using this basic particle filter for *parameter inference* in state space models has first been introduced in time series analysis [109]. The same idea has been applied for hybrid functional Petri-net (HFPN) models of biological pathways in [91] and [110]. This method has been implemented in the graphical tool DA1.0 with many useful functionalities [13]. However, these methods suffer from sample impoverishment due to repeated resampling, resulting in a collapsed representation of the parameter posterior. Since resampling involves duplicating particles exactly, in the worst case, the particle filter can end up with N particles all being the exact copy of a single one. This is especially severe when the state evolves with degenerate dynamics, as in the case of static parameters. One pragmatic solution offered in [93] is to simply increase the sample size and use massive parallelization to avoid particle collapse. A better solution to this problem would be crucial for getting a good approximation to the posterior with limited sample size.

In the context of ODE parameter inference, [94] proposes injecting noise as a way to spread particles at each step of the filter. Following [95], a multivariate Gaussian random vector $\theta' \sim \mathcal{N}(\theta_n^i, \Sigma)$ is picked around each particle thereby diversifying the otherwise static parameters carried by the particles. The main problem with this approach is that the perturbed parameters will generally not be distributed according to the posterior $\pi(\mathbf{s}_n | \mathbf{y}_{1:n})$. This can lead to biased and inaccurate estimates. Our goal in Section 3.3 will be to use a method of particle diversification, which overcomes these limitations.

3.2.4 Making predictions and evaluating particle filters

The goal of particle filtering is to construct a representation of a complicated posterior distribution such that accurate predictions can be made with respect to it. A *prediction* can be formalized as the expected value of a measurable function $f : \Theta \to \mathbb{R}$ with respect to the parameter posterior π, where Θ is the space of parameters. This expectation can be written as an integral

$$\mathbb{E}_\pi f = \int_\Theta f(\theta)\pi(\theta|\mathbf{y}_{1:T})d\theta. \tag{3.8}$$

Having a set of particles $\mathbf{s}^i = (\mathbf{x}^i, \theta^i)$ with associated weights w^i, $i \in \{1, \ldots, N\}$, the expectation can be approximated by a simple weighted average over predictions made by each particle [92] as

$$\mathbb{E}_\pi f \approx \widehat{E}_N := \frac{\sum_{i=1}^N w^i f(\theta^i)}{\sum_{i=1}^N w^i}. \tag{3.9}$$

Here f is only a function of the parameters θ since the model is deterministic. This can also be understood as an instance of Bayesian model averaging, where each particular choice of parameters is a possible hypothesis, and we are averaging over predictions made by each choice.

The theoretical convergence results that exist for particle filters are also understood in terms of such predictions. It is shown in [90] that $\lim_{N\to\infty} \widehat{E}_N = \mathbb{E}_\pi f$ with probability 1 for any bounded function f.

There is a wide variety of predictions we can make in this context. One obvious choice is $f(\theta) := \theta$, in which case the mean of the parameter vector is calculated. The prediction can also be on any of the state variables, based on the result of simulating the system with θ. For instance, the peak level of activity of a protein, or the level it settles at after a certain period of time may be of interest. We can also construct f to express whether the system, when simulated with θ, satisfies a property expressed in temporal logic (similar to the method in Chapter 4).

In the systems biology literature, the performance of particle filters [13, 94] and other sequential Bayesian parameter inference methods [103, 102] has been studied with a criterion other than the above. Namely, the criterion for showing that these methods work in [94, 103, 102] was by assuming a nominal parameter vector θ^*, generating synthetic measurement data with respect to θ^*, and showing that the mean or mode of the distribution of particles is close to θ^*. In [13], evaluation is done based on the

match of the maximum aposteriori parameter to the measurement data. These are both criteria appropriate when the goal is parameter *estimation* rather than parameter *inference*. The performance criteria have not revealed how accurately predictions are made with respect to the Bayesian posterior which is being approximated. In our case studies (see Section 3.4), we will use measures that quantify the prediction accuracy of particle filters for parameter *inference*.

3.2.5 Summary

We reviewed a state space model formulation of ODE based pathway models. The state space formulation allows us to infer hidden states and parameters sequentially by assimilating observations iteratively in time, using the recursive predict-update equations. We then discussed existing sequential inference methods including KF, EKF, UKF and particle filters, and argued that particle filters are the appropriate method in this setting. Particle filtering in its basic form (Algorithm 1) can be used to infer a parameter distribution, but sample impoverishment often leads to degenerate estimates in practice. The methods previously proposed to alleviate this problem when performing ODE pathway parameter inference have important limitations. Finally, we showed how Bayesian predictions can be made using particle filters, and argued that the accuracy of these predictions is an appropriate basis of comparison between different methods.

In the next section we propose methods for inferring pathway parameters based on the application of Markov transition kernels on the particles. The transition kernel is designed to be consistent with the underlying posterior, and will achieve diversification without adding disruptive noise.

3.3 Kernel-enhanced particle filter algorithms

In what follows, we introduce methods for dealing with sample impoverishment in particle filters. The goal is to diversify samples by spreading them randomly, but in a way that the posterior distribution (which we want to approximate) is preserved. This is possible by applying, to each particle, a Markov transition kernel, which is invariant to the posterior. The use of Markov transition kernels on particles was first proposed in the context of target tracking [96] and signal processing [111], and later extended to

static variables in [112]. Here we adapt the idea of using kernel steps to the setting of parameter inference in ODE based pathway models.

Generating a new particle $\mathbf{s}'$ according to kernel K is expressed as $\mathbf{s}' \sim K(\cdot|\mathbf{s})$, where $\mathbf{s}$ is the original particle. The key is to design K as a Markov transition kernel with stationary distribution identical to the target posterior π. Since it is hard to directly sample from such a kernel, we design the kernel in two steps, a proposal and an acceptance step.

We first look at how to design the proposal step. Recall that $\mathbf{s}_n^i = (\mathbf{x}_n^i, \theta_n^i)$ is the ith particle representing the joint state-parameter vector at time n. Since our model is deterministic, $\mathbf{x}_n^i$ is a deterministic function of θ_n^i, through the ODE solution: $\mathbf{x}_n^i = \mathcal{F}_{\theta_n^i}(t_n, \mathbf{x}(0)) = \int_0^{t_n} F(\mathbf{x}(\tau), \theta_n^i) d\tau$. Therefore, when proposing a new position for the particle, $\mathbf{s}'$, we will design a kernel move for the parameters only, and then set the states accordingly as a function of the parameters. Denote the proposal distribution by $q(\mathbf{s} \to \mathbf{s}')$, meaning the probability of proposing $\mathbf{s}'$ when the current particle is $\mathbf{s}$. Then, starting with the particle $\mathbf{s}_n^i = (\mathbf{x}_n^i, \theta_n^i)$, we propose $\mathbf{s}'$ according to the following scheme.

$$\mathbf{s}_n^i = (\mathbf{x}_n^i, \theta_n^i) \quad \longrightarrow \quad \begin{array}{l} \theta' \sim q(\theta_n^i \to \theta') \\ \mathbf{x}' := \mathcal{F}_{\theta'}(t_n, \mathbf{x}(0)) \end{array} \quad \longrightarrow \quad \mathbf{s}' = (\mathbf{x}', \theta').$$

Here the notation $q(\theta_n^i \to \theta')$ for the proposal is meant to highlight that q is only applied on θ. However, from now on we will denote the proposal as $q(\mathbf{s} \to \mathbf{s}')$, and understand that only the parameter is newly proposed by the kernel, and the associated state is set accordingly using simulation.

The acceptance step has to account for the fact that the proposed particle was picked from q, but our goal is to sample from π. We can achieve this by either accepting or rejecting the new particle. Denote the probability of accepting the new particle as $\alpha(\mathbf{s}_n^i \to \mathbf{s}')$. We need to find α such that the detailed balance holds with respect to π [113]. The detailed balance ensures the symmetry of transition between $\mathbf{s}'$ and $\mathbf{s}_n^i$ with respect to π, and is written as

$$\pi(\mathbf{s}'|\mathbf{y}_{1:n}) q(\mathbf{s}' \to \mathbf{s}_n^i) \alpha(\mathbf{s}' \to \mathbf{s}_n^i) = \pi(\mathbf{s}_n^i|\mathbf{y}_{1:n}) q(\mathbf{s}_n^i \to \mathbf{s}') \alpha(\mathbf{s}_n^i \to \mathbf{s}'). \tag{3.10}$$

With rearrangement we obtain the ratio of acceptance rates as

$$\frac{\alpha(\mathbf{s}_n^i \to \mathbf{s}')}{\alpha(\mathbf{s}' \to \mathbf{s}_n^i)} = \frac{q(\mathbf{s}' \to \mathbf{s}_n^i)\pi(\mathbf{s}'|\mathbf{y}_{1:n})}{q(\mathbf{s}_n^i \to \mathbf{s}')\pi(\mathbf{s}_n^i|\mathbf{y}_{1:n})}. \tag{3.11}$$

One choice of α, which trivially satisfies (3.11) was introduced by Metropolis and Hastings [114], and is written as

$$\begin{aligned}\alpha(\mathbf{s}_n^i \to \mathbf{s}') &= \min\left(1, \frac{q(\mathbf{s}' \to \mathbf{s}_n^i)}{q(\mathbf{s}_n^i \to \mathbf{s}')}\frac{\pi(\mathbf{s}'|\mathbf{y}_{1:n})}{\pi(\mathbf{s}_i^n|\mathbf{y}_{1:n})}\right) \\ &= \min\left(1, \frac{q(\mathbf{s}' \to \mathbf{s}_n^i)}{q(\mathbf{s}_n^i \to \mathbf{s}')}\frac{p_0(\mathbf{s}')p(\mathbf{y}_{1:n}|\mathbf{s}')}{p_0(\mathbf{s}_i^n)p(\mathbf{y}_{1:n}|\mathbf{s}_n^i)}\right).\end{aligned} \tag{3.12}$$

There are several important properties to note in (3.12). Due to the fact that only the ratio of the posterior probabilities is needed, we can ignore the normalization constant and directly write the ratios of the prior times the likelihood (second equation). It is clear that to evaluate the acceptance rate the likelihood of both $\mathbf{s}'$ and $\mathbf{s}_i^n$ needs to be calculated up to time n. To avoid simulating $\mathbf{s}_i^n$ multiple times, it is possible to save the likelihood values up to n during the filtering process. In case of the new particle $\mathbf{s}'$, the proposal itself involves simulating $\mathbf{x}'$ based on the proposed θ', and allows us to calculate the likelihood of the trajectory up to n. The trajectory likelihood is calculated as

$$p(\mathbf{y}_{1:n}|\mathbf{s}') = \prod_{k=1}^{n} g(\mathbf{y}_k|\mathbf{s}'_k), \tag{3.13}$$

where the likelihood of the particle only depends on the state, that is, $g(\mathbf{y}_k|\mathbf{s}'_k) = g(\mathbf{y}_k|\mathbf{x}'_k)$, and $\mathbf{x}'_k = \mathcal{F}_{\theta'}(t_k, \mathbf{x}(0))$ is the state at time k simulated using θ'.

3.3.1 Particle filter algorithm with kernel steps

We now present the main algorithm of this section, the particle filter with kernel steps. Algorithm 2 uses the proposal q in a generic form. There are many different choices for a proposal distribution, and the choice can have a significant effect on the performance of the algorithm. In the next section, we suggest choices that are expected to work well for pathway models.

3.3.2 Sampling strategies

Here we look at possible ways of proposing new particles based on the current ones. Due to weighting and resampling, the existing particles (even if collapsed) will generally

Algorithm 2 Kernel-enhanced particle filter

Input: Number of particles N, measurements $Y = \mathbf{y}_{1:T}$, resample threshold κ
Output: Set of particles $\mathbf{s}_T^i, i \in \{1, \ldots, N\}$

Sample initial particles from prior $\mathbf{s}_0^i \sim p_0(\mathbf{s})$, and set $w_0^i := 1/N$, $i \in \{1, \ldots, N\}$
for n=1...T **do**
    for i=1...N **do**
        Propagate particle $\mathbf{s}_n^i := \mathcal{F}(t_n - t_{n-1}, \mathbf{s}_{n-1}^i)$
        Update weights $w_n^i := w_{n-1}^i g(\mathbf{y}_n | \mathbf{s}_n^i)$
    end for
    if $N_{\text{eff}} < \kappa N$ **then**
        Resample according to probabilities $p^i = w_n^i / \sum_{j=1}^N w_n^j$
        Set $w_n^i := 1/N$, $i \in \{1, \ldots, N\}$
    end if
    for i=1...N **do**
        Propose $\theta' \sim q(\theta_n^i \to \theta')$ and simulate with θ' to get $\mathbf{x}'$
        Set $\mathbf{s}' := (\mathbf{x}', \theta')$
        Evaluate acceptance rate $\alpha(\mathbf{s}_n^i \to \mathbf{s}')$
        Generate $\eta \sim \text{Uniform}[0, 1]$
        if $\eta < \alpha(\mathbf{s}_n^i \to \mathbf{s}')$ **then**
            Accept new particle and set $\mathbf{s}_n^i := \mathbf{s}'$
        end if
    end for
end for

be concentrated in higher probability regions. Therefore it makes sense to propose the kernel moves based on the current set of particles. In each case, we need to derive an appropriate acceptance rate to account for the fact that the proposal distribution is not equal to the posterior.

Particle based fixed proposal

The first kernel we consider involves a Gaussian proposal of predetermined width around each particle. For instance, if the current particle is $\mathbf{s}_n^i = (\mathbf{x}_n^i, \theta_n^i)$, the proposal would be $\theta' \sim \mathcal{N}(\theta_i^n, \Sigma)$. Here Σ is the covariance matrix of the proposal. The entries of the covariance matrix can greatly affect its efficiency. Namely, for a covariance matrix with small entries, the random proposal will be close to the original particle. This will typically result in high acceptance rates, but the resulting particles will still cluster around the original ones. Larger variance terms will imply lower acceptance rates but potentially more diversification. In our case studies we set the diagonal values of Σ proportional to the prior range of each unknown parameter (in case of normal and log-normal priors it can be set proportional to the prior variance of each unknown parameter).

The proposal in this case is clearly symmetric, that is, $q(\mathbf{s} \to \mathbf{s}') = q(\mathbf{s}' \to \mathbf{s})$ and therefore the acceptance rate simplifies to the ratios of priors and likelihoods as

$$\alpha(\mathbf{s}_n^i \to \mathbf{s}') = \min\left(1, \frac{p_0(\mathbf{s}')p(\mathbf{y}_{1:n}|\mathbf{s}')}{p_0(\mathbf{s}_i^n)p(\mathbf{y}_{1:n}|\mathbf{s}_i^n)}\right). \tag{3.14}$$

We will refer to this method as PF-KGAUSS.

Adaptive proposal towards population mean

The previous method could have the disadvantage that proposed moves are clustered around the existing individual particles. Further, the variance entries in the proposals need to be set based on prior knowledge or tuned manually. It is possible to exploit the population of particles to build adequate proposals. Namely, we can calculate the population mean and variance of the particles, and use this information in the proposal. This idea appeared in [95] and [94], but there it was used as a method to inject random noise, rather than as a proposal step for a Markov kernel.

The population mean ($\hat{\mu}_n$) and covariance matrix ($\hat{\Sigma}_n$) of the particles can be calculated as follows.

$$\hat{\mu}_n = \frac{\sum_{i=1}^N w_n^i \theta_n^i}{\sum_{i=1}^N w_n^i}. \tag{3.15}$$

$$\hat{\Sigma}_n = \frac{\sum_{i=1}^N w_n^i (\theta_n^i - \hat{\mu}_n)(\theta_n^i - \hat{\mu}_n)^T}{\sum_{i=1}^N w_n^i}. \tag{3.16}$$

The idea is to propose new parameters such that the mean of the proposal is shifted towards the population mean, and the variance of the proposal is scaled according to the population variance. The proposed parameter θ' will be centered around the shifted mean μ_n^i as

$$\mu_n^i := a\theta_n^i + (1-a)\hat{\mu}_n \tag{3.17}$$

$$\theta' \sim \mathcal{N}(\mu_n^i, h^2\hat{\Sigma}_n), \tag{3.18}$$

where $a = (3\delta - 1)/2\delta$, $h^2 = 1 - a^2$, and $\delta \in (1/3, 1)$ is a discount factor [95]. This construction is appealing, since it preserves the original empirical mean and variance ($\hat{\mu}_n$ and $\hat{\Sigma}_n$) [115]. Further, δ is the only parameter which needs to be chosen by the user. Generally the value of δ, as recommended by [95], is chosen around $0.95 - 0.99$.

To adapt this proposal for our kernel based setting we need to derive an acceptance

rate, which guarantees the preservation of the posterior. In this case the proposal around the original parameter is not symmetric, since it is shifted towards the population mean. Consequently the ratio of proposal probabilities in (3.12) cannot be eliminated. The ratio of the proposal probabilities can be specified as

$$\frac{q(\mathbf{s}' \to \mathbf{s}_n^i)}{q(\mathbf{s}_n^i \to \mathbf{s}')} = \frac{\phi(\theta_n^i; a\theta' + (1-a)\hat{\mu}_n, h^2\hat{\Sigma}_n)}{\phi(\theta'; a\theta_n^i + (1-a)\hat{\mu}_n, h^2\hat{\Sigma}_n)}, \tag{3.19}$$

where $\phi(x; \mu, \Sigma)$ is the value of the multivariate normal density function with mean μ and covariance Σ, evaluated at x. This ratio will need to be used in (3.12) when calculating the acceptance rate.

We will refer to this method as PF-KSHIFT.

Adaptive population based proposal

It is also possible to build a proposal only based on the population mean and variance. The only way the proposed particles will depend on the original ones (and guaranteed to still be distributed according to the posterior) is through the acceptance rate. Recall the population mean and variance from (3.15) and (3.16). A new particle can be proposed as

$$\theta' \sim \mathcal{N}(\hat{\mu}_n, \hat{\Sigma}_n). \tag{3.20}$$

Each proposed particle will still be accepted or rejected with respect to its corresponding θ_n^i. Here, again, the proposal is not symmetric, and the proposal ratio appearing in the acceptance rate (3.12) needs to be calculated as

$$\frac{q(\mathbf{s}' \to \mathbf{s}_n^i)}{q(\mathbf{s}_n^i \to \mathbf{s}')} = \frac{\phi(\theta_n^i; \hat{\mu}_n, \hat{\Sigma}_n)}{\phi(\theta'; \hat{\mu}_n, \hat{\Sigma}_n)}, \tag{3.21}$$

where, again, $\phi(x; \mu, \Sigma)$ is the value of the multivariate normal density function with mean μ and covariance Σ, evaluated at x.

We will refer to this method as PF-KPOP.

3.3.3 Computational cost

Performing kernel steps requires additional computation since the acceptance rate calculation involves simulating the newly proposed particle. The basic particle filter does not use this step, and is, therefore faster.

Assume that the total number of particles is N, and that data is available at T uni-

formly spaced time points. The dominating computational cost of running the particle filter is the numerical simulation of ODEs. We will assume that the ODE solver has a linear time complexity, that is, solving the ODE equations for n time steps takes $O(n)$ time. The basic particle filter propagates particles sequentially from time n to $n+1$, for $n = 0, 1, 2 \ldots T$, meaning that each particle is simulated once up to T, resulting in a computational cost of $O(NT)$.

The kernel-enhanced filter, on top of the cost of the basic filter, will perform additional steps. At each time step n, one needs to simulate the N newly proposed particles from the initial time up n. For one iteration this is an additional cost of $O(Nn)$. The overall cost of the kernel steps for $n = 1, 2, \ldots, T$ will therefore be $N \sum_{n=1}^{T} O(n) = O(NT^2)$, resulting in a total cost of $O(NT^2)$ to run the filter. This implies that the kernel-enhanced methods will have increased computational cost if measurements are available at many time points. However, there are several other arguments to consider.

First, as the number of measurement points increases, the additional information gain from each data point will generally decrease. From the perspective of particles, this means that the effective sample size will change slowly across time steps. This implies that one could also decrease the frequency of kernel steps without degrading performance. In particular, if the frequency of kernel steps is set proportional to the *simulation time* rather than the number of measurement time points, then the original $O(NT)$ cost is preserved.

Second, the accuracy of making predictions with a *fixed number of particles* is more important than the time needed to run the particle filter. This is because generally, the particle filter is ran once, and the particles are saved as a representation of the parameter posterior for later use. Any prediction, which involves simulation, that is made subsequently with the particles, will have a cost linear in N. Therefore we argue that it is better to run the particle filter once, with higher computational cost, if the stored particles will subsequently give more accurate predictions.

A more practical argument, also made in [13], is that increasing the number of particles is limited by the available memory. There is more flexibility in increasing processing time with a fix particle number, especially if a multi-threaded architecture is available. Accuracy with a limited number of particles will be even more important when

considering a GPU implementation of the particle filter [116], which is a memory-limited environment.

Finally, as our case studies in the next section show, kernel-enhanced filters can reach the same accuracy as the basic particle filter with significantly reduced sample size, and this makes up for the additional time needed to run the filter.

3.4 Case studies

First we construct a small, synthetic example to show the robustness of kernel-enhanced methods compared to other particle filters. Then we use a model of the JAK-STAT pathway to compare different particle filters in making predictions with respect to the Bayesian posterior. This model has been used in several works related to Bayesian inference including evaluating unscented Kalman filters [103], as well as Bayesian uncertainty analysis and experimental design [117, 118]. Its use in the Bayesian inference context as a benchmark is due to the fact that there is considerable parameter uncertainty given the available experimental data. It is also relevant since wet-lab experimental data has been published for the model, allowing computational methods to be evaluated in a realistic setting.

We evaluate the three proposed kernel-enhanced particle filters, PF-KGAUSS (Gaussian proposal around each particle), PF-KSHIFT (proposal shift towards population mean) and PF-KPOP (proposal from population distribution), and compare these methods with ones previously used for pathway parameter inference. PF-BASIC is the basic particle filter shown in Algorithm 1. PF-NGAUSS injects noise into the parameters by moving them randomly using a multivariate Gaussian (similarly to PF-KGAUSS, but without the acceptance step). PF-NSHIFT implements the noise injection method proposed by Liu and West [95], and is the basis of the kernel-enhanced PF-KSHIFT method (see Section 3.3). For all particle filters, resampling was done only when the effective sample size was below $N/3$, where N is the number of particles. We implemented PF-NGAUSS and PF-KGAUSS with an uncorrelated Gaussian around each particle whose standard deviation along each parameter dimension was equal to 0.2 times the range of the parameter. For the PF-NSHIFT and PF-KSHIFT methods, we set the parameter δ to 0.95. The CVODE stiff solver was used to numerically solve ODEs [31]. We exploited

the property that particles can be propagated in parallel, and ran all experiments on 12 cores with 2.27GHz Intel Xeon CPUs.

3.4.1 Enzyme-substrate process

An important limitation of the basic particle filter is that the parameters, sampled initially, only get resampled, but their value does not change over time. Once a particle is eliminated due to resampling it cannot be recovered, even if subsequent measurements would assign higher probabilities to it. Noise injection based methods do have the potential to recover from such a situation, but the random nature of perturbing particles will decrease their efficiency. We expect that the kernel-enhanced methods, through the principled diversification of particles according to the posterior, will show much better robustness.

We illustrate this on a small, synthetic example of an enzyme catalyzed reaction consisting of 4 species and 3 parameters. The model represents an enzyme which binds to a substrate reversibly, and then releases a product, which accumulates over time. The ODE equations are given in Figure 2.3 (Section 2.2.1). We set the parameters $k_1 = 0.1$, $k_2 = 0.1$ and $k_3 = 0.35$, and set the initial conditions of substrate to 15 and enzyme to 10 units. A uniform prior distribution was assumed over the parameters, with the ranges for parameters being $k_1, k_2, k_3 \in [0, 1]$. We generated synthetic experimental data with the nominal parameters for the product concentration $[P]$ at the equally spaced time points $1, 2, \ldots, 10$, and added zero-mean Gaussian observation noise with standard deviation of 0.5.

We then perturbed the measurement of the product at time 1 by adding a uniform random number to it between 4 and 5. This perturbation has little effect on the overall likelihood landscape. However, it can have a huge effect on the performance of particle filters. This is because the filters progress forward in time and resample particles based on the likelihood of the current measurements. Hence the first measurement point could result in particles being early on resampled in a region which is not representative of the overall posterior. This leads to particle collapse from which the particle filter should, ideally, recover.

We ran each type of particle filter 200 times, independently, with $100, 1000$ and 10000 particles. We quantified how well the resulting particles fit the measurements through

the trajectory likelihood $p(\mathbf{y}_{1:T}|\mathbf{s}^i)$, where $\mathbf{y}$ are the measurements, $T = 10$ and $\mathbf{s}^i$ is the ith particle. We calculated the logarithm of the mean of the trajectory likelihoods for each run and plotted these values across 200 runs in Figure 3.2 (top). Here the interval between the 5% and 95% quantile of log-likelihood values is shown.

To categorize which particle filter runs could "recover" after particle collapse and converge to the high-probability region of parameter space, we set a threshold on the trajectory log-likelihoods at -50 and calculated the percentage of runs which ended above this threshold. We show the percentage of runs in which the particle filter could recover and converge in Figure 3.2 (bottom).

The basic particle filter performs poorly, and even with 10000 particles, the log-likelihoods are below -100. This is not surprising since the basic filter can only initially sample and then resample particles; there is no mechanism to move them towards better regions of parameter space upon particle collapse. Conversely, PF-KGAUSS and PF-KPOP consistently achieve a log-likelihood above -25, and PF-KSHIFT behaves similarly for larger particle numbers.

The noise injection based methods could in some cases converge to the correct region, but not as consistently as kernel based methods. Investigating the relatively weaker performance of PF-KSHIFT compared to the other two kernel based methods revealed that after resampling in the first time step, the kernel proposal had very low variance, and resulted in slower diversification. PF-KGAUSS and PF-KPOP had wider variability in the proposal steps, which resulted in a faster and more consistent recovery.

In Figure 3.3, simulations are shown based on the result of different particle filter methods (with 1000 particles). The gray shading shows the 5% to 95% quantile of simulation trajectories. It is clear that PF-BASIC produces a collapsed estimate, that is, only very few distinct particles remain due to resampling. PF-NSHIFT shows more diversity but does not converge to the high-probability region, and while PF-NGAUSS does converge, it has larger variability than the level of measurement noise would suggest. The kernel based methods show good convergence and their spread is consistent with the noise level of measurements. The spread of particles at each time step of the basic filter and one kernel-enhanced filter is shown in the Appendix (Figure A.1).

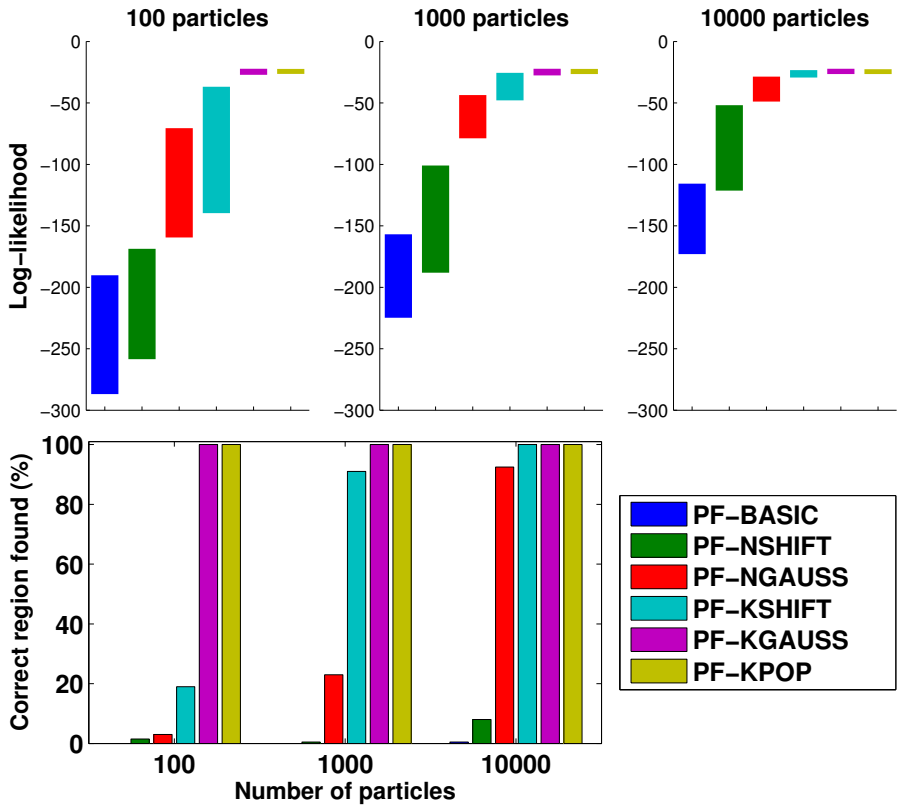


Figure 3.2: Performance of particle filters on the model of an enzyme-substrate process.

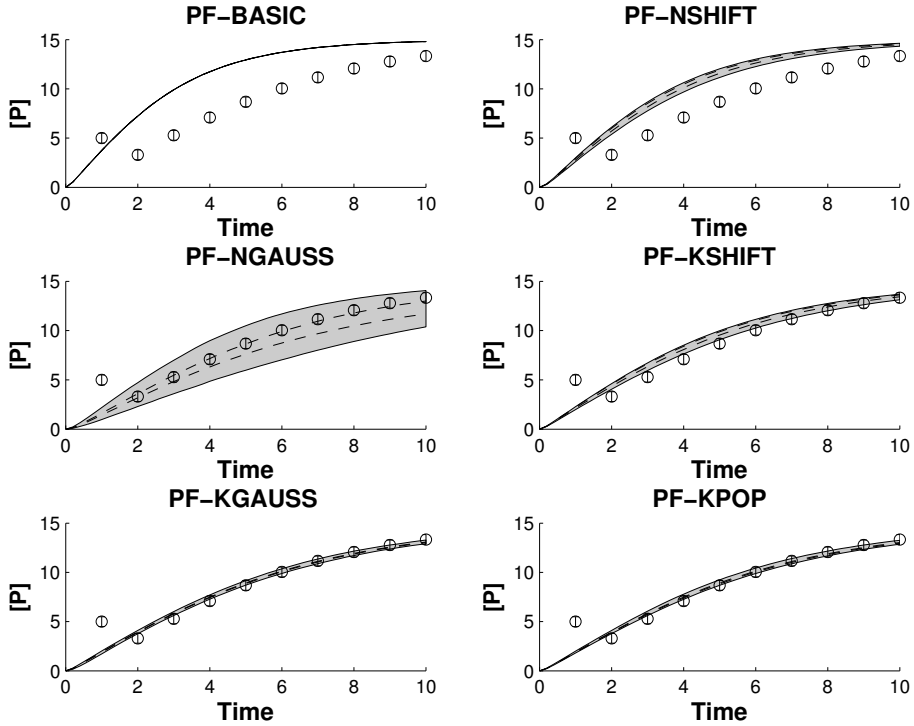


Figure 3.3: Fit of 1000 particles to measurements of an enzyme-substrate process with different particle filter methods. Gray shading corresponds to the 5% to 95% quantile covered by particle trajectories, according to particle weights. Circles with error bars show the measurement data. Dashed lines correspond to weighted mean and median trajectories.

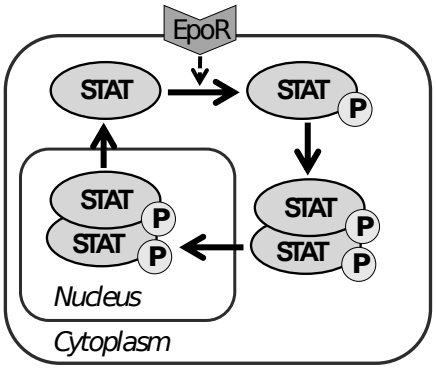
3.4.2 The JAK-STAT pathway

In this section we compare the performance of particle filters on a model of the JAK-STAT signaling pathway.

The JAK-STAT signaling cascade is initiated by erythropoietin (Epo), which, when bound to a receptor, induces the phosphorylation of STAT protein in the cytoplasm. Phosphoylated STAT dimerizes and enters the nucleus where it alters gene expression. Subsequently the nuclear STAT goes through dissociation and dephosphorylation and is transported back into the cytoplasm (see also [118]). The species in the model are listed in Table 3.2, and the set of ODE equations describing the dynamics are given in Figure 3.4.

Name	Description	Initial amount
Epo	Erythropoietin, input stimulus	2.0
STAT	Unphosphorylated STAT monomer in cytoplasm	0
STATp	Phosphorylated STAT monomer in cytoplasm	0
STATpd	Phosphorylated STAT dimer in cytoplasm	0
STATn	Total STAT in nucleus	0
$X_1 \dots X_K$	Represent delay in STAT exiting nucleus (we use $K = 10$)	0

Table 3.2: Species in the JAK-STAT model.



$$\frac{d[\text{STAT}]}{dt} = -k_1[\text{STAT}][\text{Epo}] + 2k_4[X_K]$$
$$\frac{d[\text{STATp}]}{dt} = k_1[\text{STAT}][\text{Epo}] - k_2[\text{STATp}]^2$$
$$\frac{d[\text{STATpd}]}{dt} = -k_3[\text{STATpd}] + 0.5k_2[\text{STATp}]^2$$
$$\frac{d[\text{X}_1]}{dt} = k_3[\text{STATpd}] - k_4[\text{X}_1]$$
$$\frac{d[\text{X}_j]}{dt} = k_4[\text{X}_{i-1}] - k_4[\text{X}_i], \quad j = 2 \dots K$$
$$\frac{d[\text{STATn}]}{dt} = k_3[\text{STATpd}] - k_4[\text{X}_K]$$

Figure 3.4: ODE model of the JAK-STAT pathway under Epo stimulation.

The variables in the model and the 4 kinetic rate constants ($\theta = (k_1, k_2, k_3, k_4)$) cannot be directly measured. However, experimental data for two indirect quantities (total phosphorylated STAT, and total STAT in cytoplasm) has been published in [119]. The experiments report the mean and standard deviations of these two quantities at 17 time points. Figure 3.5 shows the experimental data. We use Gaussian likelihood (based on the measurement means and standard deviations) when comparing the data to

simulated trajectories. A uniform prior distribution over a range of possible parameter values is chosen as $p_0 : k_1 \sim U[0,5], k_2 \sim U[0,30], k_3 \sim U[0,1], k_4 \sim U[0,5]$.

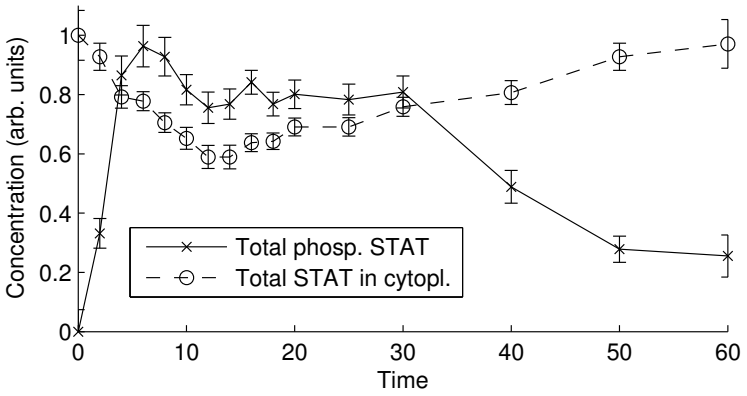


Figure 3.5: Experimental data for the JAK-STAT pathway [119]. The Gaussian likelihood is evaluated using the shown data points and standard deviations.

Prediction accuracy

In order to compare how accurately the different particle filters make predictions, we needed to establish a reference, "ground truth" estimate. We chose a simple, importance sampling based estimate with very high sample size as reference. This method involves first sampling independently from the prior as $\mathbf{s}^i \sim p_0(\mathbf{s})$, and then calculating importance weights as $w^i = p(\mathbf{y}_{1:T}|\mathbf{s}^i)$. The prediction is then made over N samples as

$$\widehat{E}_N = \frac{\sum_{i=1}^N w^i f(\mathbf{s}^i)}{\sum_{i=1}^N w^i}, \tag{3.22}$$

where f is the function being predicted. By the strong law of large numbers, this estimate will be consistent, that is $\mathbb{E}_\pi f = \lim_{N\to\infty} \widehat{E}_N$. Naturally, this method of estimation is not efficient in general since many samples will come from low-probability regions. But we found that with a sample size of at least 10^6, the estimate has small enough variance around its mean to serve as a good reference. To this end, we ran importance sampling 100 times with a total of 10^6 samples, and averaged out the $\widehat{E}_N$ values across runs to obtain the reference value.

We then ran each particle filter repeatedly, with a range of particle numbers and stored the resulting particles in order to evaluate multiple predictions. As an illustration of the result of the particle filter algorithm, we plot particles obtained in a single execution for each considered method. Figure 3.6 shows the positions of 1000 particles in

parameter space obtained using the different particle filters. The kernel-based methods show good diversity among samples and indicate that the high-probability regions are clearly represented. The basic particle filter degenerates and only 3 distinct particles remain. The noise injection methods show significantly more diversity, while weights are closer to uniform, indicating that predictions will be disrupted by samples that are spread to lower probability regions. Similar properties are seen when plotting the simulation trajectories corresponding to the 1000 particles against the experimental data, as shown in Figure 3.7.

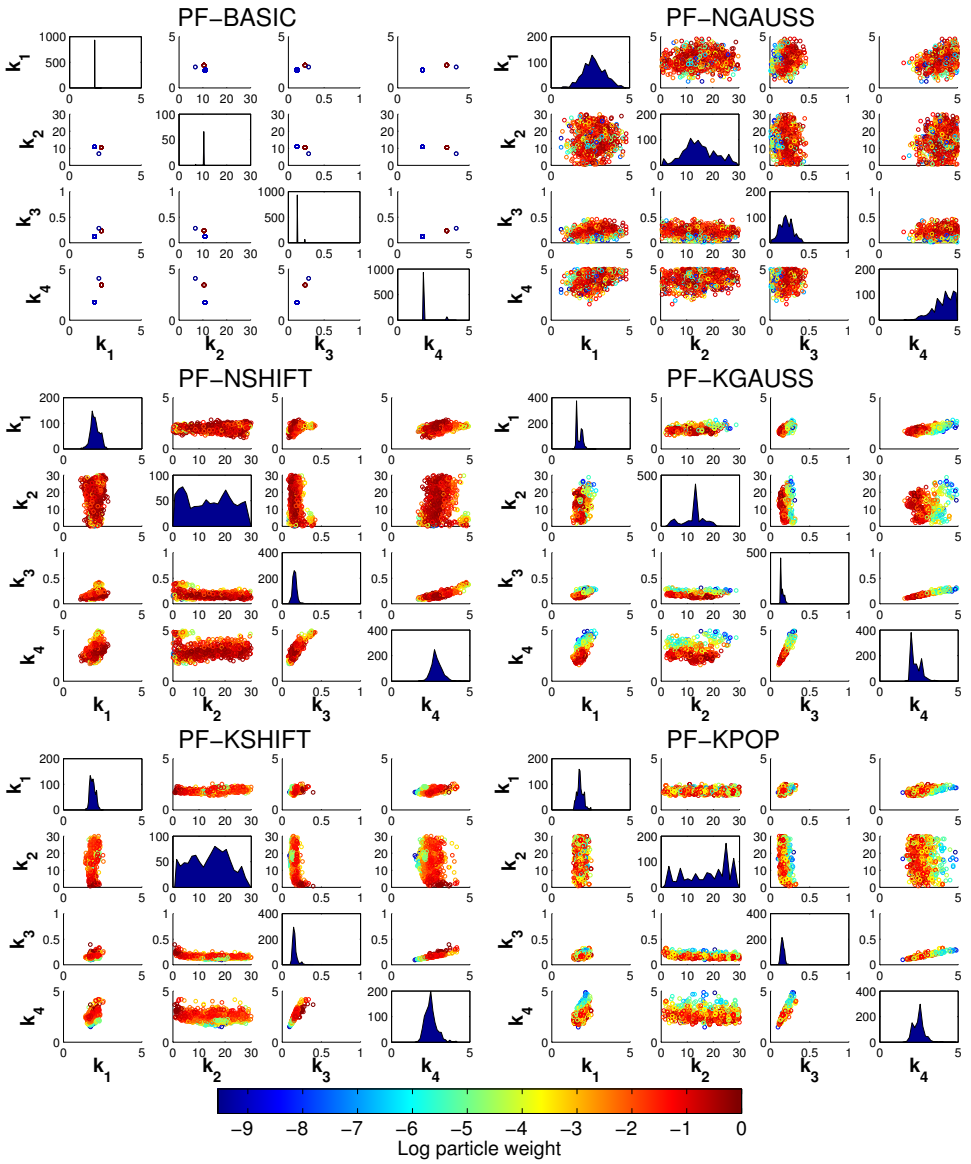


Figure 3.6: Scatter plots and histograms of 1000 particles with different particle filter methods. The plot in row i and column j shows either the 2-dimensional projection of particles on the parameters k_i and k_j if $i \neq j$, or the weighted histogram of a parameter k_i if $i = j$.

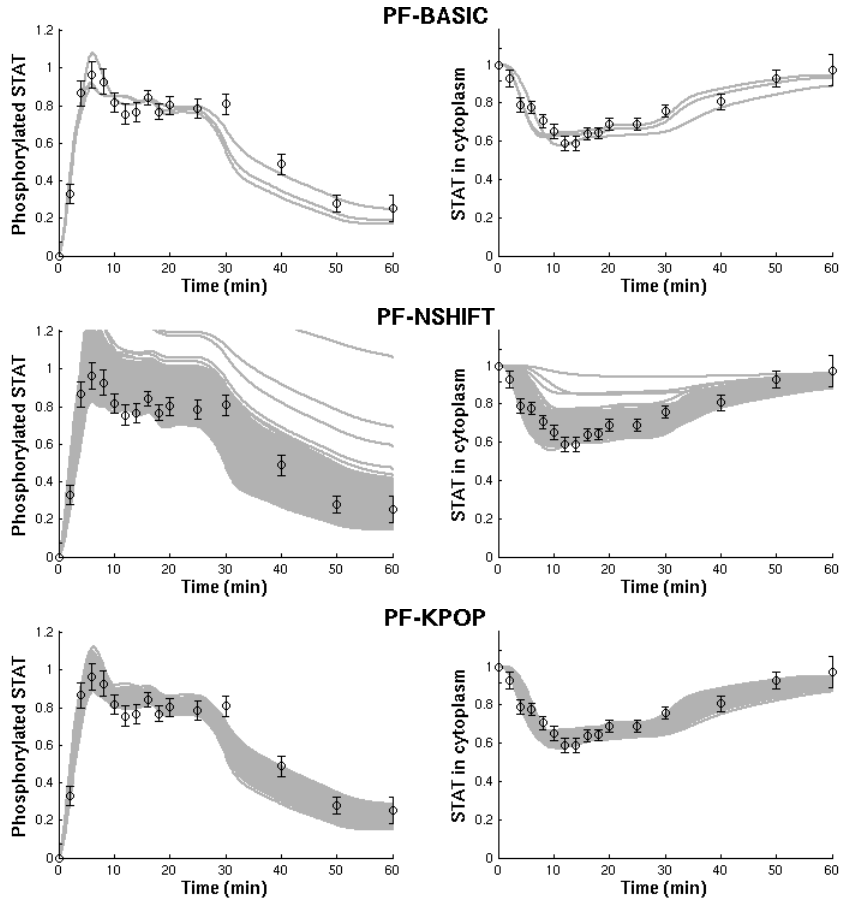


Figure 3.7: Fit to experimental data with 1000 particles with different particle filter methods. The plots for PF-KGAUSS and PF-KSHIFT are similar to PF-KPOP and are given in the Appendix (Figure A.2).

In what follows, for each particle number (10 values between 100 and 100000), we calculated statistics for 200 independent runs of each particle filter. We wanted to take into account both the variance and the bias of our estimates, and therefore chose the mean squared error (MSE) with respect to the reference as the measure of prediction accuracy. We found that PF-NGAUSS gave very poor predictions compared to other filters and therefore we chose to exclude it from the remaining results.

The first task we considered was estimating the mean of parameters k_1, k_2, k_3 and k_4, whose reference values were estimated as $1.8213, 17.6235, 0.1531$, and 2.4480, respectively. The results in Figure 3.8 show the average MSE across 200 runs for PF-BASIC, PF-NSHIFT and three kernel enhanced methods. We see that PF-NSHIFT, which injects noise at each step of the filter results in a lack of convergence to the reference in two cases (k_1, k_4). The PF-BASIC method has decreasing MSE as particle numbers grow, however, the variance across runs is much higher than in the case of the kernel-

enhanced methods (PF-KSHIFT, PF-KGAUSS and PF-KPOP), and hence its MSE is consistently above the other methods.

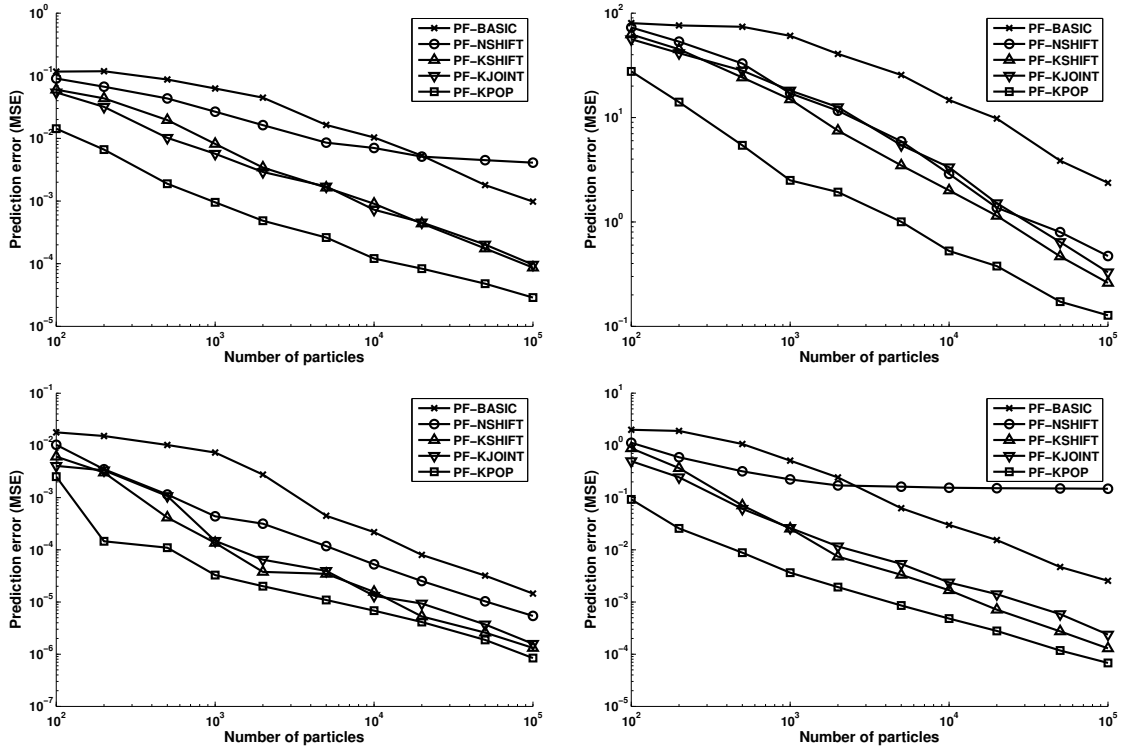


Figure 3.8: Estimating the mean of parameters k_1, k_2, k_3, k_4, showing the mean squared error of estimates.

Similar differences in performance are found when considering the behavior of the unobserved species in the pathway, as predicted by the particles. We made predictions about the behavior of nuclear STAT, specifically, we were interested in the peak amount of nuclear STAT reached within 60 minutes. The reference value, estimated using importances sampling was 1.1895. The results in Figure 3.9 show that the kernel-enhanced particle filters made accurate predictions even with very low sample sizes. The PF-NSHIFT method did not converge to the true value as the number of particles was increased. PF-BASIC showed decreasing MSE with growing sample size, but produced predictions with around 2 orders of magnitude higher MSE compared to PF-KPOP.

Next we looked at how much STAT monomer remains in the cytoplasm, once Epo stimulation has ended, at the 60 minute time point. The reference value for this estimate was 1.5387. The results are shown in Figure 3.10. Again we found that PF-BASIC had much higher MSE compared to the kernel-enhanced methods. PF-NSHIFT showed decreasing MSE up to 20000 particles, but the MSE failed to decrease further with

higher sample sizes.

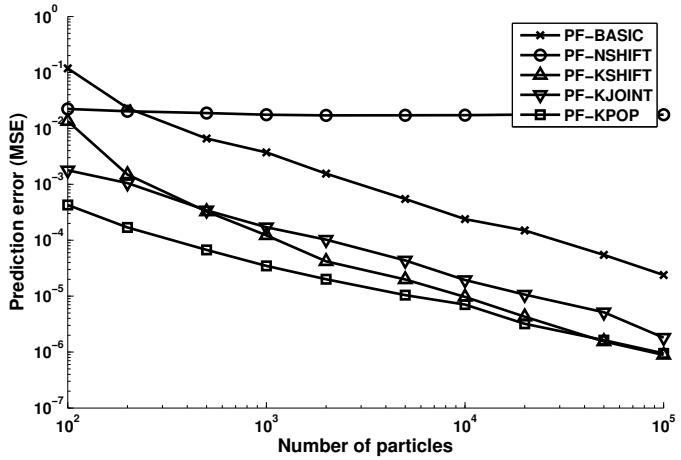


Figure 3.9: Estimating the peak amount of nuclear STAT, showing mean squared error of estimates.

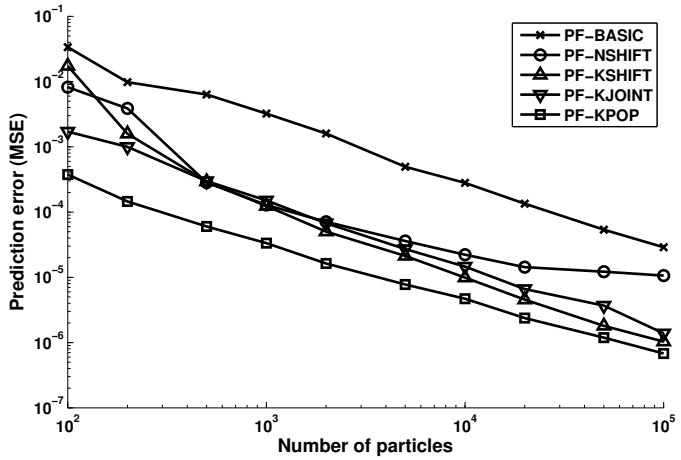


Figure 3.10: Estimating the amount of cytoplasmic STAT monomer at the last time point, showing mean squared error of estimates.

Sample size and runtime efficiency

In all cases we found that PF-KPOP provided the lowest MSE for a given number of particles. We therefore compared the relative efficiency of particle filters with PF-KPOP as a baseline. Specifically we looked at how many particles are needed with a given method to reach the same MSE as PF-KPOP with 1000 particles. We used interpolation on the mean square errors for each particle filter method to arrive at the approximate number of particles needed to match the MSE of PF-KPOP. The results are shown in Table 3.3. We see that PF-BASIC needs between 49.6 and 101.5 times as many particles as PF-KPOP to reach the same accuracy. PF-NSHIFT is comparable to

PF-KGAUSS and PF-KSHIFT in 2 out of 6 predictions, but performs much worse on predicting k_3, and fails to converge in 3 our of 6 cases. In summary, the noise injection method shows good performance in some cases but is not reliable for making predictions in general.

Prediction	PF-KPOP	PF-KGAUSS	PF-KSHIFT	PF-NSHIFT	PF-BASIC
k_1	1000	8850	9671	*	101503
k_2	1000	14585	8312	12532	95612
k_3	1000	6285	5500	17274	49558
k_4	1000	7880	4758	*	74438
Max. STATn	1000	6824	2986	*	82480
Final STAT	1000	4538	3744	5946	91006

Table 3.3: Estimated number of particles needed to reach same accuracy (MSE) as PF-KPOP with 1000 particles. (*estimate not reliable since MSE does not converge)

The results in Table 3.3 mean that the kernel-enhanced methods, and in particular PF-KPOP have much better *sample size efficiency* than both PF-BASIC and PF-NSHIFT, and PF-BASIC will need, on average 82.4 times the number of samples to reach the same accuracy as PF-KPOP.

We have seen that the sample size efficiency of kernel based methods are much better than other particle filters. This is important, since we are usually interested in finding the best possible set of particles, for a chosen sample size, to represent the parameter posterior. However, it is also important to consider the time taken to run the filter, and as discussed in Section 3.3.3, kernel-enhanced methods will generally be slower due to the additional cost of kernel steps. In Figure 3.11 we show the runtime of the PF-BASIC and PF-KPOP methods for different particle numbers. The other methods are not plotted, since the kernel based methods closely resemble PF-KPOP and the noise injection methods are similar to PF-BASIC in terms of runtime. Clearly, the basic particle filter is faster, and the ratio of runtimes for kernel based methods stabilizes at around 5 times that of the basic particle filter.

To quantify the relationship between runtime and prediction accuracy, we calculated how much time it takes to run each particle filter to reach the same accuracy as that of PF-KPOP for 1000 particles. To do this, we referred to the estimated particle numbers in Table 3.3, and then used these sample sizes to estimate the time needed to run each particle filter. The results are shown in Table 3.4. The PF-KPOP method takes, on average, 8.74 seconds with 1000 particles, and to reach the same accuracy, the PF-BASIC method will need 56.12 seconds on average. This means that, even though the

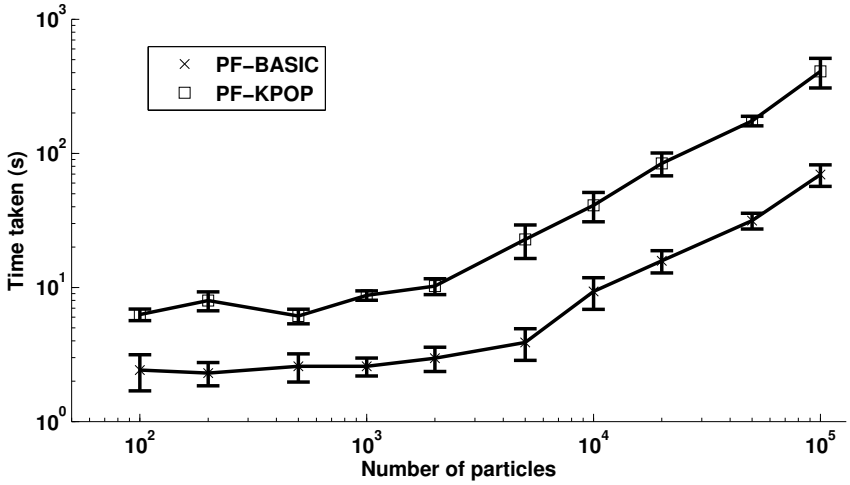


Figure 3.11: Particle filter average runtimes depending on the number of particles used. The runtime of PF-NSHIFT matches PF-BASIC, and the runtime of PF-KGAUSS and PF-KSHIFT are closely overlapped with PF-KPOP, hence they are not shown.

basic particle filter is faster, it needs 6.4 times the runtime to reach the same accuracy as PF-KPOP due to the need for a much higher number of particles. The other kernel-enhanced methods outperform the basic particle filter in all cases, but the results are close, with PF-KSHIFT being somewhat better than PF-KGAUSS. Interestingly, the noise injection method PF-NSHIFT performs only worse than PF-KPOP by a small margin in 3 out of 6 cases, but in the other 3 cases it does not converge, and therefore cannot reach the reference accuracy, even with very high sample size and runtime.

Prediction	PF-KPOP	PF-KGAUSS	PF-KSHIFT	PF-NSHIFT	PF-BASIC
k_1	8.74	39.6	38.3	*	70.6
k_2	8.74	62.6	34.0	11.5	66.1
k_3	8.74	28.2	25.0	13.3	31.3
k_4	8.74	35.3	22.4	*	50.0
Max. STATn	8.74	30.6	14.4	*	56.1
Final STAT	8.74	20.8	17.8	9.0	62.6

Table 3.4: Estimated time needed (in seconds) to reach same accuracy (MSE) as PF-KPOP with 1000 particles. (*estimate not reliable since MSE does not converge)

We finally summarize the overall sample size efficiency and runtime efficiency of each particle filter compared to PF-KPOP. The sample size and runtime required by each particle filter to reach the same accuracy as PF-KPOP with 1000 particles is shown in Figure 3.12.

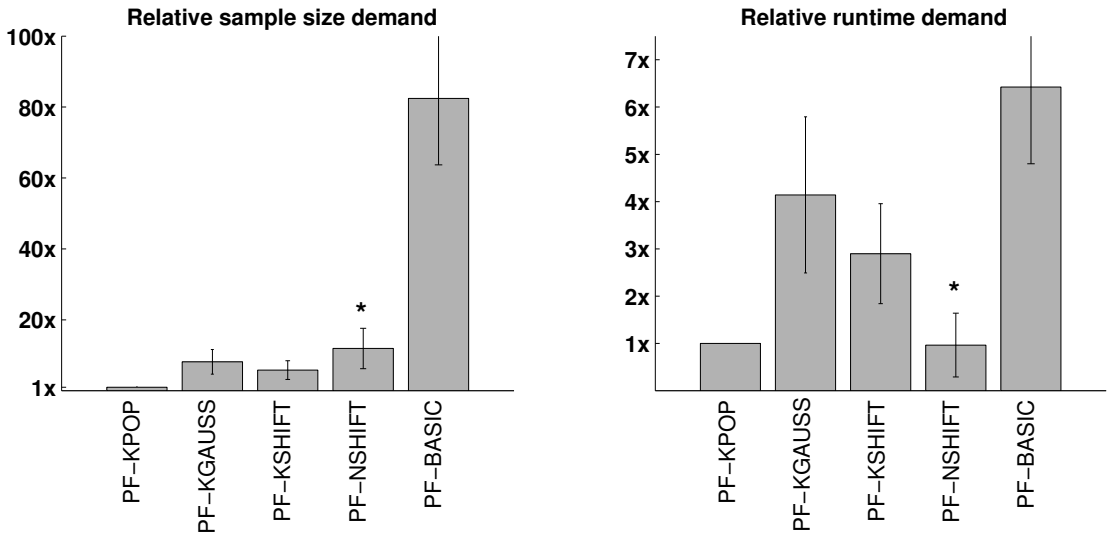


Figure 3.12: The relative number of particles (sample size) and runtime needed to match the accuracy of PF-KPOP with 1000 particles. Sample sizes and runtimes are shown relative to that of PF-KPOP, which is normalized to 1 to serve as reference. (* For PF-NSHIFT, the cases where convergence did not occur are not included.)

3.5 Summary

In this chapter we proposed improvements on previously used particle filters for parameter inference in ODE based pathway models. The main issue with particle filters is that the set of particles can collapse into a few distinct samples resulting in inaccurate estimates. We proposed three variants of kernel-enhancement, in which, at every time point, a Markov transition kernel is applied on each particle. This resolves the problem of particle collapse while still guaranteeing that the true parameter posterior is preserved. The kernel steps consist of a proposal, in which new particles are proposed in random (but usually guided) directions. Then the ratio of posterior probabilities of the old and new particles are compared to accept or reject each proposal. The acceptance or rejection is based on a formula that guarantees new samples to be distributed according to the posterior. Hence diversification is achieved while preserving the target distribution of the filter.

A small synthetic case study demonstrated that contrary to other particle filters, the kernel-enhanced methods can robustly recover from a collapsed particle state. Our case study on the JAK-STAT pathway has shown that predictions with the kernel-enhanced particle filters result in significantly reduced mean squared error compared to the basic particle filter, and particle filters relying on random noise injection. In particular, our

case study showed that the kernel-enhanced particle filter relying on proposals from the particle population statistics (PF-KPOP) was around 60 to 100 times more sample-size efficient than the basic particle filter. Even when factoring in the additional runtime cost to perform kernel-steps, the kernel-enhanced method achieved the same accuracy around 4 to 8 times faster than the basic filter.

Representing the parameter posterior in terms of particles opens up several interesting possibilities. For instance, it is possible to update the particles with new measurement data, simply by running the filter up to the time point of the new measurement and updating particle weights based on the corresponding likelihood value. This method can also be used in experimental design since it allows us to test how the parameter landscape (and, as a consequence, the uncertainty in predictions) changes when new data points are introduced. Finally, particles could be used to select between multiple alternative models based on Bayesian scores. Some of these tasks have been proposed in the context of Bayesian inference [117, 118, 102, 9] but using methods other than particle filters. Using kernel-enhanced particle filters could make these tasks more realistic in practice, since, as our case studies have shown, they give an accurate representation of the parameter posterior with limited sample size.

Chapter 4

Verification of pathway dynamics under Bayesian uncertainty

4.1 Introduction

Analysis and verification is crucial for building reliable models of biological pathways. Verifying a pathway model involves making statements about the intended behavior of the pathway expressed in temporal logic. Model checking algorithms can then be used to verify whether the model satisfies the specified properties [88]. Verifying properties of a pathway model is especially challenging when the model has a component of uncertainty. In this case manually examining simulation traces will not reveal properties of the system. Statistically valid techniques are needed to formulate properties and check whether they are satisfied with sufficiently high probability.

In this chapter we propose a novel method to verify properties of a model under parameter uncertainty. We focus on ordinary differential equations (ODE) based models whose parameters are not exactly known and cannot be directly measured. Their value can only be partially inferred from noisy and limited experimental data. As also argued in Chapter 3, treating model parameters as random variables is reasonable in this context. Using a Bayesian approach one can incorporate previous knowledge about the parameter values through a prior distribution. The prior is then updated using the likelihood of the observations to obtain a posterior distribution over the parameters. The main contribution of this chapter is providing a framework for verifying properties of a model characterized by such a Bayesian posterior distribution.

The method proposed here provides an important link between two areas of systems biology research. There is a growing body of work on Bayesian inference in systems biology [59], meanwhile, there is strong interest in the formal verification of pathway models [88]. However, the interface of the two areas has been largely unexplored. The method proposed here is important in this context since it enables the verification of an uncertain model with its possible realizations (corresponding to different parameter values) weighted according to their support from prior knowledge and experimental data.

The approximate probabilistic verification of dynamical systems usually involves simulating independent realizations of the system. However, under Bayesian parameter uncertainty, sampling independently according to the posterior distribution is usually not possible. Previously proposed statistical model checking methods [75, 74] require that the samples are independent and are therefore not applicable in this context. Our method provides a solution to perform statistical model checking when using *dependent* samples from the parameter posterior distribution of an ODE model.

We propose a Markov chain Monte Carlo (MCMC) based statistical model checking framework. The MCMC scheme produces a sequence of dependent random realizations of the model dynamics over the parameter posterior. Using this sequence of samples, we construct hypothesis tests to decide whether the model satisfies a temporal logic property with at least a given probability. Two different hypothesis tests are introduced, with one using an initially chosen fix sample size, and the other one making the decision in a sequential and adaptive manner, based on the result of previous samples. Using recent results from the theory of general state space Markov chains, we prove sample size bounds for both hypothesis tests.

There is a strong connection between this chapter and Chapter 3 in that both address parameter uncertainty in ODE based pathway models. In Chapter 3 we used particle filter algorithms to obtain samples from the parameter posterior. Here we propose a method to do model verification based on the posterior. In principle, one could use a particle filter to obtain a representation of the parameter posterior and then use the particles to perform model verification. Here, instead, we build on an MCMC methodology for several reasons. First, in the context of formal verification, an approximate answer is only acceptable if it is based on strong statistical guarantees. We are able to derive such statistical error bounds for MCMC estimates, but similar error bounds for

finite sample size particle filter estimates have not yet been established. Second, here we directly exploit the fact that MCMC collects samples in a sequence, and our sequential hypothesis test can adaptively stop the sampling procedure once enough samples have been collected. In contrast, particle filter sample sizes would need to be fixed in advance, and the filter would need to be run up to the maximal time point before verification could begin. For these reasons we build on an MCMC methodology, but view other forms of Bayesian inference to perform model verification under parameter uncertainty as an interesting direction for future research.

First, we apply our method on an ODE model of the JAK-STAT biochemical pathway [119]. The empirical results show that the proposed method enables verification with respect to the Bayesian parameter posterior with reliable error bounds. We also find that sequential testing is often significantly more efficient than the fix sample size test. Next, we use our method on a model of the extrinsically triggered apoptosis pathway (EARM1.3) [120], a large model containing 69 dynamical species and 71 unknown parameters. We find that some important qualitative properties are preserved, while others cannot be verified under substantial parameter uncertainty arising from the limited nature of experimental data. In both case studies we use wet-lab experimental data available from the literature, demonstrating the practical applicability of our method.

The rest of this chapter is organized as follows. In the next section, we review some of the basic concepts behind our method and discuss previous work. In Section 4.3 we introduce our method in detail. This includes defining the temporal logic for expressing dynamical properties and our main algorithms for performing statistical model checking using MCMC. Section 4.4 provides theoretical error bounds and sample size guarantees of our algorithms, and compares the efficiency of the hypothesis tests. Section 4.5 addresses some of the important practical aspects of using this method. In Section 4.6 we apply the proposed method to perform verification on two pathway models. We conclude with a summary of the main contributions and results.

4.2 Background and previous work

Properties about dynamical systems can be formally expressed as formulas in temporal logic. Using temporal logic, one can conveniently describe both qualitative and quan-

titative dynamical properties of interest. The technique of *model checking* [70] is used to automatically verify if a model satisfies these properties. Model checking has been used for the analysis of dynamical embedded systems [121] and hybrid systems [122], and is of significant interest in systems biology [123, 77, 124]. Both temporal logic and model checking techniques have also been extended to analyze dynamical systems with a component of stochasticity. In this context one verifies if a property is satisfied with a certain probability.

When the state space of a stochastic model can be explicitly enumerated (for instance in the case of discrete state space Markov chain models), the reachable states can be traversed to find the probability of a dynamical property being satisfied [72]. However, due to state space explosion this is intractable for large pathway models. It will also be impossible in the case of non-linear ODE models where an explicit reconstruction of the state transitions is not possible. In this case a statistical model checking approach can be used to perform implicit verification based on repeated simulation of the model dynamics.

Statistical model checking (see also Section 2.4) aims to verify whether a dynamical system $\mathcal{S}$ satisfies a temporal logic property ψ with probability at least r, or more formally, whether $\mathcal{S} \models \mathbb{P}_{\geq r}(\psi)$. For a single realization (also called a *trajectory*) of the system, ψ is either satisfied or not. By imposing a probability measure over the set of trajectories, one can define the probability of satisfaction of ψ, denoted P_ψ. This probability is compared to a threshold r, and the verification problem is generally posed as a hypothesis test between $H_0 : P_\psi \geq r + \delta$ and $H_1 : P_\psi \leq r - \delta$, with δ being a chosen indifference region [75]. The hypothesis test is solved using statistical approximations based on repeated simulation of the system [74]. Importantly, statistical model checking assumes that each sampled realization of the system is statistically independent.

As we have discussed in Chapters 2 and 3, the dynamics of ODEs is governed by a set of kinetic parameters whose value is often not known and not directly observable. In a Bayesian framework, the uncertainty in the parameter values is represented by a probability distribution. Following Chapters 2 and 3, we refer to the probability distribution of parameters conditioned on a set of observations as the *posterior* distribution. As the parameter values determine the system dynamics, the parameter distribution also induces a probability measure over the possible realizations of the system dynam-

ics. However, obtaining *independent* samples from a posterior distribution is impossible in all but very special cases. Verification relying on independent samples cannot be used in this setting, and we are not aware of any previous work that has addressed this limitation.

Sampling independent system trajectories according to a prior distribution is usually straightforward. In [81], we showed that when there is a uniform distribution on the model parameters, one can use random sampling and a standard statistical model checking scheme to verify if a property is satisfied with a given probability. This was used with the motivation of modeling small, local variability in rate constants among individual cells. A more complete characterization of a pathway model's parameter space with respect to logic properties was addressed in [73], where piecewise-multiaffine differential equations are considered, for which it is possible to identify regions of parameter space where the model satisfies a temporal logic property. The above approaches deal with parameter uncertainty as a *prior* property of the system, rather than its uncertainty conditioned on data, which we address here. A recent work [125] considers the problem of parameter learning in CTMC models. The goal is to find parameters such that the probability of satisfaction of a set of temporal logic properties becomes most likely. Even though parameters are treated in a Bayesian framework, there are important differences from our setting, namely in [125] (i) the properties themselves are used as "data" with respect to which parameters are tuned and (ii) Bayesian inference is not performed, instead, a global optimization procedure is used to find the single maximum likelihood or maximum aposteriori-probability parameter estimate.

In our approach we use a Markov chain Monte Carlo (MCMC) method to collect a sequence of samples from the space of possible parameters. The chain is designed to ensure that the samples are distributed according to the parameter posterior. MCMC is a general strategy to obtain samples from distributions that are hard to sample from [126]. These samples can then be used to analyze properties of interest. The MCMC method has been used in systems biology to explore the parameter posterior of ODE models [127, 128, 16]. These works, however, have not addressed the problem of probabilistic verification. Further, these works do not use error estimates for the results obtained using MCMC. Claims for the correctness of the approach rely on methods such as the Gelman-Rubin diagnostic [129, 130] to qualitatively assess convergence, rather than pro-

vide quantitative confidence intervals. In contrast, here we build on recent results in statistical theory to explicitly bound the error in our MCMC estimates. In fact, the efficiency and accuracy of our statistical model checking algorithms crucially depend on MCMC error bounds for finite sample sizes.

4.3 Statistical model checking under Bayesian uncertainty

In this section we develop the methodology for performing statistical model checking under Bayesian uncertainty. We first define our temporal logic to express dynamical properties. Then we show how hypotheses can be posed to express whether the model satisfies a property with at least a given probability. Then a Markov chain Monte Carlo method is presented to collect samples from the parameter posterior, and our two main algorithms, a fix sample size test and a sequential test is proposed to decide between the hypotheses.

4.3.1 ODE models with Bayesian parameter uncertainty

An ODE model describes the time-derivative of a set of variables $\mathbf{x}(t) \in \mathbb{R}^{d_x}$ through a system of (possibly non-linear) equations (see also Section 2.2.1). The equations are stated as

$$\dot{\mathbf{x}}(t) = F(\mathbf{x}(t), \theta).$$

Here $\theta \in \mathbb{R}^{d_\theta}$ is a vector of model parameters. To simulate the model, initial conditions $\mathbf{x}(0)$ need to be set, and throughout the rest of the chapter we assume that these are given. However, if this is not the case, initial conditions could also be treated as part of the set of unknown model parameters.

We constrain model parameters (whose values are not exactly known) to be in a bounded set $\Theta \subset \mathbb{R}^{d_\theta}$, and for simplicity define this set as the hypercube arising by constraining parameter θ_i to the interval $[a_i, b_i]$, where $a_i < b_i \in \mathbb{R}$, $1 \leq i \leq d_\theta$. The set of possible parameter values will thus be $\Theta = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_{d_\theta}, b_{d_\theta}]$.

Importantly, we assume that a prior probability density $p_0(\theta)$ is given over Θ. One can use the prior to encode existing knowledge about the joint distribution of parameters.

In the simplest case, $p_0(\theta)$ will be uniform over Θ, defined as

$$p_0(\theta) = \begin{cases} c & \text{if } \theta \in \Theta \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Here $c = \left(\prod_{i=1}^{d_\theta}(b_i - a_i)\right)^{-1}$ is a constant ensuring that p_0 integrates to 1.

When more is known about the value of the parameters, a Gaussian or a log-normal distribution is used as a prior. For instance, an uncorrelated Gaussian prior would have the form $p_0(\theta) = \prod_{i=1}^{d_\theta} \mathcal{N}(\mu_{\theta_i}, \sigma^2_{\theta_i})$. In principle, truncation may be needed in these cases to integrate to 1 over the bounded Θ, but in practice the tail probabilities will decay fast enough that this effect is negligible.

Now assume that we have a set of observations Y, and recall from Section 2.3 that $Y_{i,j}$ denotes the measured value for species i at time point t_j. We will use the Gaussian likelihood to assess how well a particular set of parameters explains the experimental data. The form of the likelihood is

$$p(Y|\theta) = C \prod_{i=1}^{n} \prod_{j=1}^{m} \exp\left(\frac{(Y_{i,j} - x_i(t_j)|_\theta)^2}{\sqrt{2}\sigma^2_{i,j}}\right), \tag{4.2}$$

where C is a constant ensuring a unit integral, $x_i(t_j)|_\theta$ is the value of species i at time t_j when using parameters θ, and $\sigma_{i,j}$ are the standard deviations of each data point. In case not all variables are observed or if the state is observed through a function g of the state, as $\mathbf{y}(t_j) = g(\mathbf{x}(t_j))$, the likelihood can be modified in the obvious way.

Using the prior and the likelihood, we can express the posterior distribution of parameters as

$$\pi(\theta|Y) = \frac{p(Y|\theta)p_0(\theta)}{p(Y)} = \frac{p(Y|\theta)p_0(\theta)}{\int p(Y|\theta)p_0(\theta)d\theta}. \tag{4.3}$$

The parameter posterior will introduce a component of uncertainty in the ODE model. Intuitively, it will "weight" realizations of the model based on their support from prior knowledge and observations. In the following sections, we provide methods to verify whether properties expressed in probabilistic temporal logic are met with a given probability, with respect to the posterior.

4.3.2 Probabilistic properties and verification

In this section we first define a temporal logic on trajectories of the ODE model. The logic will be able to express dynamical properties on the system variables such as "the concentration of STAT starts in the interval [0,0.1], after which it reaches the interval [1,2] and stays in the interval [1,2]". We then extend the logic to a probabilistic setting where there is a probability distribution over the set of trajectories. Finally, we show how deciding whether a property holds with at least a given probability can be posed as a hypothesis test.

Properties on a single trajectory

To specify the dynamical properties of a single realization of the system, we first encode them as formulas in a specification logic. We assume that analyzing the dynamics of the system is of interest only up to a maximal time point T. Accordingly, we use a bounded version of linear time temporal logic (BLTL)[70]. The formulas in this logic are interpreted at a finite set of time points $\mathcal{T} = \{0, 1, \ldots, T\}$ corresponding to all the relevant time points of interest. With a slight abuse of notation, we will directly use these integer time indices to specify the state at the corresponding time point. For instance $\mathbf{x}(1)$ will denote the state at time point 1.

In our setting a trajectory is represented by ς_θ, which (given fix initial conditions) is fully defined by the choice of parameters θ since the ODE system is deterministic. A trajectory will be defined by the set of states $\varsigma_\theta = (\mathbf{x}(0)|_\theta, \mathbf{x}(1)|_\theta, \ldots, \mathbf{x}(T)|_\theta)$, where $\mathbf{x}(i)|_\theta$ is the value of system variables at time point i when the corresponding ODEs are simulated with the parameter set θ. $\varsigma_\theta(t) = \mathbf{x}(t)|_\theta$ for $t \in \mathcal{T}$. The transitions from $\mathbf{x}(i)|_\theta$ to $\mathbf{x}(i+1)|_\theta$ is ensured by the fact that once we fix the parameters values, the systems of ODEs has a unique solution and is characterized by a continuous function (for more details, see [81]).

Atomic propositions in BLTL will be of the form (i, L, U) with $L \leq U$. This is interpreted as "the value of x_i falls in the interval $[L, U]$". When describing the case study, for easier readability, we will use the $[L \leq x_i \leq U]$ notation with the same intended meaning.

The syntax of formulas in BLTL are defined as follows:

- Every atomic proposition is a BLTL formula.
- The constants *true* and *false* are BLTL formulas.
- If ψ and ψ' are BLTL formulas then $\neg\psi$ and $\psi \vee \psi'$ are also BLTL formulas.
- If ψ is a BLTL formula then $\mathbf{G}^{\leq t}\psi$ and $\mathbf{F}^{\leq t}\psi$ are also BLTL formulas, where $t \leq T$ is a positive integer.
- If ψ and ψ' are BLTL formulas then $\psi \mathbf{U}^{\leq t}\psi'$ is a BLTL formula, where $t \leq T$ is a positive integer.

Logic operators such as $\wedge$, $\vee$ and $\Rightarrow$ are defined in the standard way and allow the construction of complex combinations of statements.

Properties of the system dynamics defined by the ODEs can be efficiently expressed using BLTL. The semantics of BLTL will be defined by $\varsigma_\theta, t \models \psi$ (expressing that the trajectory ς_θ satisfies the property ψ are time t), as follows.

- $\varsigma_\theta, t \models (i, L, U)$ iff $L \leq \varsigma_{\theta,i}(t) \leq U$ where $\varsigma_{\theta,i}(t)$ is the ith component of $\varsigma_\theta(t)$.
- $\varsigma_\theta, t \models \psi \vee \psi'$ iff $\varsigma_\theta, t \models \psi$ or $\varsigma_\theta, t \models \psi'$.
- $\varsigma_\theta, t \models \neg\psi$ iff $\varsigma_\theta, t \not\models \psi$.
- $\varsigma_\theta, t \models \psi \mathbf{U}^{\leq k}\psi'$ iff there exists k' such that $k' \leq k$, $t + k' \leq T$, $\varsigma, t + k' \models \psi'$ and $\varsigma_\theta, t + k'' \models \psi$ for every $0 \leq k'' < k'$.

Finally, we say that $\varsigma_\theta \models \psi$ (that is, the trajectory ς_θ satisfies ψ) iff $\varsigma_\theta, 0 \models \psi$. The derived temporal operators $\mathbf{G}^{\leq t}$ and $\mathbf{F}^{\leq t}$ are defined as follows.

$$\mathbf{F}^{\leq t}\psi \equiv true\mathbf{U}^{\leq t}\psi,$$
$$\mathbf{G}^{\leq t}\psi \equiv \neg\mathbf{F}^{\leq t}\neg\psi.$$

This provides the basis for verifying a single trajectory of the ODE system. We next introduce a probabilistic extension of BLTL which applies to sets of trajectories.

Statistical model checking as a hypothesis test

Under assumptions of continuity and measurability on the ODE equations, we can assign a probability to the trajectories satisfying a given formula ψ with respect to the distribution of parameters (for a detailed discussion, see [81]). We can define the probability

of the system satisfying a formula ψ with respect to the parameter posterior $\pi(\theta|Y)$ as

$$P_\psi = \int_\Theta \pi(\theta|Y) I(\varsigma_\theta \models \psi) d\theta, \tag{4.4}$$

where I is the indicator function taking value 1 if $\varsigma_\theta \models \psi$, and 0 otherwise.

To express properties of this nature, we will encode them in a formalism called PBLTL[131], which is a probabilistic extension of BLTL. Formulas in PBLTL are of the form $\mathbb{P}_{\geq r}(\psi)$ (or $\mathbb{P}_{\leq r}(\psi)$), where ψ is a BLTL formula and r is a real number in $(0, 1)$. The PBLTL formula $\mathbb{P}_{\geq r}(\psi)$ expresses that we want to verify whether the probability measure of the trajectories satisfying ψ (or P_ψ) is at least r.

Deciding whether $\mathbb{P}_{\geq r}(\psi)$, can be posed as a hypothesis test [132, 75, 74]. Our goal is to decide between the following two hypotheses.

$$\begin{aligned} H_0 : &\quad P_\psi \geq r + \delta, \\ H_1 : &\quad P_\psi \leq r - \delta, \end{aligned} \tag{4.5}$$

where $\mathbb{P}_{\geq r}(\psi)$ is a PBLTL formula and δ is a parameter defining an indifference region. We refer to an algorithm that chooses between the two hypotheses as a *hypothesis test*. The hypothesis test makes an error, if it chooses H_0 when H_1 holds (Type-I error), or if it chooses H_1 when H_0 holds (Type-II error). If neither hypothesis holds (that is, $P_\psi \in (r - \delta, r + \delta)$), either decision is considered correct. For simplicity, we will just consider the overall error rate (denoted by ϵ) covering either error type. It is important that the desired error rate ϵ, along with δ and r can be chosen by the user.

Due to the difficulty of sampling independently from the posterior, we will use a Markov chain Monte Carlo method, which produces a sequence of dependent samples from the posterior, to decide between the hypotheses.

4.3.3 MCMC for statistical model checking

Our goal in this section is to design a Markov chain in a way that its stationary distribution matches the parameter posterior $\pi(\theta|Y)$. Then, if the chain is stationary, it will provide a sequence of samples from the posterior. The chain starts at an initial parameter sampled from the *prior*. In each subsequent step of the chain, one first uses a *proposal distribution* to pick the next candidate parameter, and then applies the *acceptance ratio* to accept or reject the proposed candidate. At each step of the Markov

chain, the trajectory corresponding to the current parameter vector is verified, and these samples are used to decide whether $\mathbb{P}_{\geq r}\psi$.

There are many possible ways to construct an adequate proposal distribution and acceptance rate. The only requirement is that the resulting Markov chain have a unique stationary distribution, and that the stationary distribution is identical to the posterior $\pi(\theta|Y)$. A sufficient condition for the existence of a stationary distribution is *detailed balance*, which states that the probability of being in state θ^n and move to the state θ^m is equal to the probability of moving from θ^m to θ^n, or more formally,

$$\pi(\theta^n|Y)P(\theta^n \to \theta^m) = \pi(\theta^m|Y)P(\theta^m \to \theta^n). \tag{4.6}$$

Here P is the overall probability of the transition, which is a combination of the proposal and the acceptance rate. The uniqueness of the stationary distribution holds if the Markov chain is *ergodic*. A simple sufficient condition of ergodicity is that there exists a finite integer k such that any state can be reached from any other state in exactly k steps.

It is important to point out that our statistical model checking scheme, and theoretical analysis in Section 4.4 will hold for *any* Markov chain that fulfills the above properties. The design of an efficient Markov chain is often problem-specific and is an orthogonal problem to our main contribution (for more details on constructing efficient chains, we refer to [133]). Here we introduce a standard MCMC scheme based on the Metropolis-Hastings algorithm [114].

We denote the proposal by $q(\theta^n \to \theta)$, which represents the probability of proposing θ if the current parameter value is θ^n. A good choice for the proposal is $q(\theta^n \to \theta) = \mathcal{N}(\theta^n, \Sigma_{\text{MH}})$, a d_θ-dimensional multivariate Gaussian with mean identical to the current parameter vector, and covariance matrix Σ_{MH}. Here Σ_{MH} can be diagonal with entries representing variances along each dimension independently.

$$\Sigma_{\text{MH}} = \begin{pmatrix} \sigma^2_{\text{MH},1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma^2_{\text{MH},\text{d}_\theta} \end{pmatrix}$$

In practice it is important to choose the entries of the covariance matrix carefully, since it affects the mixing properties of the chain. In the case of pathway models,

prior knowledge about individual parameters, or initial test runs will provide sufficient information to choose the diagonal entries.

The acceptance rate needs to be chosen in a way that the detailed balance is kept with respect to the posterior. The acceptance rate follows from the Metropolis-Hasting scheme, where the candidate is accepted with probability α, determined by the proposal and the posterior as follows.

$$\alpha = \min\left(1, \frac{q(\theta' \to \theta^n)}{q(\theta^n \to \theta')} \frac{\pi(\theta'|Y)}{\pi(\theta^n|Y)}\right) = \min\left(1, \frac{q(\theta' \to \theta^n)}{q(\theta^n \to \theta')} \frac{p_0(\theta')p(Y|\theta')}{p_0(\theta^n)p(Y|\theta^n)}\right). \tag{4.7}$$

Note that the normalization constant $p(Y)$ appearing in the posterior is eliminated, and one only needs to evaluate the prior and the likelihood at the original and at the proposed parameter value. In our case the proposal q is symmetric, and therefore equation (4.7) simplifies to

$$\alpha = \min\left(1, \frac{p_0(\theta')p(Y|\theta')}{p_0(\theta^n)p(Y|\theta^n)}\right). \tag{4.8}$$

This choice of acceptance rate will ensure detailed balance, and imply reversibility. It also is easy to see that the resulting chain will be ergodic. This simply follows from the fact that the proposal Gaussian will propose any parameter in Θ with nonzero probability, and the acceptance rate is also always positive in Θ. Therefore any state can be reached from any other state within *one* step, implying ergodicity.

The above conditions ensure that the Markov chain will *converge* to the true posterior. To ensure that the chain has sufficiently converged, we take an initial t_0 number of steps in the Markov chain, called the *burn-in time*. The estimation done with respect to the samples (in our case the verification task) is only started after the burn-in phase.

We now introduce the function getMCMCsample, which takes as input the current parameter values, takes a single step in the Markov chain, and returns the new parameter values.

Function getMCMCsample

Input: parameter vector θ_{in}. Output: parameter vector θ_{out}

1: Sample a new parameter vector based on proposal: $\theta' \sim q(\theta_{\text{in}} \to \theta)$
2: Calculate acceptance ratio $\alpha = \min\left(1, \frac{p_0(\theta')p(Y|\theta')q(\theta' \to \theta_{\text{in}})}{p_0(\theta_{\text{in}})p(Y|\theta_{\text{in}})q(\theta_{\text{in}} \to \theta')}\right)$
3: Generate $\eta \sim \text{Uniform}[0, 1]$
4: **if** $\eta < \alpha$ **then**
5: **return** $\theta_{\text{out}} := \theta'$
6: **else**
7: **return** $\theta_{\text{out}} := \theta_{\text{in}}$
8: **end if**

We are now ready to present two tests between the hypotheses in (4.5). These tests use getMCMCsample as a subroutine. The main idea is that at each step of the Markov chain, the trajectory corresponding to the current parameter value is simulated and verified with respect to a given property ψ. Once a sufficient number of samples have been verified, we can stop the Markov chain and decide between hypotheses H_0 and H_1.

4.3.4 Fix sample size hypothesis test

A fix sample size hypothesis test is shown in Algorithm 1. This test assumes that we have fixed N, the total number of samples to collect, and thus a choice of either H_0 or H_1 is returned after exactly N steps. The algorithm first takes t_0 burn-in steps to ensure convergence to the posterior. Then in each of N steps, a property ψ is verified. If the number of times it is verified to be true is at least rN then H_0, otherwise, H_1 is chosen. With parameters δ and r, and chosen error rate ϵ, N should be set to

$$N := \left\lceil \frac{\log(1/\epsilon)}{\gamma\delta^2} \right\rceil, \tag{4.9}$$

where γ is the spectral gap of the Markov chain, and is discussed in detail in Section 4.5.1. We will prove in Section 4.4 that this choice of N achieves an error rate of ϵ in deciding the hypothesis test.

Algorithm 1 Fixed sample size hypothesis test

Input: BLTL property ψ, threshold probability r, observations Y, number of samples N, number of burn-in steps t_0, prior p_0, proposal q.
Output: Choice of H_0 or H_1.

```
Sample initial parameter vector from the prior ϑ^0 ~ p_0(θ)
for i := 1 ... t_0 do
    ϑ^i := getMCMCsample(ϑ^{i-1})
end for
Set S := 0 and θ^0 := ϑ^{t_0}
for n := 1 ... N do
    θ^n := getMCMCsample(θ^{n-1})
    Simulate the trajectory ς_{θ^n}
    if ς_{θ^n} ⊨ ψ then
        S := S + 1
    end if
end for
if S ≥ Nr then
    return H_0
else
    return H_1
end if
```

4.3.5 Sequential hypothesis test

A sequential sample size hypothesis test is shown in Algorithm 2. This test uses sequential hypothesis testing to adaptively set the number of steps before stopping (based on the result of verification on samples gathered so far). The stopping condition is governed by a threshold M. At each step, a property ψ is verified. The number of times up to step number n when ψ was verified to be true is counted by S. If S crosses the upper threshold $nr + M$ then H_0 is chosen, if it crosses the lower threshold $nr - M$ then H_1 is chosen, otherwise we continue taking samples.

Figure 4.1 shows the lower and upper stopping conditions and an example trace of S crossing the upper threshold.

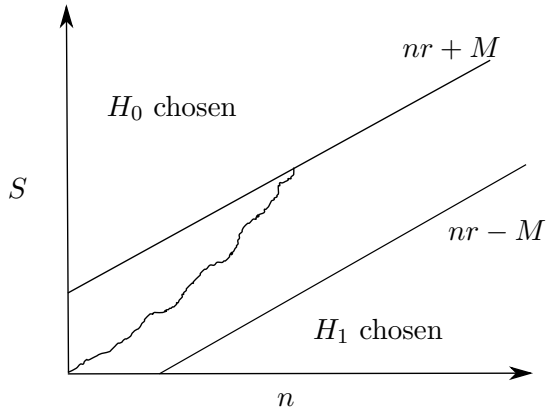


Figure 4.1: Sequential hypothesis test with an example running sum crossing the upper stopping condition.

It is important to note that the sequential hypothesis test will always stop within a finite number of iterations with probability 1, as shown in Section 4.4.

4.4 Theoretical analysis

In this section we present bounds on the error of the hypothesis tests introduced in the previous section. In what follows, we state results in more general terms. Namely, results hold for any reversible Markov chain on state space $\Omega \subset \mathbb{R}^d$ with a unique stationary distribution π. The Markov chain described in the previous section is an instance of such a chain. In our dynamical systems verification context, Ω will correspond to $\Theta \subset \mathbb{R}^{d_\theta}$, the set of possible model parameters, and π corresponds to the posterior $\pi(\theta|Y)$. For the general discussion, we assume that a square-integrable function f (in other words $f \in L^2(\pi)$), with $0 \leq f(x) \leq 1$ for $x \in \Omega$ is evaluated at each step of the Markov

Algorithm 2 Sequential hypothesis test

Input: BLTL property ψ, threshold probability r, observations Y, stopping condition M, number of burn-in steps t_0, prior p_0, proposal q.
Output: Choice of H_0 or H_1.

```
Sample initial parameter vector from the prior ϑ^0 ~ p_0(θ)
for i := 1 ... t_0 do
    ϑ^i := getMCMCsample(ϑ^{i-1})
end for
Set n := 1, S := 0 and θ^0 := ϑ^{t_0}
loop
    θ^n := getMCMCsample(θ^{n-1}).
    Simulate the trajectory ς_{θ^n}
    if ς_{θ^n} ⊨ ψ then
        S := S + 1
    end if
    if S ≥ nr + M then
        return H_0
    else if S ≤ nr − M then
        return H_1
    else
        Set n := n + 1 and continue
    end if
end loop
```

chain. The function f corresponds to evaluating $I(\varsigma_\theta \models \psi)$, taking 1 or 0, depending on whether the property of interest is satisfied with the current parameters. Consequently, the expectation $\mathbb{E}_\pi f$ will correspond to P_ψ in our setting.

4.4.1 Concentration of the Markov chain estimate

In this section we show a Hoeffding-type concentration inequality for estimates made using dependent samples obtained from a Markov chain. It provides an upper bound on the probability that the sum of $f(X_i)$ deviates from its expected value. This will be the basis of the stopping conditions for our hypothesis tests. This result was proven for Markov chains on finite state spaces in [134]. The result has only recently been extended to general state spaces in [135]. This form of the result follows from [134] and [135] with rescaling to exploit the fact that the values of f are bounded between 0 nd 1.

Theorem 4.1 (Hoeffding inequality for reversible Markov chains)**.** *Let $X_1, \ldots, X_n$ be a stationary reversible Markov chain with spectral gap γ, and unique stationary distribution π. Let $f \in L^2(\pi)$ such that $0 \leq f(x) \leq 1$ for every $x \in \Omega$. Let $S_n := \sum_{i=1}^n f(X_i)$, then for any $t \geq 0$,*

$$P(|S_n - n \cdot \mathbb{E}_\pi f| \geq t) \leq 2 \exp\left(-\frac{t^2 \cdot \gamma}{n}\right). \tag{4.10}$$

Note that the requirement that the chain be *stationary* is ensured by choosing a sufficiently long t_0 initial burn-in time.

4.4.2 Sample sizes and error bounds for the tests

Suppose that $X_1, X_2, \ldots$ is a reversible Markov chain on $\Omega \subset \mathbb{R}^d$, with unique stationary distribution π, and $f : \Omega \to [0,1]$ is a bounded function. Our objective is to do a test between the following two hypotheses, given $r \in (0,1)$ and $\delta \in (0, \min(r, 1-r))$, (this is a more general form of (4.5)).

$H_0 : \mathbb{E}_\pi f \geq r + \delta,$

$H_1 : \mathbb{E}_\pi f \leq r - \delta.$

In the following sections, we discuss two tests to choose between these hypotheses. The first one is a fixed sample size test (a generalized form of Algorithm 1), while the second one is a sequential test (a generalized form of Algorithm 2).

Fixed length hypothesis test

Suppose that we have a sample of length n consisting of the values $f(X_1), \ldots, f(X_n)$. Let $S_n := f(X_1) + \ldots + f(X_n)$. We will use the following hypothesis test.

- If $S_n \geq nr$, accept hypothesis H_0,
- otherwise accept hypothesis H_1.

The next theorem bounds the Type-I and Type-II errors of the test, which correspond to accepting hypothesis H_1 when in fact hypothesis H_0 holds, and vice-versa.

Theorem 4.2 (Error bound for fixed length hypothesis test)**.** *For the fixed length hypothesis test, both the Type-I and Type-II errors are bounded by*

$$\exp\left(-\gamma\delta^2 n\right). \tag{4.11}$$

Proof. Suppose that H_1 holds, implying that $\mathbb{E}_\pi f \leq r - \delta$. A false decision is made (H_0 is chosen) if $S_n \geq nr$. From here

$$S_n - n\mathbb{E}_\pi f \geq nr - (nr + n\delta) = n\delta$$

Applying the Hoeffding inequality (Theorem 4.1), we get

$$P(S_n - n\mathbb{E}_\pi f \geq n\delta) \leq \exp\left(-\gamma\delta^2 n\right).$$

The same holds under the opposite hypothesis. □

This result is directly applicable to Algorithm 1, and implies that in order for the probability of a false decision to be smaller than ϵ, the sample size N should be chosen as

$$N \geq \frac{\log(1/\epsilon)}{\gamma\delta^2}. \tag{4.12}$$

Sequential hypothesis test

We now turn to establishing a stopping condition for the sequential hypothesis test. The sequential test is as follows. We fix a threshold $M > 0$. Denote $S_n := f(X_1) + \ldots + f(X_n)$ as before. First, set $n = 1$, then

- if $S_n \geq nr + M$, accept hypothesis H_0,
- if $S_n \leq nr - M$, then accept hypothesis H_1,
- otherwise, set $n := n + 1$ and repeat.

In contrast to the fix length hypothesis test where n was chosen in advance, here, n is variable and M is the parameter chosen in advance to guarantee an error rate of ϵ. The following proposition bounds the errors of the test with a particular choice of M.

Theorem 4.3 (Error bound for sequential hypothesis test)**.** *For the sequential hypothesis test, both the Type-I and Type-II errors are bounded by*

$$\exp\left(-2\gamma\delta M\right) \cdot \exp\left(-\frac{M}{1-r} \cdot \gamma\delta^2\right) \cdot \frac{2}{\gamma\delta^2}. \tag{4.13}$$

Proof. Suppose that H_1 holds, and thus $\mathbb{E}_\pi f \leq r - \delta$. A false decision occurs (H_0 is chosen) when, for any $n \geq 1$, $S_n \geq nr + M$. From here $S_n - \mathbb{E}_\pi f \geq nr + M - (nr - n\delta) = M + n\delta$. Applying the Hoeffding inequality (Theorem 4.1), we get

$$\begin{aligned} P(S_n - \mathbb{E}_\pi f > n\delta + M) &\leq \exp\left(-\frac{\gamma(n\delta + M)^2}{n}\right) \\ &\leq \exp\left(-2\gamma\delta M\right) \cdot \exp\left(-n\gamma\delta^2\right). \end{aligned}$$

It is easy to see that the probability of a false decision is then bounded by the sum of these terms over all possible n. We also exploit the fact that since $S_n \leq n$, when $n \leq M/(1-r)$, it always holds that $S_n \leq nr + M$. Therefore the first $\lfloor M/(1-r) \rfloor$ terms of the sum will be 0.

$$\begin{aligned}\sum_{n=\lfloor M/(1-r)\rfloor}^{\infty} P(S_n - \mathbb{E}_\pi f > n\delta + M) &\leq \exp(-2\gamma\delta M) \sum_{n=\lfloor M/(1-r)\rfloor}^{\infty} \exp(-n\gamma\delta^2) \\ &\leq \exp(-2\gamma\delta M) \exp\left(-\frac{M}{1-r} \cdot \gamma\delta^2\right) \cdot \frac{1}{1-\exp\left(-\gamma\delta^2\right)} \\ &\leq \exp\left(-2\gamma\delta M\right) \cdot \exp\left(-\frac{M}{1-r} \cdot \gamma\delta^2\right) \cdot \frac{2}{\gamma\delta^2}.\end{aligned}$$

The same holds under the opposite hypothesis. □

This result is directly applicable to Algorithm 2, and implies that in order for the probability of a false decision to be smaller than ϵ, the parameter M should be chosen as

$$M = \frac{\log(2/(\epsilon\gamma\delta^2))}{2\gamma\delta + \gamma\delta^2/(1-r)}. \tag{4.14}$$

4.4.3 Efficiency of fix length and sequential tests

We have already shown that having chosen r, δ and ϵ, the fix sample size test will take $n = \frac{\log(1/\epsilon)}{\gamma\delta^2}$ steps. The number of steps before the sequential test stops (either H_0 or H_1 is chosen) is a random variable. We now examine the *expected* number of steps (we refer to this as the stopping time T) of the sequential test. In what follows, we will be able to show the following important properties

- With the same choice of r, δ and ϵ, the expected stopping time of the sequential test is around half that of the fix sample size test if either H_0 or H_1 holds, that is, $\mathbb{E}_\pi f \notin [r-\delta, r+\delta]$.
- If $\mathbb{E}_\pi f$ is "further" from the edges of the indifference region. the stopping time of the sequential test decreases even more.
- Even if neither hypothesis holds, that is, $\mathbb{E}_\pi f \in [r-\delta, r+\delta]$, the test will stop with probability 1, and "forcing" a decision after a $2M/\delta$ steps will have only marginal effect on error.

First we derive an upper bound for the expected stopping time of the sequential test.

Theorem 4.4. *For the sequential test, with M chosen as in* (4.14)*, the expected stopping time satisfies*

$$\mathbb{E}(T) \leq \frac{\log(1/\epsilon)}{2\gamma\delta^2} + O\left(\sqrt{\frac{\log(1/\epsilon)}{2\gamma\delta^2}}\right), \tag{4.15}$$

under both hypotheses.

Proof. Under hypothesis H_1, the probability of the event that $T \geq M/\delta + t$ is less than equal to the event that $S_n > nr - M$ at step $n := \lfloor M/\delta + t \rfloor$. Under H_1, $\mathbb{E}f \leq r - \delta$, thus.

$$\begin{aligned} P(S_n > nr - M) &= P(S_n > n\mathbb{E}_\pi f + n\delta - M) \\ &= P(S_n > n\mathbb{E}_\pi f + (t-1)\delta), \end{aligned}$$

Using the Hoeffding inequatiy we see that for any $t > 0$

$$P\left(T \geq M/\delta + t\right) \leq \exp\left(-\gamma\delta^2 \cdot \frac{(t-1)^2}{M/\delta + t}\right). \tag{4.16}$$

To arrive at the expected stopping time, we use the fact that for any real valued random variable, $\mathbb{E}(X) \leq \int_{t=0}^{\infty} P(X \geq t)$.

$$\begin{aligned} \mathbb{E}(T) \leq & \frac{M}{\delta} + \int_0^\infty P(T \geq M/\delta + t)dt \\ \leq & \frac{M}{\delta} + \int_0^\infty \exp\left(-\gamma\delta^2 \cdot \frac{(t-1)^2}{M/\delta + t}\right) \\ \leq & \frac{M}{\delta} + 2\sqrt{\frac{M + 2\delta}{\gamma\delta^3} + \frac{2}{\gamma\delta^2}} \;=\; \frac{M}{\delta} + O\left(\sqrt{\frac{M}{\delta}}\right) \end{aligned}$$

Where the second step comes from an upper bound for the exponential integral. We now substitute M as in (4.14) to get

$$\mathbb{E}(T) \leq \frac{\log(1/\epsilon)}{2\gamma\delta^2} + O\left(\sqrt{\frac{\log(1/\epsilon)}{2\gamma\delta^2}}\right), \tag{4.17}$$

The same holds under the opposite hypothesis. □

By comparing (4.12) with (4.15), we can see that the sequential test typically requires less than half of the samples of the fix length test.

Our bound in (4.15) is based on the worst case, when $\mathbb{E}_\pi f$ is exactly on the border of either hypothesis region. In practice $\mathbb{E}_\pi f$ will often be further from r, and $|\mathbb{E}_\pi f - r| > \delta$ will hold. It is easy to show that in this case the denominator of (4.15) can be changed from $2\gamma\delta^2$ to $2\gamma\delta(|r - \mathbb{E}_\pi f|)$, resulting in earlier expected stopping.

It is also clear that the test will stop in a finite amount of time almost surely, even if none of the two hypotheses is satisfied (that is, if $\mathbb{E}_\pi f \in [r-\delta, r+\delta]$). In practice, however, we do not know whether this is the case, and one may need to stop the run after a certain number of steps, depending on available resources. By (4.16), the probability that the chain runs for more than $2M/\delta$ steps is less than

$$\exp\left(-\gamma\delta^2 \cdot \frac{(2M/\delta - 1)^2}{2M/\delta}\right), \tag{4.18}$$

under both hypotheses, which is very small in practice. Therefore, if this happens, one can stop and choose H_0 if $S_{2M/\delta} \geq (2M/\delta)r$ and H_1 otherwise. This modification of the original test only changes the Type-I and Type-II errors at most by the amount (4.18).

4.5 Practical considerations

4.5.1 Estimating the speed of mixing

The sample size bounds assume that we know the *spectral gap* of the Markov chain, denoted γ. The spectral gap is a quantity describing the mixing properties of the Markov chain. Intuitively, a small value of γ corresponds to a slowly mixing chain, meaning that many steps have to be taken before samples become approximately independent. Conversely, a fast mixing chain (with large γ) will produce a sequence of samples that become close to independent within a few steps. As seen from the sample size bounds in Section 4.4, a slowly mixing chain will require a larger sample size to carry out hypothesis testing.

The spectral gap is a property defined on reversible Markov chains. For Markov chains on finite, discrete state spaces, the spectral gap is simply the difference between the two largest eigenvalues of the transition matrix. In the general state space case, which applies in our setting, the spectral gap is somewhat more difficult to define, and the precise definition is given in Appendix B. Here we focus on how to estimate γ in practice.

In practice, the spectral gap has to be estimated from the output of the Markov chain. In general the output of the Markov chain is a function $f : \Omega \to \mathbb{R}$, where Ω is the state space of the chain. In our case, f denotes the model checker verifying a single trajectory based on a chosen parameter, and therefore $f : \Theta \to \{0, 1\}$. The main issue

is that the spectral gap is an *inherent* property of the Markov chain, independent of f, but we can only monitor the output of f to estimate the spectral gap. If the output of the chain is computed for several functions f, then we recommend to compute $\hat{\gamma}$ for each of them and use the minimum of those estimates. Once a reliable estimate for γ has been obtained, the same value holds for any f used in later runs.

At each step $X_i \in \Omega$ of the chain we have access to $f(X_i)$, and we need to estimate γ from the sequence of values $f(X_1), \ldots, f(X_N)$. In [14], we have shown that a good estimator for γ can be derived by estimating two related values: (1) $V_f := Var_\pi f$, the variance of f with respect to π, and (2) σ^2, the asymptotic variance of the mean of f with respect to π as $N \to \infty$. More precisely, σ^2 is defined as

$$\sigma^2 = \lim_{N\to\infty} \frac{1}{N} Var_\pi \left[\sum_{i=1}^{N} f(X_i) \right]. \tag{4.19}$$

A simple and consistent estimator for V_f is through the sample variance, namely

$$\hat{V}_f := \frac{1}{N - t_0} \left(\sum_{i=t_0+1}^{N} f^2(X_i) \right) - \left(\frac{1}{N - t_0} \sum_{i=t_0+1}^{N} f(X_i) \right)^2. \tag{4.20}$$

Estimating σ^2 is more difficult, and several methods have been proposed in the literature [136]. The estimation typically relies on the lagged autocovariance of the output sequence $f(X_i)$. In [14] we have proposed such an estimator, and showed that it has good theoretical properties and is reliable in practice. Let $\hat{\rho}_k$ denote the empirical k-lagged autocovariance. Then the estimate for the asymptotic variance is

$$\hat{\sigma}^2 = \left(\hat{\rho}_0 + 2 \sum_{k=1}^{\eta} \hat{\rho}_i \right), \tag{4.21}$$

where η is the chosen maximal lag, and the standard estimator for the autocovariance terms is

$$\hat{\rho}_k = \frac{\sum_{i=t_0+1}^{N-\eta} f(X_i) f(X_{i+k})}{N - t_0 - \eta} - \frac{1}{2} \frac{(\sum_{i=t_0+1}^{N-\eta} f(X_i))^2 + (\sum_{i=t_0+1}^{N-\eta} f(X_{i+k}))^2}{(N - t_0 - \eta)^2}. \tag{4.22}$$

As for the choice of t_0 and η, in [14], we show that setting $t_0 := 0.1N$ and $\eta := 10N^{1/3}$ will result in a consistent estimate for σ^2. It is important to note that N and t_0 chosen for estimating σ^2 and V_f is not necessarily the same as ones used in subsequent runs of the Markov chain.

Having obtained $\hat{\sigma}^2$ and V_f, we can now calculate the estimated spectral gap as

$$\hat{\gamma} := 2\hat{\sigma}^2/\hat{V}_f. \tag{4.23}$$

4.5.2 Decoupling sampling and model checking

Typically, one will be interested in verifying several different properties of a model. It is impractical to re-run the full MCMC procedure for each property independently. We can exploit the fact that the Markov chain based sample collection is essentially independent from the model checking task. The sequence of parameter samples collected by the Markov chain only depends on the model and the experimental data, and not on the property that is being verified.

Running the Markov chain and performing model checking both involve simulating the system. Namely, the Markov chain simulates the system with each proposed parameter value, and evaluates the likelihood with respect to experimental data. Independent from this, the model checker needs to simulate the system (possibly under different initial conditions) to verify against a given temporal logic property. For these reasons, in practice, it is better to first run the Markov chain for a large number of steps "off-line", and store the collected parameter samples for later use in verification.

Assuming that a sufficiently long sequence of parameter samples has been stored, it is possible to run the same fix sample size or sequential hypothesis tests on this stored set of samples. In fact, there are two important optimizations that this enables in practice.

First, many of the parameters that the Markov chain generates are identical. This is because each time a proposed parameter is rejected, the previous parameter is kept (the Markov chain stays in its original state). Depending on the design, the Markov chain will typically have an acceptance rate between $10-40\%$. Naturally, it is enough to perform verification with each distinct parameter, and take into account the multiplicity of the parameter in the hypothesis test. For instance, assume parameter θ_i is kept for m steps before another parameter is accepted, then one would add $m \cdot I(\varsigma_{\theta_i} \models \psi)$ to the sum tracking the number of samples satisfying the property ψ in the hypothesis tests (see Algorithms 1 and 2 in Section 4.3).

Second, it is possible to parallelize the decoupled verification phase. For the fix sample size test, massive parallelization is possible, since each stored parameter can be

verified independently. For the sequential test, it is possible to introduce *batches* of samples that are verified in parallel. After verifying a batch of samples, the stopping condition of the sequential test is checked, and the procedure either stops and makes a decision, or another batch of samples is simulated and verified.

4.6 Case studies

The proposed method was implemented in C++, and runtimes were measured on a 2.83 GHz computer with 8GB of RAM. We first apply our method on a model of the JAK-STAT pathway and then on a larger model of extrinsic apoptosis.

4.6.1 The JAK-STAT pathway

A model of the JAK-STAT signaling pathway was introduced in Section 3.4.2. Here we use the same ODE model and experimental data, which have been presented. The model contains 4 unknown parameters. We use a Markov chain as introduced in Section 4.3 to collect samples from the space of parameters according to the posterior distribution, while evaluating the corresponding trajectories against properties of interest. The parameter ranges and the square root of the covariance matrix diagonal entries (σ_{MH}) used to define the MCMC proposal distribution are provided in Table 4.1.

Parameter	Limits	σ_{MH}
k_1	$[0, 5]$	0.02
k_2	$[0, 30]$	0.5
k_3	$[0, 1]$	0.01
k_4	$[0, 5]$	0.02

Table 4.1: Parameter ranges and entries in the proposal covariance matrix

We chose 16 discrete time points between 0 and 60 minutes to represent trajectories with respect to BLTL formulas (in the formulas below we will use the absolute time rather than the discrete time index).

We are mainly interested in the dynamics of nuclear STAT (STATn), since it is involved in gene expression [119]. Specifically, we verify dynamical properties of STATn under various types of Epo stimulation (Epo is an input set externally and does not appear in the formulas).

Property 1 STATn *reaches a* high *level and then settles at a* low *level under transient*

Epo stimulation

$$\psi_1 = F^{\leq 60}([1 \leq \text{STATn} \leq 2] \wedge F^{\leq 60}(G^{\leq 60}([0 \leq \text{STATn} \leq 0.5]))). \tag{4.24}$$

Experimental analysis of algorithms

We use the verification of $\mathbb{P}_{\geq r}(\psi_1)$ to empirically analyze different aspects of our approach. We ran $m = 2000$ independent instances of the MCMC sampler for a total of $2 \cdot 10^5$ steps each (with $t_0 = 10^4$ burn-in steps). We use the output of the m independent chains as a basis for constructing empirical results.

To get a reliable estimate of the true underlying probability of satisfaction P_{ψ_1}, we took the overall average of the estimates from all m chains, and treated the obtained value $P_{\psi_1} \approx \widehat{P}_{\psi_1} = 0.888$ as the reference for P_{ψ_1}. The method described Section 4.5.1 was used to estimate the value of the spectral gap to obtain $\gamma = 0.0127$.

We first examine the empirical error rate of the fixed sample size hypothesis test. For a fixed sample size n, we define the empirical error rate E_n as the ratio of chains choosing H_0 if H_1 holds (or the ratio choosing H_1 if H_0 holds). If neither H_0 nor H_1 holds (when $r - \delta < P_\psi < r + \delta$), then $E_n := 0$. We set $r = \widehat{P}_{\psi_1} - \delta$ and calculated E_n for a range of sample sizes up to $n = 2 \cdot 10^5$. For the same set of sample sizes, we calculated the error rate bound derived from equation (4.12) as $\epsilon_n = \exp(-n\gamma\delta)$.

Figure 4.2 shows E_n and ϵ_n as a function of n for different values of δ. It is apparent that E_n decreases monotonically with increasing sample size n, and that E_n is higher for lower values of δ. Importantly, Figure 4.2 demonstrates that the theoretical bounds on the error rate are reliable in practice, since the empirical false negative rate is consistently below this upper bound ($E_n \leq \epsilon_n$ for all examined n, δ).

We next look at results for sequential hypothesis testing. The sequential hypothesis test method enables the early termination of the sampling chain if the running estimate crosses a threshold. We refer to the number of samples collected in the Markov chain before a decision is made as the *stopping time* (see also Section 4.4). In Figure 4.3, the empirical cumulative distribution of stopping times is shown for the hypothesis test on P_{ψ_1} for a set of r values in $(0, 1)$. Here the value of $\delta = 0.05$ and $\epsilon = 0.01$ is fixed. As a reference, we note that the corresponding fixed sample size test would take $1.45 \cdot 10^5$ samples. The plot shows that for values of r distant from the true probability, sequential sampling consistently terminates with small variability at low sample sizes. When r is

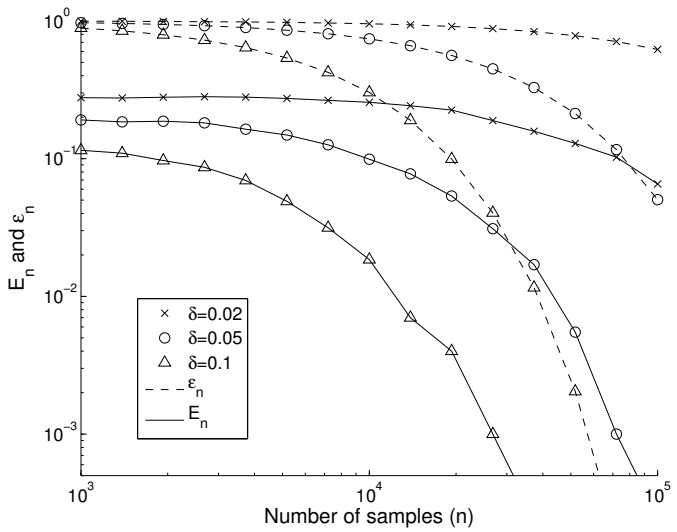


Figure 4.2: Empirical error rates for the fixed sample size test for a range of sample sizes. Dashed lines show the theoretical upper bounds derived from (4.12). Here $r = \widehat{P}_{\psi_1} - \delta$ and $\epsilon = 0.01$ are fixed, and 3 distinct δ values are shown.

close to the true probability, the stopping times show significantly higher variability. Figures 4.4 and 4.5 show the mean empirical stopping times for a range of r values for different values of δ, and different values of ϵ, respectively. For values of r close to $\widehat{P}_{\psi_1}$, some chains did not stop within $2 \cdot 10^5$ samples, and the corresponding mean values are therefore not determined. These empirical results are consistent with sequential hypothesis testing in the independent sample setting [132].

We also evaluated the empirical error rate in the sequential hypothesis test, and found that out of the $m = 2000$ parallel runs, no error was made under all examined choices of r, ϵ, δ. This, on one hand shows that the specified error bound (4.13) was indeed met. On the other hand, it suggests that the bound (4.13) might not be sharp and M could be chosen even smaller than described by (4.14), resulting in earlier stopping.

Further properties

We now look at two further properties regarding STATn. Recall that property ψ_1 specified the behavior of STATn under transient Epo stimulation. Here we specify the behavior of STATn under two rounds of transient Epo stimulation (ψ_2) and under sustained Epo stimulation (ψ_3), (again, Epo is set externally and thus does not appear in the formulas). Figure 4.6 shows 3 different time courses for the externally set Epo stimulation used when verifying with respect to ψ_1, ψ_2 and ψ_3.

Property 2 STATn *reaches a* high *level and then settles at a* medium *level under two*

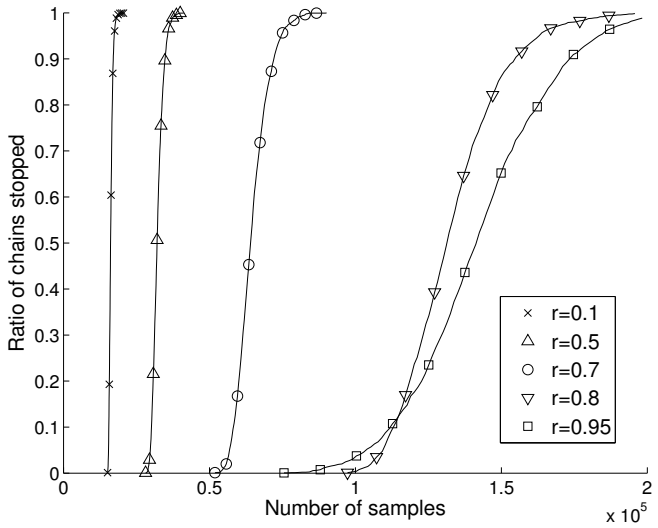


Figure 4.3: Empirical distribution of stopping times with sequential hypothesis test for different values of r. Here $\delta = 0.05$ and $\epsilon = 0.01$ is used.

rounds of transient Epo stimulation

$$\psi_2 = F^{\leq 60}([1 \leq \text{STATn} \leq 2] \wedge F^{\leq 60}(G^{\leq 60}([0.5 \leq \text{STATn} \leq 1]))). \tag{4.25}$$

Property 3 STATn *reaches a* very high *level and then settles at a* very high *level under sustained Epo stimulation*

$$\psi_3 = F^{\leq 60}(G^{\leq 60}([1.5 \leq \text{STATn} \leq 2])). \tag{4.26}$$

Table 4.2 summarizes the results of the verification with properties ψ_1 to ψ_3.

Property	r	δ	ϵ,	Outcome	Samples/time taken (sequential)	Samples/time taken (fixed)
$\mathbb{P}_{\geq r}(\psi_1)$	0.8	0.05	0.01	True	143384/315s	145045/319s
$\mathbb{P}_{\geq r}(\psi_2)$	0.8	0.05	0.01	True	54789/125s	145045/328s
$\mathbb{P}_{\geq r}(\psi_3)$	0.8	0.05	0.01	False	13698/36s	145045/317s

Table 4.2: Verification results on properties of the JAK-STAT pathway model. Outcomes (true or false) were the same for fixed length and sequential tests.

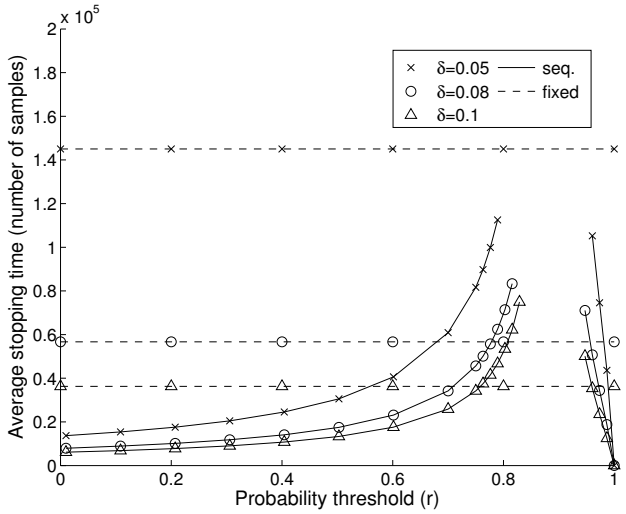


Figure 4.4: Mean empirical stopping times for sequential hypothesis test for different values of δ, with $\epsilon = 0.01$. Dashed lines show constant sample sizes required for the fixed sample size test.

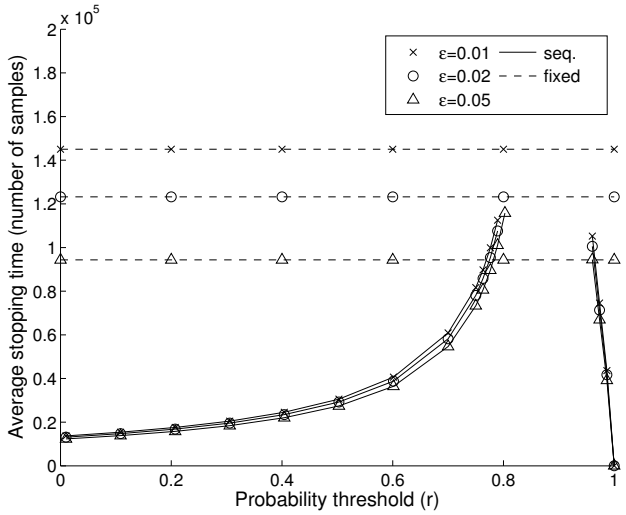


Figure 4.5: Mean empirical stopping times for sequential hypothesis test for different values of ϵ, with $\delta = 0.05$. Dashed lines show constant sample sizes required for the fixed sample size test.

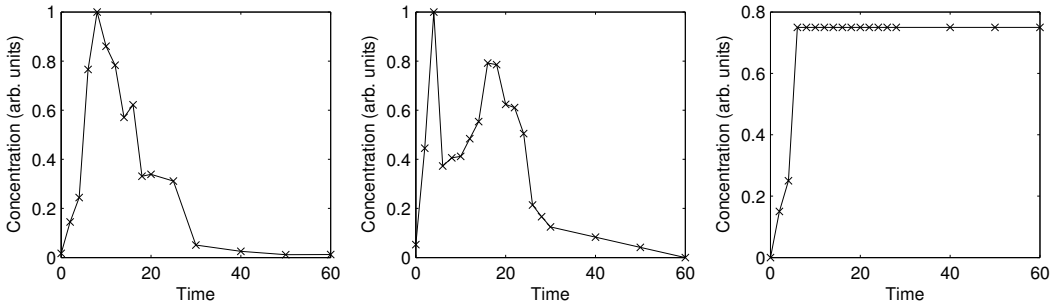


Figure 4.6: Epo stimulation dynamics. These time courses are used as (deterministic, externally fixed) inputs when verifying ψ_1, ψ_2 and ψ_3 respectively. Transient stimulation [119] (left), two rounds of transient stimulation [119] (center), sustained stimulation (right).

4.6.2 Extrinsic apoptosis reaction model

The extrinsic apoptosis reaction model (EARM) is an ODE model of apoptosis in response to the death ligand TRAIL. Several versions of the model were published in recent years, and the model has been used to classify cell types based on the apoptotic process, to study cell-cell variability due to varying initial conditions, and variability due to stochastic dynamics [138, 139, 140, 141, 142]. The work in [139] is specifically relevant, since there, model checking is used to characterize properties of the model under a range of initial conditions. We will verify similar properties of the model here, but under Bayesian parameter uncertainty. Here we use the version of the model called EARM1.3, following [16, 140].

The EARM1.3 model is considerably large. It contains 69 dynamical variables representing the active and inactive form of proteins and their complexes. We assume that 71 parameters are unknown, the set of unknown parameters being the ones representing the kinetic rates of binding, release and catalysis.

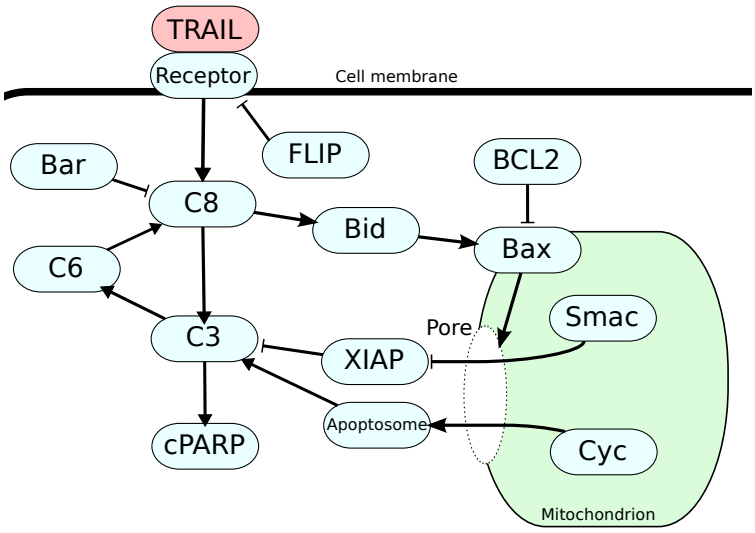


Figure 4.7: Simplified diagram of the EARM1.3 extrinsic apoptosis model.

Measurement data is available for the effector caspase reporter protein (EC-RP), which is a metric for the direct initiation of cell death. The EC-RP data is normalized to arbitrary units, and is compared to the model variable cPARP (cleaved PAPR protein) using a Gaussian likelihood function. There is significant uncertainty in many of the parameters (as also observed in [16]), partly due to the fact that measurements are noisy,

and are only available for the single, most downstream species. Accordingly, there is also variation in the dynamics of the internal, unobserved species. It is important to verify that certain characteristic properties hold, even under the model uncertainty arising from the limited data. In our setting, the EARM1.3 model is set up to represent HeLa cancer cells. We choose values for initial conditions, parameter prior distributions, degradation and synthesis rates following [140]. The ligand amount was set to 3000 units, which corresponds to 50ng/ml of TRAIL treatment in HeLa cells.

Name	Initial amount	Name	Initial amount
Receptor	1000	PARP	1000000
Flip	2000	Bid	60000
Caspase-8	10000	Mcl-1	20000
Bar	1000	Bax	80000
Caspase-3	10000	Bcl-2	30000
Caspase-6	10000	Pore	500000
Caspase-9	100000	CytoC_m	500000
Apaf	100000	Smac	100000
XIAP	100000		

Table 4.3: Initial amounts in the EARM1.3 pathway model.

According to [16], we use independent log-normal prior distributions for the parameters. The prior mean and variance for each parameter is shown in Table 4.4.

Name	(μ, σ^2)	Name	(μ, σ^2)	Name	(μ, σ^2)	Name	(μ, σ^2)
kf1	$(-6.4, 2.0)$	kr1	$(-6.0, 2.0)$	kc1	$(-2.0, 1.2)$	kf2	$(-6.0, 2.0)$
kr2	$(-3.0, 2.0)$	kf3	$(-7.0, 2.0)$	kr3	$(-3.0, 2.0)$	kc3	$(0.0, 1.2)$
kf4	$(-6.0, 2.0)$	kr4	$(-3.0, 2.0)$	kf5	$(-7.0, 2.0)$	kr5	$(-3.0, 2.0)$
kc5	$(0.0, 1.2)$	kf6	$(-7.0, 2.0)$	kr6	$(-3.0, 2.0)$	kc6	$(0.0, 1.2)$
kf7	$(-7.0, 2.0)$	kr7	$(-3.0, 2.0)$	kc7	$(0.0, 1.2)$	kf8	$(-5.7, 2.0)$
kr8	$(-3.0, 2.0)$	kc8	$(-1.0, 1.2)$	kf9	$(-6.0, 2.0)$	kr9	$(-3.0, 2.0)$
kc9	$(1.3, 1.2)$	kf10	$(-7.0, 2.0)$	kr10	$(-3.0, 2.0)$	kc10	$(0.0, 1.2)$
kf11	$(-6.0, 2.0)$	kr11	$(-3.0, 2.0)$	kf12	$(-7.0, 2.0)$	kr12	$(-3.0, 2.0)$
kc12	$(0.0, 1.2)$	kf13	$(-2.0, 2.0)$	kr13	$(0.0, 2.0)$	kf14	$(-4.0, 2.0)$
kr14	$(-3.0, 2.0)$	kf15	$(-3.7, 2.0)$	kr15	$(-3.0, 2.0)$	kf16	$(-4.0, 2.0)$
kr16	$(-3.0, 2.0)$	kf17	$(-3.7, 2.0)$	kr17	$(-3.0, 2.0)$	kf18	$(-4.0, 2.0)$
kr18	$(-3.0, 2.0)$	kf19	$(-4.0, 2.0)$	kr19	$(-3.0, 2.0)$	kc19	$(0.0, 1.2)$
kf20	$(-3.7, 2.0)$	kr20	$(-3.0, 2.0)$	kc20	$(1.0, 1.2)$	kf21	$(-3.7, 2.0)$
kr21	$(-3.0, 2.0)$	kc21	$(1.0, 1.2)$	kf22	$(0.0, 2.0)$	kr22	$(-2.0, 2.0)$
kf23	$(-6.3, 2.0)$	kr23	$(-3.0, 2.0)$	kc23	$(0.0, 1.2)$	kf24	$(-7.3, 2.0)$
kr24	$(-3.0, 2.0)$	kf25	$(-8.3, 2.0)$	kr25	$(-3.0, 2.0)$	kc25	$(0.0, 1.2)$
kf26	$(0.0, 2.0)$	kr26	$(-2.0, 2.0)$	kf27	$(-5.7, 2.0)$	kr27	$(-3.0, 2.0)$
kf28	$(-5.2, 2.0)$	kr28	$(-3.0, 2.0)$	kf31	$(-3.0, 2.0)$		

Table 4.4: Prior distributions for the parameters of the EARM model. The priors are log-normal, with variance σ^2 and mean chosen as $\mu := \log_{10} k_{\text{nom}}$, the logarithm of the nominal parameter values.

When running MCMC on the space of parameters, we propose steps in the logarithmic space with $\sigma_{\text{MH}} := 0.1875$. With this choice, an acceptance rate of $0.15 - 0.2$

was reached resulting in good mixing. We estimated the spectral gap of the Markov chain according to Section 4.5.1 and obtained $\gamma = 0.002$. We then used our fixed sample size and sequential hypothesis testing methods (with parameters $r = 0.9$, $\epsilon = 0.01$ and $\delta = 0.05$) to verify properties of the model. Note that with these parameters the fix sample size hypothesis test will always take 921035 samples, irrespective of the property being verified and the choice of r. We therefore ran the MCMC procedure for a total of 10^6 steps (with 10000 burn-in steps) and stored the sequence of parameters for subsequent verification (see the decoupling method proposed in 4.5.2). The MCMC procedure took less than 6 hours to run.

Time of apoptosis

First we were interested in verifying the timing of apoptosis. Apoptosis is marked by the cleavage of PARP, and cells can be considered dead once the amount of cleaved PARP (cPARP) reaches 50% of total PARP. With our initial conditions, this is at $5 \cdot 10^5$. The property φ_{1a} specifies that apoptosis will happen within 5 hours.

$$\varphi_{1a} = \mathbf{F}^{\leq 5}[\text{cPARP} > 5 \cdot 10^5]. \tag{4.27}$$

The property was verified to be *true*. We next verified whether all cells were alive within the first 3 hours.

$$\varphi_{1b} = \mathbf{G}^{\leq 3}[\text{cPARP} < 5 \cdot 10^5]. \tag{4.28}$$

This, again, was verified to be *true*. From here we see that the time of apoptosis falls between 3 and 5 hours after ligand treatment.

Effector caspase delay

In some cell lines the activity of the effector caspase almost immediately follows the activity of the upstream initiator caspase. In other cases there is significant delay between the two, possibly due to the need for mitochondrial membrane permeabilization (MOMP). We want to verify whether either the delayed effect or the immediate effect is preserved under model uncertainty. We express the activity of the initiator caspase through total cleaved Bid (tBid) reaching at least 10% of total Bid within 4 hours. Once this happens, we specify that cPARP stays below 50% of total PARP for 1 hour, mean-

ing that the effector caspase is not active during this time. The property is written as follows:

$$\varphi_{2a} = \mathbf{F}^{\leq 4}([\text{tBid} > 6000] \wedge \mathbf{G}^{\leq 1}[\text{cPARP} < 5 \cdot 10^5]). \tag{4.29}$$

The property is verified to be *false*. We tested the opposite property, namely that cells die *within* 1 hour of initiator caspase activity, expressed as

$$\varphi_{2b} = \mathbf{F}^{\leq 4}([\text{tBid} > 6000] \wedge \mathbf{F}^{\leq 1}[\text{cPARP} > 5 \cdot 10^5]). \tag{4.30}$$

This property was also verified as *false*. Hence we find that neither the immediate, nor the delayed effector caspase activity holds with high probability. In this case, our limited information about parameter values (the model uncertainty) does not allow us to conclude that either property holds motivating us to add more measurements to constrain our parameters and predictions.

We added measurements available for IC-RP [120, 16], the initiator caspase reporter protein. This reporter measures the activity of initiator caspases, and is a metric for the formation of tBid. We then ran the MCMC procedure again, this time using both EC-RP and IC-RP data. The results show that the new data was sufficient to reduce parameter uncertainty in making predictions about effector caspase delay. The properties on the time of apoptosis (φ_{1a} and φ_{1b}) still held *true* with at least 0.9 probability. Further, property φ_{2a} was now verified to be *true*. This indicates that there is, indeed, substantial delay in effector caspase activity.

Next we were interested in whether the property of effector caspase delay holds directly on the levels of the active caspases. This would require that EC-RP and IC-RP data together are sufficient to constrain the uncertainty in caspase dynamics. We formulated properties on the activity of Caspase-8 and Caspase-3 by assuming that the caspases can be considered active once the active form (marked by asterisk) reaches 0.01 times the total amount of caspase. The property φ_3 expresses that there is at least 1 hour of delay in Caspase-3 activation following Caspase-8 activation.

$$\varphi_3 = \mathbf{F}^{\leq 4}([\text{Caspase-8}^* > 100] \wedge \mathbf{G}^{\leq 1}[\text{Caspase-3}^* < 100]). \tag{4.31}$$

Interestingly, φ_3 could not be verified to hold true with high probability. The opposite property (stating that there is no delay) similarly, could not be verified to hold true with sufficiently high probability.

In summary, we found that model predictions depend heavily on the amount and nature of experimental data used to constrain model parameters. We could reliably predict the timing of cell death given EC-RP data. Adding IC-RP data constrained cleaved Bid dynamics sufficiently to find that, with high probability, there is substantial delay between initiator and effector caspase activity. However, the uncertainty in parameters prevented us from making reliable predictions using the model about caspase dynamics.

More details about the dynamics of species appearing in the specified properties is given in Appendix B.

4.7 Summary

In this chapter we proposed a method for performing statistical model checking on pathway models under Bayesian parameter uncertainty. This enabled us to verify if properties of interest expressed in temporal logic hold with respect to the parameter posterior distribution. This is important in systems biology since due to the large number of model parameters and limited, noisy experimental data, parameter values cannot usually be constrained to a single value. Despite parameter uncertainty, certain important properties of the model dynamics may be preserved, and our method enables the verification of these properties.

The main technical challenge in performing verification is that obtaining independent samples from the posterior is not possible. We relied on dependent realizations of the system collected using a Markov chain Monte Carlo method. Using error bounds on the finite sample size estimates obtained with MCMC, we derived hypothesis tests to decide whether a temporal logic property holds with at least a given probability.

Our first case study on the JAK-STAT pathway showed that even though parameter values cannot be well constrained, predictions made using the model (specifically the dynamics of nuclear STAT protein) are reliable. We also found that the empirical false decision rates for both the fixed sample size and sequential hypothesis tests are in all cases below our derived error bounds. Next we turned to a large ODE model of extrinsically triggered apoptosis. Here a very wide region of parameters could fit measurement data, and the dynamics of many species showed large variability. We found that properties involving species directly constrained by experimental data are

verified to hold with high probability. However, certain important properties of the model could not be verified to hold due to variability in dynamics.

Here we considered dynamical systems described as systems of ODEs. It is possible to generalize the methodology to continuous-time Markov chain (CTMC) and stochastic differential equation (SDE) models [143, 60]. In these models the system state over time is described by a stochastic process. By conditioning on observations, one can consider the posterior distribution of the system state (and any unobserved parameters), and verify the system's behavior with respect to this distribution. MCMC methods have been proposed for sampling the posterior in such models [144], and our probabilistic verification methods could be adapted to this context. It will also be interesting to examine the use of methods other than MCMC, such as particle filters or approximate Bayesian computation [64], however, rigorous bounds on the required sample size in these settings is still an open question.

Our case study on the extrinsic apoptosis model was set up to represent the typical characteristics of HeLa cells. There are versions of the model for other cell lines such as HCT115, SKW6.4 and T47D [139], and it would be interesting to verify properties for these cell lines as well. More generally, the case study showed that our proposed method works on large biochemical pathway models with real experimental data. This motives us to apply the method on other pathway models where limited experimental data could result in large uncertainty.

Chapter 5

Learning dynamic Bayesian network models of pathway dynamics

5.1 Introduction

Changes in signal transduction are a key component of many complex diseases including cancer. These changes are systemic in that their ground cause cannot usually be reduced to a single element or even a single pathway. Some of the acquired characteristics of cancer cells, including the ability to evade apoptosis and the insensitivity to anti-growth signals can be linked to simultaneous changes in multiple signaling pathways [145]. A key challenge is to understand the changes involved in the transformation to a diseased state, and to find treatments to reverse or mitigate the effect of these changes.

Computational models are essential in combining prior knowledge with experimental data, and making predictions in this context [4, 1]. The modeling formalism chosen needs to capture the dynamics of representative elements (usually the phosphorylation of proteins) across several relevant pathways. The commonly used ordinary differential equations (ODEs) formalism, which we have studied in Chapters 3 and 4, requires detailed information about the reactions and their kinetics. It is difficult to capture effects between elements that do not directly interact with each other, or when the mechanism of the interaction is not well understood. In addition, ODE models do not capture the stochasticity inherent in the dynamics of the underlying system. An alternative is to

work with stochastic models whose dynamics are described in terms of continuous time Markov chains [146, 147, 148]. However, these models are also based on molecular level interactions and due to many unknown rate constants, it is computationally challenging to deal with realistic biochemical networks. Scaling up is therefore restricted in both the number of elements that can be modeled, and also in the portion of overall cellular signaling covered. These properties highlight the need for a more abstract formalism that can overcome these limitations.

Probabilistic graphical models including Bayesian networks and dynamic Bayesian networks have received much attention as pathway models [12, 149, 150], primarily in the context of gene regulatory networks. Here random variables (nodes in the graphical model) are associated with the activity of a molecular species. In contrast with ODEs and CTMCs, these models capture the interactions between elements as conditional probability distributions, and do not require explicit knowledge of the underlying biochemical mechanisms. Boolean and logical models offer a similar, abstract description of relationships, but they are built on fixed, deterministic logical rules [47, 51]. Probabilistic models will be able to better handle and incorporate several sources of uncertainty. These include the noisiness of experimental data, the uncertainty arising from stochastic dynamics, the variability among a population of cells, and also the uncertainty induced by missing (unmodeled) elements.

In this chapter we propose dynamic Bayesian networks (DBNs) [10] as an appropriate model of signal transduction in this context. We develop a novel method for learning the parameters of dynamic Bayesian networks from experimental data, and then show how the model can be used to make predictions. Specifically, we use the learned DBN models to find "treatment" conditions that achieve specified system behaviors.

Via a separation of concerns, we assume the structure of the DBN is known. Specifically, we assume that directed interactions between the species involved in the pathway can be obtained using prior knowledge networks [151, 152]. We then tackle the problem of learning the relevant parameters of the DBN that are induced by the conditional probability tables associated with the nodes of the DBN. Existing Bayesian network learning algorithms [153, 49] rely on discretizing the data, and then counting the occurrence of value combinations in the data to determine conditional probability parameters. This approach will require a very large amount of data, and necessitates the discretization

of otherwise continuous data. Further, the conditional probabilities whose associated value combination did not appear in the data set will be undefined.

To address these shortcomings, we propose to learn DBN parameters using a constrained optimization approach. Linear constraints are used to incorporate prior knowledge about the nature of interactions, which can be typically classified as activations or inhibitions. The match to experimental data is captured by an objective function, which can be posed as a linear expression. Procedurally, the learning starts from an initial time slice, where the marginal probability distribution of each model variable is known. We formulate and solve a linear programming problem for each time slice by using the marginal distributions at the previous time slice. Linear programming (LP) is often tractable even for large problems and is guaranteed to find a globally optimal solution when it exists [154]. Previous work has proposed LP based approaches to structure learning [155, 156] for gene regulatory networks in a deterministic setting. In contrast, our method learns the *dynamics* of a probabilistic system model.

The second component of our work is a framework for finding treatment conditions to achieve specified dynamics. During disease progression cells undergo a transformation from a healthy to a diseased state. An important aspect of this transformation is a change in the way cells respond to external stimuli including growth factors and inflammatory signals. Using targeted perturbations (for instance with drugs that inhibit specific kinases) it is possible to reduce the effect of some of these changes, and this is the basis of several existing molecular drugs [157]. Using a DBN model it is possible to simulate the effect of such perturbations on elements of (possibly multiple) signaling pathways. An important challenge is that the state of the DBN is described by a complicated probability distribution. Consequently, we need to provide a way to formulate the biologically relevant properties which should be attained through perturbations, and show how to evaluate them on a DBN model.

We propose a framework based on a probabilistic temporal logic adapted for DBNs following [158]. The logic can be used to specify dynamical properties such as: "AKT reaches a high level with high probability under insulin stimulation". These types of statements are straightforward to construct and can be algorithmically verified when written as a temporal logic formula. Model checking has been previously applied to pathway models [71, 78] and also specifically to DBN models [158]. However, in these

studies the goal was to assert and verify the properties of a model. Here, instead, we exploit model checking to monitor the system dynamics under various treatment conditions, and determine which conditions will ensure the specified temporal patterns. If the set of all possible treatment conditions is discrete and small, we can exhaustively search through them and report the ones satisfying a property. Otherwise, we can use an optimization procedure on the space of possible treatments, and find ones with which the number of satisfied properties is maximized.

We used our approach to model a network of signaling proteins involved in the progression of liver cancer. This part of the work is in collaboration with the Department of Systems Biology, Harvard Medical School, where all experiments were performed. Protein phosphorylation data was collected for 12 key proteins taken from major cancer pathways in 4 cell types. The cell types comprise human primary hepatocytes (HPH), an immortalized hepatocyte cell line (HHL5), an early stage transformed hepatocyte line (HepG2), and a late stage transformed hepatocyte cell line (Focus). These cell types cover a range of stages from a healthy to a diseased, cancerous state.

The experimental data consists of measurements under multiple perturbations. Each perturbation involves stimulation with one of five ligands, combined with one of five small molecule drugs (kinase inhibitors). We have constructed a prior knowledge network (PKN) based on the signaling pathway database GeneGO (www.genego.com) from which we derived a DBN structure. We then used our approach to learn the parameters of a DBN model for each cell type. We validated our models by training with only a part of the available data and comparing predictions with the remaining, unseen data. Further, a limited number of additional treatment combinations (multiple ligands combined with multiple inhibitors) were measured for the HepG2 cell line, and our DBN model could accurately predict activity under these conditions.

Subsequently, using our probabilistic model checking method we searched for combinations of kinase inhibitors that modify certain properties of diseased cells to match those of healthy cells. This component of our work is motivated by increasing evidence that *combinations* of drugs can be significantly more effective than individual drugs for complex diseases including cancer [159]. Our approach can be adopted to search for such treatments and guide experiments in promising directions.

We defined a set of 18 characteristic dynamical properties that are verified to hold on

the DBN representing healthy cells. For instance, in one of the properties, we specified that the protein cJUN stays at a low level with high probability under TNFα stimulation. This holds in healthy cells but does not hold in diseased cells, and in fact, none of the single inhibitors can prevent sustained cJUN activation in Focus cells. We then searched the set of possible combinations of kinase inhibitors and evaluated them with respect to each of the 18 properties. Our analysis yielded interesting results, for instance, we found that on the Focus cell line, with any single inhibitor at most 6 of the properties can be satisfied. But when using the combination of 2 inhibitors, as much as 10 characteristic properties of healthy cells can be met.

The rest of this chapter is organized as follows. In Section 5.2, we introduce dynamic Bayesian networks as models of biological pathways and review previous work. We show how parameters defining the dynamics of these models can be learned using linear programming in Section 5.3. In Section 5.4, we propose a temporal logic and model checking framework to evaluate treatment conditions using the DBN model. In Section 5.5 we present a case study of our approach on liver cancer cell lines, and conclude the chapter with a summary.

5.2 Background and previous work

5.2.1 Bayesian and dynamic Bayesian networks

A technical introduction to Bayesian and dynamic Bayesian networks was given in Section 2.2.2. Here we review their use in the context of modeling biological pathways.

Bayesian networks have been used extensively in computational biology [160], and specifically in representing gene regulatory networks [149, 161, 10]. They are a natural choice in this context, since they allow the modeling of *indirect* and probabilistic influences among genes of interest. The structure of the model gives an easy to interpret, graphical representation of dependences (and conditional independences) between genes, without the need to include elements related to the exact mechanism of the effect, including mRNA and proteins. BNs have also been used to capture the structure of signaling pathways in [11]. Here phosphorylation was measured at a single cell level, on proteins involved in T-cell signaling. The single-cell data under multiple perturbation conditions enabled the learning of a Bayesian network structure among the proteins.

One limitation of BNs is their static nature, that is, the fact they do not include a time component. This precludes the modeling and prediction of time course dynamics. Even though time-resolved data is often available, this information is not exploited, and instead, activity or no activity is summarized by a single value (as in [11]). Another important limitation of Bayesian networks is that their structure is limited to acyclic graphs. It is not possible to model feedback loops – a very important characteristic of biological pathways – using Bayesian networks. These limitations are resolved by dynamic Bayesian networks, which we discuss next.

DBNs are a special class of Bayesian networks [50], which include an explicit time component. Each variable is represented at multiple time points, and edges represent dependencies among variables in time. A basic assumption about DBN structure when modeling biological pathways is that edges point forward in time [162]. This is a natural assumption when describing temporal physical processes. This implies that DBNs are guaranteed to be acyclic, and therefore feedback effects can be modeled across time points. The first work to propose using DBNs for learning gene regulatory pathways was [162]. Here the structural expectation maximization (SEM) algorithm was used to infer the topology of a synthetic network consisting of 5 genes. The ability to model feedback loops and to use time-course data motivated many other works in which the aim is to learn the structure of a gene regulatory network, including [163, 164, 165, 166] and [167].

More recently, DBNs were applied in the context of protein signaling in [12]. Here a conditional Gaussian parametrization of the variables is assumed, and with a specific choice of priors, the posterior probability of possible network structures can be obtained in a closed form. The method was used to learn the structure among signaling proteins in a breast-cancer cell line using reverse-phase protein array data. The structure inference revealed possible links among proteins that have not been known before.

In all the above works (both for BNs and DBNs), the goal was inferring the *structure* of a pathway model. These studies can elucidate important topological information, but do not consider DBNs as executable simulation models of pathway dynamics. A relevant work in this context by Liu et al. [15] proposes learning a DBN as an approximate but more efficient representation of an ODE model. The structure of the DBN is directly derived from the ODE equations, and repeated simulation of the ODEs is performed to

populate the conditional probability tables of the discrete DBN. Approximate inference methods on the DBN can then be used to perform simulation, sensitivity analysis and model checking [158]. An important limitation of this approach is that it requires an existing ODE model, which is in itself a considerable effort to build. Further, even with a very large set of generated ODE trajectories, many of the conditional probabilities in the DBN will be undefined. Our approach differs from [15] in several aspects. First, here we propose a method to learn conditional probabilities *directly* from experimental data rather than from an existing dynamical model. Further, we do not rely on discretizing and counting value combinations to determine conditional probability parameters. Instead we solve a constrained optimization problem in a way that the data is kept on its original, continuous scale. A useful feature of the constrained optimization approach is that through the imposed constraints, all entries in the conditional probability tables will be set.

5.2.2 Identifying drug effects

Biological networks are redundant and key processes are often controlled by several interacting pathways. Consequently, many diseases are difficult to treat using a drug targeting a single protein or gene. One possible solution is to use multiple drugs targeted at distinct pathway elements to achieve a coordinated effect [168]. However, experimentally evaluating such combinations is an arduous and expensive process due to the exponential blowup of possible combinations. Computational models of pathways could play a very important role in predicting responses to combination treatments, and several recent works have attempted to do this. A structure learning approach is considered in [169] and [170], where the effect of a set of cancer drugs is learned through the removal of edges from a signaling network model. This reveals interesting differences in the effect of each drug, however, does not provide a predictive, dynamical model of combinatorial drug effects. In [171] and [172], a generic template of ODE based coupling is proposed between proteins. In this model, the time derivative of a protein's concentration depends on a non-linear transfer function of the linear combination of its parents' concentration levels. The goal is to find an optimal model structure with respect to steady-state experimental data. The model is then used to predict the steady-state response under multiple perturbations. However, predicting transient *dynamics* under

perturbations are not considered. Further, a systematic way of finding combinations meeting given criteria is also missing. In Section 5.4 we will address these shortcomings.

5.3 Learning DBN parameters using linear programming

In this section we describe our method to learn the parameters of a DBN model from experimental data.

5.3.1 Structure from prior knowledge

First we assume that a signed prior knowledge network (PKN) is available. The PKN is defined on a set $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$ of N molecular species of interest. The PKN gives a signed parenthood relationship between these species via the function $PA : \mathcal{S} \to 2^{\mathcal{R}}$, where $\mathcal{R} = \mathcal{S} \times \{+, -\}$. If $(S_j, +)$ (or $(S_j, -)$) is in $PA(S_i)$ then this indicates that S_i acts directly as an activator (or inhibitor) on S_i. Such structural information is available from several free and commercial interaction network databases such as KEGG [173], GeneGo (www.genego.com), Cell Signal Technology (www.cellsignal.com) Science Database of Cell Signaling (www.stke.sciencemag.org), and the NCI Pathway Interaction Database [174]. All of these databases also provide information on the qualitative nature of connections, including labels for activation or inhibition.

The next step is constructing a DBN structure from the PKN. We assume that the structure of the DBN does not change over time, that is, the parenthood relationship of the PKN is kept through all time points of interest. This could be a strong assumption when considering long-timescale processes such as gene expression during development [175]. However, it is reasonable when modeling signal transduction due to the shorter time scale of interest.

The DBN is constructed by representing the set of species $\mathcal{S}$ as random variables at a finite, discrete set of time points $0, 1, \ldots, T$. At each time point t the node set will consist of $\mathcal{S}^t = \{S_i^t \mid 1 \leq i \leq N\}$. The edges of the DBN are defined through the parenthood relationships, namely, $(S_j^{t-1}, S_i^t) \in E$ iff $(S_j, -) \in PA(S_i)$ or $(S_j, +) \in PA(S_i)$. We also include edges from a variable to itself, that is, $(S_i^{t-1}, S_i^t) \in E$, for all $1 \leq i \leq N$. These edges (also called persistence edges in [162]) are used to express that once a species is activated, it may remain active (or deactivate slowly) irrespective of its parents. We

always set persistence edges with a positive sign.

A key assumption made when constructing a DBN is that edges only span between neighboring time points $t-1$ to t. This is a first-order Markov assumption on the dynamics of the underlying process. Namely, it implies $\mathcal{S}^{t+1} \perp\!\!\!\perp \mathcal{S}^{t-1} | \mathcal{S}^t$, which, with the parenthood relationship, is equivalent to the assumption that for each node S_i^t,

$$S_i^{t+1} \perp\!\!\!\perp (\mathcal{S}^{t-1}, \mathcal{S}^t \backslash PA(S_i^t)) \mid PA(S_i^t). \tag{5.1}$$

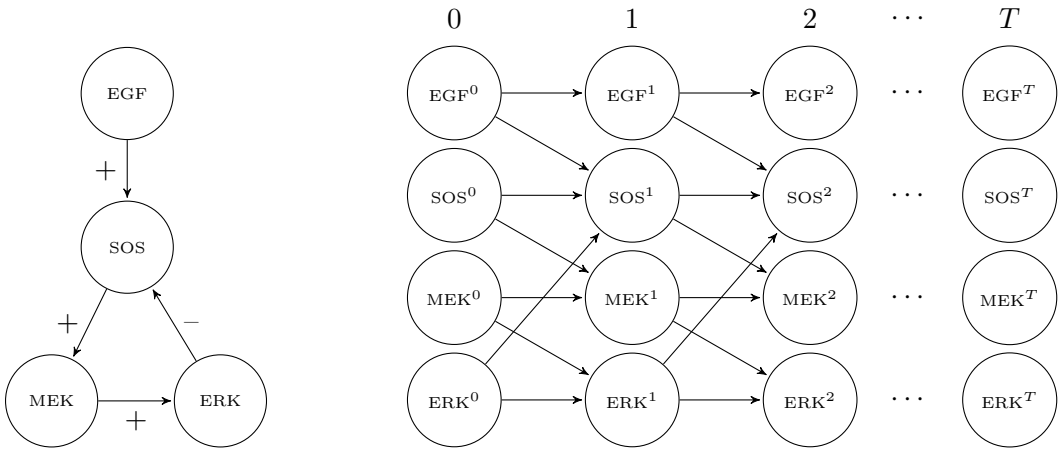
This will not hold if species other than the parents of S_i^t influence its value when going from $t-1$ to t. Therefore the accuracy of the DBN representation relies on choosing time points sufficiently close to each other. If measurement times are far apart compared to the speed of underlying dynamics, one solution is to use interpolation on the data points, as described in Section 5.3.5.

Example We use a small running example to illustrate some aspects of our approach. The model includes 4 proteins involved in epidermal growth factor (EGF) signaling. The PKN for the model is shown in Figure 5.1(a), and the corresponding DBN in Figure 5.1(b). The set of species in the PKN are $\mathcal{S} = \{\text{EGF, SOS, MEK, ERK}\}$, and the parenthood relationship describes, that, for instance, $PA(\text{SOS}) = \{(\text{EGF}, +), (\text{ERK}, -)\}$. In the DBN representation, random variables are introduced at each time point, for instance, at $t = 1$, we have nodes $S^1 = \{\text{EGF}^1, \text{SOS}^1, \text{MEK}^1, \text{ERK}^1\}$. The parenthood relationship is extended to the DBN as, for instance, $PA(\text{SOS}^2) = \{\text{SOS}^1, \text{EGF}^1, \text{ERK}^1\}$. The first-order Markov assumption imposed by the DBN would imply that, for instance, $\text{MEK}^1 \perp\!\!\!\perp \text{EGF}^0$, and $\text{MEK}^2 \perp\!\!\!\perp \text{EGF}^0 \,|\, (\text{SOS}^1, \text{MEK}^1)$.

5.3.2 Parametrization and constraints

The concentration level of each species in the DBN is represented as a discrete value taken from a finite set. For convenience we assume all species take values from $V = \{v_1, v_2, \ldots, v_K\}$, where V is a set of non-negative numbers with $v_1 < v_2 < \ldots < v_K$. However, our construction will naturally hold when variables take values from different discrete sets. With these choices, the parameters of the DBN can be summarized as a set of conditional probability tables (CPTs), one for each node.

The CPT for the node S_i^t is denoted Θ_i^t, $1 \leq t \leq T$, and it describes the probability



(a) PKN (b) DBN

Figure 5.1: The prior knowledge network (PKN) and the derived dynamic Bayesian network (DBN) representation of a small pathway.

of S_i^t taking a value in V, given a value assignment to its parents. As suggested by the notation, CPTs are assumed to be *time-variant*, meaning that, in general, Θ_i^t will not be equal to $\Theta_i^{t'}$ if $t \neq t'$. This is important especially if the time steps between available measurements are not uniform. The entries in the CPTs can be written as $\Theta_i^t = \{\theta_{i,j,k}^t\}$, where $\theta_{i,j,k}^t = P(S_i^t = v_k \mid PA(S_i^t) = \pi_{i,j})$. Here $\pi_{i,j}$ is the jth possible value assignment to the parents of S_i, and $j \in \{1, \ldots, K^{|PA(S_i)|}\}$, $i \in \{1, \ldots, N\}$ and $k \in \{1, \ldots, K\}$. We will naturally require that $\sum_{k=1}^{K} \theta_{i,j,k}^t = 1$.

For the variable S_i, the number of parent value assignments (we will also refer to these as parent configurations) is $K^{|PA(S_i)|}$, and for each parent configuration the CPT contains the entries of a conditional probability distribution described by K values. The total number of parameters in one time slice of the DBN is therefore $\sum_{i=1}^{N} K^{|PA(S_i)|+1}$. The number of parameters in each CPT grows exponentially with the number of parents, but does not depend on the overall number of variables or time steps. This highlights the fact that a sparse DBN structure (in which a node has few parents compared to the overall number of nodes) is a succinct representation of a large stochastic model.

A key idea in our approach is exploiting the labels in the PKN to impose constraints on our DBN parameters. We call these constraints *monotonicity constraints*, and base them on the following insights. A positive or activating relationship between a parent species and its target implies that if the parent takes on a higher value, the target's value should not decrease. Conversely, for a negative or inhibitory relationship, the parent's higher value implies that the target's value should not increase. In the context

of the DBN, we impose these constraints on the expected value of each species at time t conditioned on the value of its parents at time $t-1$. For instance, suppose that Y is a parent of X with a positive relationship, and suppose that $V = \{L, H\}$, with $L < H$. Then we demand that

$$\mathbb{E}(X^t | Y^{t-1} = L) \leq \mathbb{E}(X^t | Y^{t-1} = H), \tag{5.2}$$

capturing the intuition that as the concentration level of Y increases the concentration level of X ought to increase (or at least not decrease). The conditional expected values are easy to derive from the CPT entries, since

$$\mathbb{E}(S_i^t | PA(S_i^t) = \pi_{i,j}) = \sum_{k=1}^{K} v_k P(S_i^t = v_k | PA(S_i^t) = \pi_{i,j}) = \sum_{k=1}^{K} v_k \theta_{i,j,k}^t. \tag{5.3}$$

More systematically, we can define a partial ordering relation on the set of parent configurations via:

$$\pi_{i,j_1} \preccurlyeq \pi_{i,j_2} \text{ iff } \forall Y \in PA(S_i) \begin{cases} \pi_{i,j_1}(Y) \leq \pi_{i,j_2}(Y) \text{ and } Y \text{ is an activator, or} \\ \pi_{i,j_1}(Y) \geq \pi_{i,j_2}(Y) \text{ and } Y \text{ is an inhibitor.} \end{cases} \tag{5.4}$$

Here we use the notation $\pi_{i,j}(Y) \in V$ to refer to the value which is assigned to Y by the parent configuration $\pi_{i,j}$.

Example (continued) We continue the running example from the previous section. Assume that the set V contains two values, $v_1 = 0$ and $v_2 = 1$, and so $K = 2$. The CPT of MEK^1 (in generic notation S_3^1) would consist of 8 entries of the form

$$\begin{aligned} \theta_{311}^1 &= P\left(\text{MEK}^1 = 0 | \text{MEK}^0 = 0, \text{SOS}^0 = 0\right) \\ \theta_{312}^1 &= P\left(\text{MEK}^1 = 1 | \text{MEK}^0 = 0, \text{SOS}^0 = 0\right) \\ \theta_{321}^1 &= P\left(\text{MEK}^1 = 0 | \text{MEK}^0 = 1, \text{SOS}^0 = 0\right) \\ &\cdots \\ \theta_{342}^1 &= P\left(\text{MEK}^1 = 1 | \text{MEK}^0 = 1, \text{SOS}^0 = 1\right). \end{aligned}$$

Here, for instance, parent configuration $\pi_{3,1}$ corresponds to $\left(\text{MEK}^0 = 0, \text{SOS}^0 = 0\right)$ and $\pi_{3,2}$ to $\left(\text{MEK}^0 = 1, \text{SOS}^0 = 0\right)$. MEK has a positive edge from both of its parents, and the monotonicity constraints will give a partial ordering on the parent configurations as $\pi_{3,1} \preccurlyeq \pi_{3,2}$, $\pi_{3,1} \preccurlyeq \pi_{3,3}$, $\pi_{3,2} \preccurlyeq \pi_{3,4}$ and $\pi_{3,3} \preccurlyeq \pi_{3,4}$. The relationship between $\pi_{3,2}$ and $\pi_{3,3}$ is not constrained. The partial ordering is described by a total of 4 monotonicity constraints. In terms of the CPT entries, for example, the relationship $\pi_{3,1} \preccurlyeq \pi_{3,2}$ is

expressed as the constraint

$$P\left(\mathrm{MEK}^1 = 0 | \mathrm{MEK}^0 = 0, \mathrm{SOS}^0 = 0\right) \cdot 0 + P\left(\mathrm{MEK}^1 = 1 | \mathrm{MEK}^0 = 0, \mathrm{SOS}^0 = 0\right) \cdot 1 \leq$$
$$P\left(\mathrm{MEK}^1 = 0 | \mathrm{MEK}^0 = 1, \mathrm{SOS}^0 = 0\right) \cdot 0 + P\left(\mathrm{MEK}^1 = 1 | \mathrm{MEK}^0 = 1, \mathrm{SOS}^0 = 0\right) \cdot 1.$$

5.3.3 Experimental data

Next, we assume that a set of measurement data D is available, containing measurements for each modeled species $S_i \in \mathcal{S}$. The data set consists of observations of each species S_i at time points $1, \ldots, T$, under $C \geq 1$ experimental conditions, hence D is a data matrix of shape $N \times T \times C$. We denote by $D^t_{i,c}$ the measurement of S_i at time t under experimental condition $c \in \{1, \ldots, C\}$. Each experimental condition can be characterized by (i) a set of initial conditions (marginals) and (ii) a set of clamped nodes. The initial conditions describe the probability distribution of variables at the initial time point. The initial joint distribution is usually uncorrelated, and can be simply represented as a product of marginal distributions for each node. Clamped nodes have a fixed marginal distribution through the whole duration of the experiment. For instance, an inhibited node may be represented by a marginal distribution taking a low value with probability 1. We also allow for repeats of the same experiment by simply considering each repeat as an experimental condition, but with identical initial conditions and clamped nodes. Importantly, we assume that the experimental data is normalized (or the set of discrete value V is set) in a way that guarantees $v_1 \leq D^t_{i,c} \leq v_K$ for all i, t, and c. Since the data is on a continuous scale between v_1 and v_K, it makes sense to draw correspondence between the *expected value* of S^t_i under experimental condition c and the data point $D^t_{i,c}$. In the next section we describe an optimization problem that achieves this.

Example (continued) A dataset of our running example could consist of two experimental conditions ($C = 2$), corresponding to the existence or lack of EGF stimulation. This can be interpreted on the DBN by having the marginal of EGF clamped to a high value ($m^t_{1,1}(0) = 0, m^t_{1,1}(1) = 1, t \in \{0, \ldots, T\}$) in the first condition, and to a low value ($m^t_{1,2}(0) = 1, m^t_{1,2}(1) = 0, t \in \{0, \ldots, T\}$) in the second condition. In both conditions, the initial distributions of the remaining three species could be set to a low value: $m^0_{i,c}(0) = 1, m^0_{i,c}(1) = 0, i \in \{2, 3, 4\}, c \in \{1, 2\}$.

5.3.4 Parameter optimization

Our goal is to find parameters for our DBN model such that the expected values of species, as inferred from the DBN, match the experimental data. Taken together with the monotonicity constraints, we can pose this as a constrained optimization problem.

The main idea behind our parameter optimization is to start from the initial marginals defined by the experimental conditions, and iteratively (i) learn the parameters for the next time point based on the data for that time point, and then (ii) infer the marginals for each experimental condition at the next time point to be able to continue with step (i). This is an iterative process, which, after T steps will give us the parameters at all time steps.

Since both the monotonicity constraints and the fit to data will be defined in terms of expected values, it makes sense to pose the optimization problem in terms of conditional expectations rather than directly the CPT entries, which are conditional probabilities. We denote the conditional expectations as $\theta^t_{i,j} = \mathbb{E}(S^t_i \mid PA(S^t_i) = \pi_{i,j})$, and draw the connection between these parameters and the CPT entries as

$$\theta^t_{i,j} := \sum_{k=1}^{K} v_k \theta^t_{i,j,k}. \tag{5.5}$$

Inductively assume we have marginals available at $t-1$ with $t \geq 1$. (Note that the initial marginals ($t = 0$) are fully defined by the experimental conditions.) Each marginal will be a vector describing the probability distribution of node S^t_i for each experimental condition c, and is denoted as $m^{t-1}_{i,c} := (m^{t-1}_{i,c}(v_1), \ldots, m^{t-1}_{i,c}(v_K))$. Through the (yet unknown) parameters $\theta^t_{i,j}$, we can project these marginals, approximately, to expected values at time step t through the formula

$$\mathbb{E}(S^t_{i,c}) = \sum_{j=1}^{K^{|PA(S_i)|}} \theta^t_{i,j} P_c(PA(S^t_i) = \pi_{i,j}) \tag{5.6}$$

$$\approx \sum_{j=1}^{K^{|PA(S_i)|}} \theta^t_{i,j} \prod_{S_\ell \in PA(S_i)} m^{t-1}_{\ell,c}(\pi_{i,j}(S_\ell)). \tag{5.7}$$

In the first equation $P_c(PA(S^t_i) = \pi_{i,j})$ expresses the joint probability of the parent configuration $\pi_{i,j}$ under experimental condition c. The second equation then uses an assumption of independence to approximate this joint probability by a product of marginal probabilities, where $m^{t-1}_{\ell,c}(\pi_{i,j}(S_\ell))$ is the marginal probability assigned to $S_\ell \in PA(S_i)$

at time $t-1$ by the configuration $\pi_{i,j}$. The approximation of joint probabilities by a product of marginals also appears in the widely used factored frontier (FF) algorithm [176], and is shown to often work well in practice. This approximation is necessary for large models, since representing the joint distribution would be intractable.

We are now ready to formulate the constrained optimization problem for time t by collecting what we have discussed so far, as follows:

$$\begin{aligned}
\text{Boundary constraints:} \quad & \forall i,j : v_1 \leq \theta^t_{i,j} \leq v_K, \\
\text{Monotonicity constraints:} \quad & \forall i, j_1, j_2 \text{ for which } \pi_{i,j_1} \preccurlyeq \pi_{i,j_2} : \theta^t_{i,j_1} \leq \theta^t_{i,j_2}, \\
\text{Objective value:} \quad & F^t = \sum_{i=1}^{N}\sum_{c=1}^{C} d\left(E^t_{i,c}\,,\, D^t_{i,c}\right) \\
& = \sum_{i=1}^{N}\sum_{c=1}^{C} d\left(\sum_{j=1}^{K^{|PA(S_i)|}} \theta^t_{i,j} \prod_{S_\ell \in PA(S_i)} m^{t-1}_{\ell,c}(\pi_{i,j}(S_\ell))\,,\, D^t_{i,c}\right)
\end{aligned} \tag{5.8}$$

Here d is a distance measure between a data point and an expected value. Notice that the constraints are linear, and the expected value is also a linear function of the parameters. Therefore, if we choose d to be the L1 distance, the problem can be expressed as a linear programming (LP) problem. It is also possible to choose the L2 distance, in which case the problem becomes one of quadratic programming (QP), or more specifically, a linearly constrained least-squares problem. Section 5.5.4 will show that the predictive performance of using either method is similar. Therefore we focus on LP, since it can be solved more efficiently.

The general form of linear programming is

$$\begin{aligned}
&\text{minimize } f^T x \\
&\text{subject to } Ax \leq b,
\end{aligned} \tag{5.9}$$

where x is a vector of unknown variables (corresponding to the vector of parameters $\theta^t_{i,j}$ in our case), the coefficients in f correspond to the objective function, and A and b specify the constraints. The boundary and monotonicity constraints in (5.8) are trivially expressed through A and b. We now show how the objective value can be expressed as

a linear function. The objective value, using the L1 distance, is

$$F^t = \sum_{i=1}^{N} \sum_{c=1}^{C} \left| E^t_{i,c} - D^t_{i,c} \right| . \tag{5.10}$$

This can be expressed as a linear function by using slack variables $y^t_{i,c}$ (see also [154]). Namely, we can add the constraints

$$y^t_{i,c} \geq E^t_{i,c} - D^t_{i,c} \text{ and} \tag{5.11}$$

$$y^t_{i,c} \geq D^t_{i,c} - E^t_{i,c}, \tag{5.12}$$

and use the objective value $F'^t := \sum_{i=1}^{N} \sum_{c=1}^{C} y^t_{i,c}$.

The linear programming problem can be solved efficiently using standard algorithms available in many free and commercial software packages (for more details, see Section 5.3.5). The solution of the optimization problem gives us the values of conditional expected value parameters $\theta^t_{i,j}$. We now need to reconstruct conditional distributions from these values. This is an ill-posed problem for $K \geq 2$, since the distribution will have K entries, while the expected value is a single scalar. To resolve this, we use the minimum variance criterion, which finds the distribution with minimum variance, whose expected value is consistent with the conditional expectation. That is, we set $\theta^t_{i,j,k}$ such that

$$\begin{aligned} &\sum_{k=1}^{K} v_k \theta^t_{i,j,k} = \theta^t_{i,j} \text{ and} \\ &\sum_{k=1}^{K} (v_k - \theta^t_{i,j})^2 \theta^t_{i,j,k} \text{ is minimized .} \end{aligned} \tag{5.13}$$

The minimum variance criterion is reasonable, since, lacking information about the exact shape of the distribution, we are characterizing it by its first moment.

Having obtained $\theta^t_{i,j,k}$, for all i, j, k, we can now infer marginals at time t from time $t-1$ using the same assumptions as in equation (5.6) as

$$m^t_{i,c}(v_k) := \sum_{j=1}^{K^{|PA(S_i)|}} \theta^t_{i,j,k} \prod_{S_\ell \in PA(S_i)} m^{t-1}_{\ell,c}(\pi_{i,j}(S_\ell)). \tag{5.14}$$

Since we now have available all the marginals at time t for experimental conditions $c \in \{1, \ldots, C\}$, we can proceed to the next time step to learn $\theta^{t+1}_{i,j,k}$, and continue iteratively up to the last time T.

5.3.5 Properties and extensions

Properties of the LP problems

We observe that in (5.8) the constraints on the parameters are independent for each species. Further, the objective function is a summation of species-wise objective functions. This is because the optimization is solved step by step in time, hence the parameters, and the marginals for each condition at the previous time point are fixed. Therefore, in the present construction, the parameters can be learned locally for each species. In fact, at each time point t, we can solve the N local LP problems in parallel.

This decomposability is also crucial, since it means that the size of each LP problem is locally determined. Namely, for each node, the LP will be of exponential size in the number of parents but independent of the overall number of nodes.

Proposition 1. *(1) The number of variables (other than slack variables) solved for by the decomposed LP problem for S_i^t is $K^{|PA(S_i)|}$. (2) The number of monotonicity constraints needed to fully specify the partial ordering is $|PA(S_i)|K^{|PA(S_i)|-1}(K-1)$.*

Proof. Part (1) holds since the variables in the LP problem are the set of conditional expected values $\{\theta_{i,j}^t\}$. There is one such expected value for each parent configuration, the total number of which is $K^{|PA(S_i)|}$. For part (2), consider that with all other parents fixed, a parent will have $K-1$ monotonicity constraints between its value assignments $v_1, v_2, \ldots, v_K$, and The number of possible fixed configurations for all other parents is $K^{|PA(S_i)|-1}$. □

In a pathway, the number of species that *directly* influence any particular species is often limited and thus the graph of DBN will usually be sparsely connected. As seen from Proposition 1 each individual LP problem will remain tractable even for large pathways.

We can also easily see that the LP problem for each S_i^t will always have a solution.

Proposition 2. *The LP problem posed at node S_i^t will always be feasible and have an optimal solution.*

Proof. The LP problem is feasible if there exists a value assignment to the unknown variables which satisfy all the constraints. In our case, simply setting $\theta_{i,j}^t = \nu$, where ν is a constant such that $v_1 \leq \nu \leq v_K$, will trivially satisfy both the boundary and

monotonicity constraints. The slack variables will also always be feasible simply by $y^t_{i,c} \geq |D^t_{i,c} - E^t_{i,c}|$. Further, since the slack variables are bounded from below, and we are solving a minimization problem on the sum of slack variables, an optimal solution is guaranteed to exist. □

Solving the LP problems

Any optimal solution to a feasible LP problem is also globally optimal, and can be found in polynomial time [154]. Several algorithms including the simplex method and interior point methods have been proposed and implemented in tools to solve large LP problems efficiently in practice. It is often the case that more than one optimal solution exists for the optimization problem. One shortcoming of LP solvers is that they only return a single optimal solution. Therefore our learned DBN will be based on only one of potentially many existing solutions.

The most important free LP solvers include lp_solve [177] and the GNU Linear Programming Kit (www.gnu.org). Several commercial solvers including IBM CPLEX (www.ibm.com), and ones included with software such as MATLAB (www.mathworks.com) are also available. Benchmarks show that current solvers can handle on the order of 10^6 variables and constraints [178].

Interpolation

Until now, we have taken the time slices of the DBN model to correspond to the time points where measurement data is available. However, effects in the underlying system could propagate at a speed which is not necessarily consistent with the time points when measurement data is obtained. The first-order Markov assumption may be too restrictive to adequately represent the system dynamics. This can be resolved by introducing additional, intermediate time slices in the DBN. By introducing these additional time slices, variables that are correlated in the underlying system can be correlated in the DBN as well. However, data is needed to formulate the objective function of the optimization problem at each time step. Therefore we provide a simple interpolation method by which the formulation of an objective function becomes possible at intermediate time steps.

Based on the DBN structure we first compute T_{corr}, the number of time steps after

which all variables that can get correlated will be correlated. Between any two time points t and $t+1$ at which we have data, we introduce $T_{\text{corr}} - 1$ additional time points and use linearly interpolated values as data for these time points using the actual data values available at t and $t+1$. Suppose we introduce the time points $(\tau, \tau+1, \ldots, \tau+T_{\text{corr}})$ to replace $(t, t+1)$. Then the associated data points are set as

$$\widetilde{D}_{i,c}^{\tau+k} = D_{i,c}^{t} + \frac{k}{T_{\text{corr}}}(D_{i,c}^{t+1} - D_{i,c}^{t}) \text{ for } 0 \leq k \leq T_{\text{corr}}. \tag{5.15}$$

Using these data points, we can formulate a sequence of LP problems to learn the DBN parameters.

Example (continued) In our running example, $T_{\text{corr}} = 3$, since EGF^0 will be first correlated with ERK^3 through the path $\text{EGF}^0 \rightarrow \text{SOS}^1 \rightarrow \text{MEK}^2 \rightarrow \text{ERK}^3$.

5.4 Treatment evaluation using model checking

The second main contribution of this chapter is developing the means for identifying treatment conditions using which the dynamics of the pathways under study can be reshaped. Treatment conditions typically consist of the addition of ligands or inhibitors, which stimulate or block certain pathway components. Under treatment conditions, pathways will change their dynamics in response to internal or environmental cues. It is important to predict these changes computationally, since the set of possible treatments is usually combinatorial, and only a small subset can be studied experimentally. If we are able to find possibly relevant treatments with the help of the model, experiments can be focused towards verifying this limited set of conditions.

Treatments on a pathway can be translated to DBN models through imposing initial conditions and clamping nodes. Having learned a DBN model using the method described in Section 5.4, we can use inference to recover the state distribution under various treatments. However, characterizing the "behavior" of a DBN model is challenging since the system state is described by a sequence of probability distributions in time. A language or formalism is needed to make statements about the state trajectory. This formalism needs to be interpretable on a DBN, and needs to be able to express biologically relevant behavior types. Here we propose to use a probabilistic temporal logic to make statements about the temporal evolutions of the probabilistic system state.

In a discrete DBN the marginal distribution of variables can be efficiently recovered using inference. This motivates us to make temporal logic statements about the state evolution of the system through its sequence of marginal distributions. An appropriate temporal logic will therefore be interpreted on a sequence of probability distributions. This is different from the logic used in Chapter 4 (PBLTL) in that here we will make statements about the marginal distributions of variables, rather than individual realizations of the state dynamics. Such logics have been developed before in the literature for Markov chains [179, 180], and also specifically for DBNs [158]. We adopt here a probabilistic bounded linear temporal logic (PBL) from [158]. By making statements explicitly on marginal distributions, PBL is well suited for performing model checking on DBNs.

5.4.1 Probabilistic temporal logic for the DBN

The atomic propositions of PBL are defined as $(i, v)\#r$, where i refers to the species $S_i \in \mathcal{S}$, $v \in V$, $\# \in \{\leq, \geq\}$ and r is a probability threshold in $(0, 1)$. The atomic proposition $(i, v) \geq r$ is satisfied at time t if and only if $m_i^t(v) \geq r$ (similarly for $(i, v) \leq r$), where $m_i^t(v)$ is the marginal probability of species S_i being at level v at time t.

The formulas of PBL are generated by the following syntax:

- The truth constants $true$ and $false$ are formulas.
- Every atomic proposition $(i, v)\#r$ is a formula.
- If φ and φ' are formulas then so are $\neg\varphi$ and $\varphi \vee \varphi'$.
- If φ and φ' are formulas then so is $\varphi \mathbf{U} \varphi'$.

The temporal operators **G** (always from now), **F** (some time from now) can be derived as $\mathbf{F}\varphi = true\mathbf{U}\varphi$; $\mathbf{G}\varphi = \neg\mathbf{F}(\neg\varphi)$. The logical operators $\wedge, \Rightarrow$ and $\Leftrightarrow$ are defined as follows. $\varphi \wedge \varphi' = \neg(\neg\varphi \vee \neg\varphi')$, $(\varphi \Rightarrow \varphi') = (\neg\varphi \vee \varphi')$, and $(\varphi \Leftrightarrow \varphi') = (\varphi \Rightarrow \varphi') \wedge (\varphi' \Rightarrow \varphi)$. This syntax allow us to express a range of system dynamics patterns as illustrated in Table 5.1.

The formulas of PBL are interpreted on the sequence of marginal distribution vectors $\sigma = s_0 s_1 \dots s_T$ of the DBN. Here the state $s_t = (m_1^t, m_2^t, \dots, m_n^t)$ is a vector of marginal

Required dynamics	**Formula**	**Explanation**
ERK is not activated	$\mathbf{G}((\text{ERK}, v_L) \geq 0.8)$	ERK is always at a low level with high probability.
ERK is transiently activated	$\mathbf{F}((\text{ERK}, v_H) \geq 0.8) \wedge \mathbf{F}(\mathbf{G}((\text{ERK}, v_L) \geq 0.8))$	ERK reaches a high level after which it remains at a low level.
ERK is activated and sustained	$\mathbf{F}(\mathbf{G}((\text{ERK}, v_H) \geq 0.8))$	ERK reaches a high level and stays at a high level.

Table 5.1: Examples of PBL properties for the dynamics of the protein ERK. Here $V := \{v_L, v_H\}$, and v_L denotes a low value and v_H a high value.

distribution for each variable in $S_i^t \in S^t, 1 \leq i \leq N$. We let $\sigma(t) = s_t$ and $s_t(i) = m_i^t$. We let $\sigma(t) \models \varphi$ denote that φ holds at t, and define its semantics as follows.

- $\sigma(t) \models (i, v) \geq, r$ iff $m_i^t(v) \geq r$,
- $\sigma(t) \models (i, v) \leq, r$ iff $m_i^t(v) \leq r$,
- $\sigma(t) \models \neg\phi$ iff $\sigma(t) \not\models \varphi$
- $\sigma(t) \models \phi \vee \phi'$ iff $\sigma(t) \models \varphi$ or $\sigma(t) \models \varphi'$
- $\sigma(t) \models \mathbf{O}\varphi$ iff $\sigma(t+1) \models \varphi$,
- $\sigma(t) \models \varphi \mathbf{U} \varphi'$ iff $\exists t' : t \leq t' \leq T$ and $\sigma(t') \models \varphi'$, and $\forall t'' : t \leq t'' < t', \sigma(t'') \models \varphi$.

Using the above inductive definition, we say that the DBN $\mathcal{D}$ meets the specifications φ iff $\sigma(0) \models \varphi$, and we denote this as $\mathcal{D} \models \varphi$.

The task of a model checker is to determine whether a specification φ is met. Given a sequence of marginals σ and a specification φ the model checker returns either *true* or *false* depending on whether $\sigma(0) \models \varphi$. We perform model checking in an on-line fashion, in which it is possible to stop early if the truthhood of the property can already be determined. Such an on-line model checker is presented in [158], which we refer to for more details.

In our setting one performs inference on the DBN to reproduce the sequence of marginals, which the model checker can verify. We now look at inference algorithms on the DBN.

5.4.2 Inference on the DBN

Inference on a DBN involves recovering the joint probability distribution of the set of variables at each time point. The joint distribution at a given time point is called the *belief state*. Exact inference on DBNs is intractable in general [50, 181]. In fact, for DBNs with many variables, even representing the belief state is intractable, since with N variables, K^N entries are needed to represent it. Approximate inference algorithms rely on breaking up the belief state into a product of lower dimensional distributions. The Boyen-Koller (BK) algorithm [181] propagates joint distributions on smaller clusters of variables, whose product is a surrogate for the full belief state. The factored frontier (FF) algorithm [50] uses a cruder representation of the belief state, by only maintaining the marginal distribution of each node. The belief state is then approximated by the product of individual marginals. The marginals at the previous time slice are mapped directly to the ones at the next time slice, without reconstructing the full belief state. Recently a parameterized version of FF called hybrid FF (HFF) was proposed [182]. The idea behind HFF is to maintain a limited number of *spikes* corresponding to entries in the full belief state, resulting in improved accuracy.

Here we use the factored frontier algorithm (FF) since it is efficient for large pathway models. This is crucial, since we will need to use inference repeatedly, under a large number of conditions, when searching for treatments. We now describe the FF algorithm in more detail.

We denote the marginal distribution of the variable S_i^t as m_i^t with $m_i^t(v)$ denoting the (marginal) probability of S_i^t assuming the value v, with $v \in V$. Given the marginal distribution of species at the initial time point $t = 0$, the task is to infer the marginals at the time points $t = 1, 2, \ldots, T$. The approximate marginals FF computes are denoted as $\hat{m}_i^t$. Starting with $\hat{m}_i^0 = m_i^0$, FF uses the following scheme to propagate the approximate marginals:

$$\begin{aligned} \hat{m}_i^t(v) &= \sum_{j=1}^{K^{|PA(S_i)|}} P(S_i^t = v | PA(S_i) = \pi_{i,j}) \prod_{S_\ell \in PA(S_i)} \hat{m}_\ell^{t-1}(\pi_{i,j}(S_\ell)) \\ &= \sum_{j=1}^{K^{|PA(S_i)|}} \theta_{i,j,k}^t \prod_{S_\ell \in PA(S_i)} \hat{m}_\ell^{t-1}(\pi_{i,j}(S_\ell)). \end{aligned} \tag{5.16}$$

Recall that $\pi_{i,j}(S_\ell) \in V$ denotes the value assigned to $S_\ell \in PA(S_i)$ in the parent

configuration $\pi_{i,j}$, and the conditional probability parameters $\theta^t_{i,j,k}$ are obtained using the method outlined in Section 5.3.

The set of marginals thus computed can be viewed to be an approximate, factored representation of the joint system state and can be used to monitor the evolution of the system state over time.

5.4.3 Treatment evaluation

Model checking is commonly used to verify inherent properties of a model. Here we propose to use model checking as a method to monitor the behavior of a DBN model under multiple conditions, and find ones with which a specified property is met. A treatment condition can be characterized by clamping the marginal distribution of a subset of nodes to an externally set value. For instance, we can model the effect of a molecular *inhibitor* by clamping its target to a marginal taking a low value with high probability. Alternatively, if the inhibitor itself is a node in the DBN, one can set its marginal to one that assigns high probability to a high value.

More generally, assume that we can clamp a subset of species $Z \subset S$, and let $Z = \{Z_1, Z_2, \ldots, Z_m\}$. The clamped species can be associated with their corresponding DBN nodes in the obvious way. A treatment condition can be described by the set of externally fixed marginals $M_Z := \{m^t_{Z_i}\}$, $1 \leq i \leq m$, $0 \leq t \leq T$. There are technical restrictions on the marginals, namely, we require $m^t_{Z_i}(v) \geq 0$ and $\sum_{v \in V} m^t_{Z_i}(v) = 1$.

The effect of a treatment can be evaluated by performing inference on the DBN with respect to M_Z. In Section 5.4.1 we defined how $\mathcal{D} \models \varphi$ (that is, that the DBN $\mathcal{D}$ meets the property φ) can be determined using the sequence of marginals for each time point. We will denote the DBN model subject to the treatment condition M_Z as $\mathcal{D}_{M_Z}$. Consequently, we will say that the DBN meets the property φ under the treatment M_Z if the marginals, when performing inference under M_Z, meet φ. We denote this as $\mathcal{D}_{M_Z} \models \varphi$. This gives the conceptual basis of evaluating treatment conditions with respect to dynamical properties on a DBN.

Example (continued from Section 5.3) Recall the example DBN model in Figure 5.1. Assume that we have inhibitors available for SOS or MEK ($Z = \{\mathrm{SOS}, \mathrm{MEK}\}$). One treatment condition could involve inhibiting MEK alone ($Z' = \{MEK\}$). This

can be achieved by clamping MEK to a low value with high probability, that is, $M_{Z'}$ is defined as setting $m^0_{\text{MEK}}(0) := 1, m^0_{\text{MEK}}(1) := 0$, and then letting $m^t_{\text{MEK}} := m^{t-1}_{\text{MEK}}$ for $t = 1, 2, \ldots, T$. It is then possible to verify if $\mathcal{D}_{M_{Z'}} \models \varphi$, where φ is a PBL property.

5.4.4 Treatment finding

Having the means to evaluate the effect of a given treatment, we can also pose the inverse problem of finding a treatment with given properties. There will usually be several properties of interest $\varphi_1, \varphi_2, \ldots, \varphi_L$ which should be satisfied using a treatment condition. Our goal will be to look for treatments that maximize the number of satisfied properties. Until now we have considered combinations of both ligand stimuli and molecular inhibitors as possible treatment conditions. However, here it makes sense to focus only on inhibitors, since stimulation by ligands is controlled by the cell's environment, and usually not part of a drug treatment regime. Further, we will assume that if a node is clamped, its marginal is set to a pre-determined fixed value and the marginal entries are not, themselves, subject to optimization. With these assumptions a treatment will be fully characterized by $Z' \subseteq Z$, since the marginals $M_{Z'}$ will be a fix function of Z'.

Our goal will now be to

$$\begin{aligned} &\text{find } Z' \subseteq Z \\ &\text{such that } \sum_{i=1}^{L} \mathcal{D}_{M_{Z'}} \models \varphi_i \text{ is maximized.} \end{aligned} \tag{5.17}$$

In other words, we are searching for the subset of clamped nodes with which the highest number of properties are satisfied. There are other constraints we can include in the optimization problem. For instance, we can restrict the maximum number of clamped nodes to a fixed value (e.g. require that $|Z'| = 2$ to search for the best *pair* of clamped nodes).

If the set of possible treatment conditions is low, an exhaustive search over the subsets of Z will be possible. Otherwise we can use a global optimization method on the discrete search space to find the best treatment. A treatment condition can be encoded as a binary vector of length $|Z|$, where the ith component of the vector is 1 iff $Z_i \in Z'$. Many discrete search methods exist that can solve a maximization problem on such a vector, including tabu search [183] and genetic algorithms [57]. More recently a discrete version of the Hooke-Jeeves method was used to perform optimization on

pathway models [158] and can also be applied here.

5.5 Modeling signaling in liver cancer cell lines

In this section we apply our proposed method to model signaling in the progression of liver cancer [184, 43]. We learn 4 DBN models corresponding to 4 cell types ranging from healthy to late stage cancer cells. We use cross validation and additional experiments to assess if our models can predict the dynamics of protein activity under previously unseen perturbations.

Then we use our treatment evaluation formalism to specify characteristic properties of healthy cells and search for combinations of small molecule drugs (kinase inhibitors) with which the most number of properties can be satisfied on each of the diseased cell line models. Our results are valuable for guiding further experimentation, and could lead to a better understanding of changes in signaling during cancer progression and possible treatments.

This experimental study was conducted in collaboration with the Department of Systems Biology, Harvard Medical School, where all wet-lab experiments were performed.

5.5.1 Experimental data

To identify differences in signaling associated with tumor progression, experiments were conducted on four human liver cell types. Primary hepatocytes (HPH) are isolated from human donors and represent healthy liver cells. HHL5 cells are representative of immortalized but not overtly transformed cells [185]. HepG2 [186] and Focus [187] cell lines are both derived from hepatocarcinomas but Focus cells represent a higher pathological grade of cancer.

Measurements were conducted under multiple conditions. Cells were exposed to one of 5 ligands, in the presence or absence of one of 5 small molecule kinase inhibitors. Multiplex bead-based immuno-sandwich assays using Luminex technology (BioPlex assays; www.luminexcorp.com) were performed. This provided simultaneous quantitative measurements on the levels of modification of 12 signaling proteins under a total of 30 conditions (plus controls). The ligands and inhibitors used, and the measured proteins are listed in Table 5.2. Measurements of signaling activity were obtained $10, 30, 90$ and

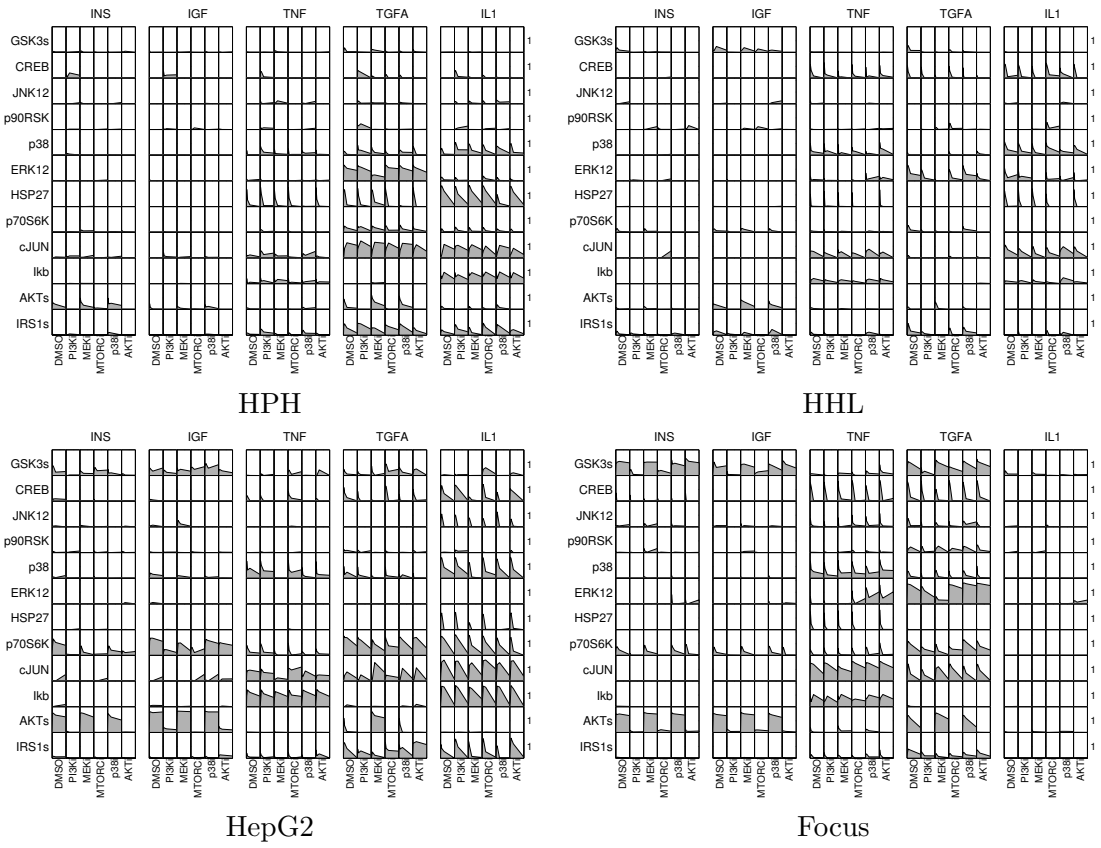


Figure 5.2: Experimental data on primary liver cells (HPH), an immortalized cell line (HHL5) and two transformed liver cancer cell lines (HepG2, Focus).

360 minutes after applying ligand stimulation. Previous studies have established the selectivity and linearity of these measurements [43]. The phosphorylation events assayed correspond to sites of activating modification and therefore serve as a surrogate for the level of activity of the signaling protein.

Data was normalized to the $[0, 1]$ range with respect to the no-treatment control and the measurements at the initial time point, following the methodology of [42]. The data organization and processing was carried out in the MATLAB toolbox DataRail, which was developed for high-throughput data management and integration [188]. Figure 5.2 shows the full experimental data set visualized with DataRail. The data consists of measurements on 4 cell types (plotted separately), across 5 ligand treatments (columns), combined with no inhibitor (DMSO) or one of 5 small molecule drug inhibitors (subcolumns). Each rectangle contains a filled time-course plot of measured activity for the initial time and 4 measurement time points of 12 proteins (rows).

Ligand	Description
INS	Insulin
IGF-1	Insulin related growth factor
TGFα	Transforming growth factor
TNFα	Tumor necrosis factor
IL-1α	Interleukin-1

Inhibited species	Inhibitor drug
AKT	MK2206
mTORc	AZD8055
PI3K	ZSTK474
p38	PHA818637
MEK	PD325901

Protein	Phosphorylation site
AKT	S473
cJUN	S63
CREB	S133
ERK1/2	T202/Y204, T185/Y187
GSK-3	S21/S9
HSP27	S78
Ikb	S32/S36
IRS-1	S636/S639
JNK	T183/Y185
p38	T180/Y182
p70S6K	T421/S424
p90RSK	T359/S363

Table 5.2: Ligands, inhibitors and measured proteins used to collect experimental data for the liver cancer study.

5.5.2 Prior knowledge network

A prior structure was first assembled from the GeneGo database containing a total of 29 nodes and 71 edges. The set of nodes includes 5 ligands, which only have outgoing edges, 12 proteins, whose phosphorylation is measured, and 12 hidden (unmeasured) nodes, out of which 4 can be inhibited in experiments using small molecule drugs (p38 is in the set of measured proteins but can also be inhibited). This prior structure is shown in Figure 5.3 (a).

We reduce the prior structure to a prior knowledge network (PKN), as defined in Section 5.3.1, by two transformation steps. First, we compress out hidden nodes as follows. The prior structure is a directed graph in which two nodes are connected if there is a sequence of directed edges (a path) between them. Starting from each ligand, we find paths on which the internal nodes are hidden nodes, until the first measured node is reached. We then add a direct edge from the ligand to this measured node. The same is done starting from each measured node, which has at least one edge to an unmeasured node. The signs on the new edges are set consistent with the signs along the path they replace. For instance, a path with two negative edges will be replaced by a positive edge, and a path with one negative and one positive edge with a negative one. Thereby, all hidden nodes are removed, and their effect is represented through the newly added edges. Second, we model inhibitors explicitly, as input nodes, with outgoing edges to the targets of the species they inhibit, but with opposite sign. For instance, MEKi (the inhibitor of MEK) will be modeled as an input node with outgoing

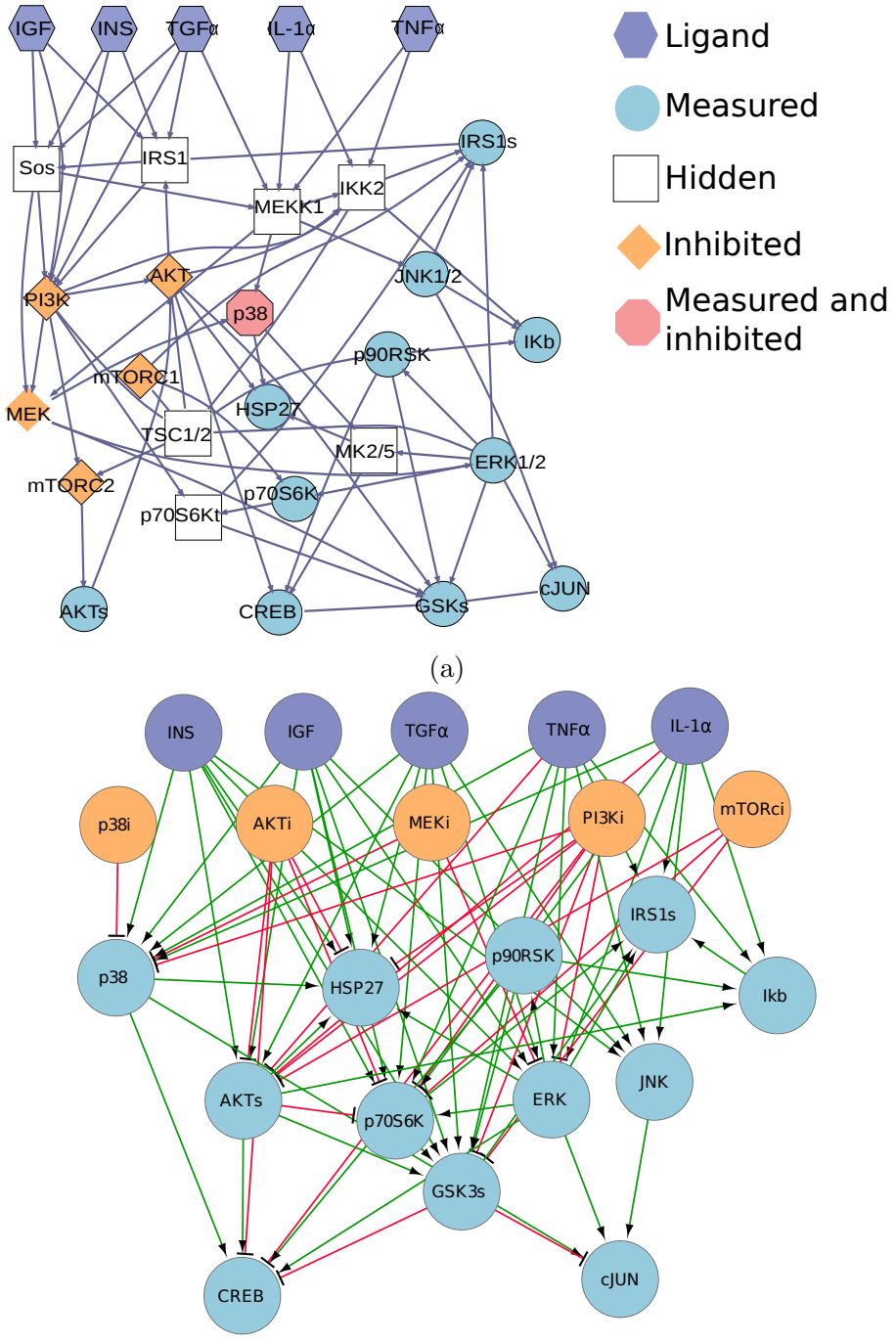


Figure 5.3: Prior knowledge network for signaling in liver cancer. (a) Prior structure extracted from databases. (b) Compressed PKN used as basis for DBN construction. Green arrows show activation, and red arrows inhibition.

inhibitory edges to p38, ERK and GSKs (since MEK had outgoing activation edges to p38, ERK and GSKs). Following this procedure, the resulting structure has 22 nodes, including 5 ligands, 5 inhibitors and 12 measured species. The total number of edges is 78. The PKN is shown in Figure 5.3 (b).

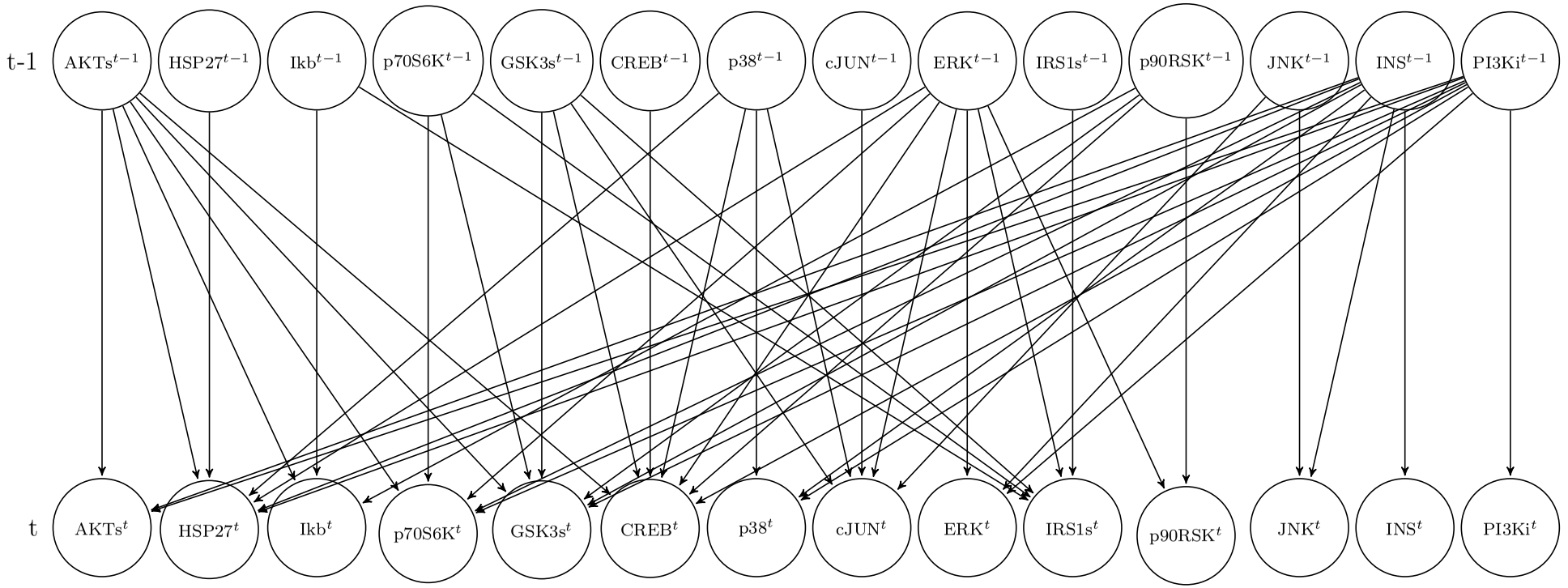


Figure 5.4: Two time slices of the DBN model structure for signaling in liver cancer, derived from the prior knowledge network. Only one of 5 ligands (INS) and one of 5 inhibitors (PI3Ki) is shown.

5.5.3 Model learning

Learning the DBN parameters involves solving one LP problem for each of 12 measured species at 4 time points. For the 12 proteins in the DBN we chose 3 discrete levels at $0, 0.5$ and 1, and for the nodes representing ligands and inhibitors we set two levels at 0 and 1. We added 1 intermediate time point between each measurement time (the correlation time $T_{\text{corr}} = 2$ for this pathway). Thus, the total number of LP problems solved is 96. For the 12 measured species in the pathway, the number of DBN parameters ranged between 9 and 6912, and the number of LP constraints were between 72 and 41532 (for p90RKS and p70S6K respectively). The time taken to learn the DBN was, on average 44.3 seconds for each cell line, on a 3.3GHz desktop computer. The LP problems were solved using the CPLEX (www.ibm.com) solver. We note that the LP problems at each time slice could also be solved in parallel, further speeding up learning.

5.5.4 Validation with test data

We first show that a DBN learned using our proposed approach has predictive ability. To do so, we performed cross validation as follows. Recall that the original data contains combinations of a ligand (one out of 5) combined with either no inhibitor or a single inhibitor (one out of 5). To imitate the existence of unmeasured combinations, we masked data for all species at all time points for 5 such combinations. Each run of cross validation uses a different set of masked cases. For instance, in the first run, the combinations INS+PI3Ki, IGF+MEKi, TNFα+MTORCi, TGFα+p38i and IL-1α+AKTi were masked, and the pairings were rotated to generate the remaining 4 cross validation runs. In each run, we used the remaining data as a training set, learned the corresponding DBN model, and then used inference on the resulting model to predict the masked data.

We chose the mean absolute deviation as a measure of prediction accuracy, motivated by the fact that DBN parameters were learned from the training data using the same distance measure. The mean absolute deviation in predicting the time course dynamics by the cell type specific DBNs was 0.064 for HPH, 0.041 for HHL5, 0.097 for HepG2 and 0.085 for FOCUS. These error rates are all below 10% of the possible range of predicted values, and are comparable to the magnitude of measurement noise in the data set. We also compared these values to predictions against scrambled data (the

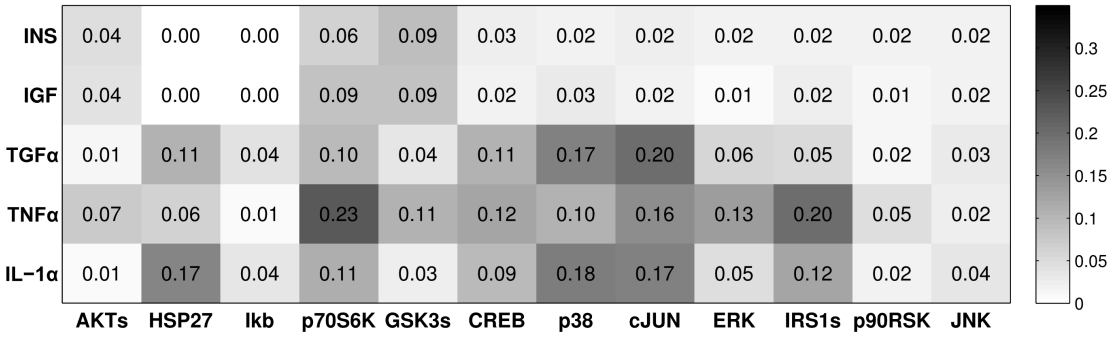


Figure 5.5: Prediction accuracy (mean absolute deviation) with respect to masked data for each species, under different ligand treatments. Values averaged across 5 cases, all cell lines and time points.

Cell type	DBN	Scrambled	Random
HPH	0.064	0.104 ($p < 10^{-12}$)	0.446
HHL5	0.041	0.071 ($p < 10^{-12}$)	0.474
HepG2	0.097	0.161 ($p < 10^{-16}$)	0.439
Focus	0.085	0.176 ($p < 10^{-30}$)	0.444

Table 5.3: Mean absolute deviation of estimates from the true data values for liver cell lines. The p-value of t-tests between the DBN predictions and the predictions with respect to scrambled data are also shown indicating the significance of the difference.

order of masked data values were randomly scrambled when comparing to predictions) and random predictions (uniform random predictions in the $[0, 1]$ interval). These results are shown in Table 5.3.

The prediction accuracy broken down by ligand and measured protein is shown in Figure 5.5. There is some variability across ligands and proteins, but in all cases the mean absolute deviation is below 0.25, and in most cases below 0.1. We also compared the accuracy of predictions by the time point of measurement (Figure 5.6, top). The early dynamics (10 and 30 minutes) are generally harder to predict, and better accuracy is seen in predicting the later dynamics (90 and 360 minutes).

Next we looked at the effect of the number of interpolation time points used. DBNs were learned with 0 to 4 interpolation time points; the prediction accuracies are shown in Figure 5.6 (bottom). There is only a slight increase in accuracy with an increasing number of interpolation time points, but the time needed to learn the model grows linearly with the number of interpolation times. Therefore the choice of 1 interpolation time point (equal to the correlation time of the DBN) is a reasonable choice.

Finally, we compared whether posing the constrained optimization problem using the L1 measure (linear programming, LP) or the L2 measure (quadratic programming,

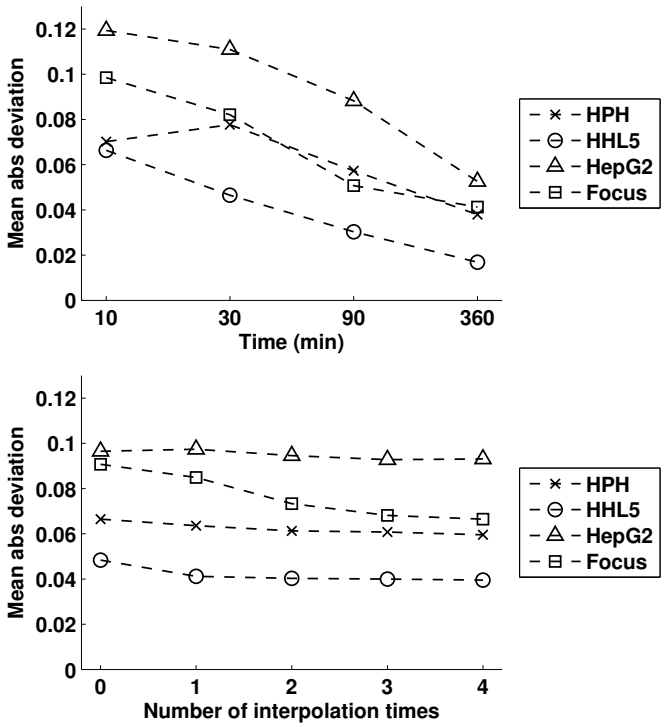


Figure 5.6: Prediction accuracy on cross validation data for data at different time points (a) and when learning with different number of interpolation time points (b)

QP) affects prediction accuracy. Using QP is more computationally intensive than LP, and therefore its use is only justified if it provides considerable improvements. The results in Figure 5.7 show that there is little difference between using LP or QP in terms of accuracy. In fact, the mean absolute deviation of predictions is better for only 1 out of 4 cell types with QP. When using the root mean square error to assess prediction accuracy (a measure more well suited for QP), there is only a very small change in the accuracy of LP and QP. Learning DBN parameters using QP took, on average, 330.8s for each cell line, making it around 7 times slower than the same procedure with LP. We conclude that the more efficient and scalable LP is a good choice for posing the constrained optimization problem used to learn DBN parameters.

5.5.5 Experimental validation

An important component of our work is predicting dynamics under combination of ligands and inhibitors that have not appeared in the training data set. In the previous section we assessed the performance of learned DBNs on predicting a masked portion of the training set. In this section we make use of additional experimental data measuring the effect of some further treatment combinations. These combinations are summa-

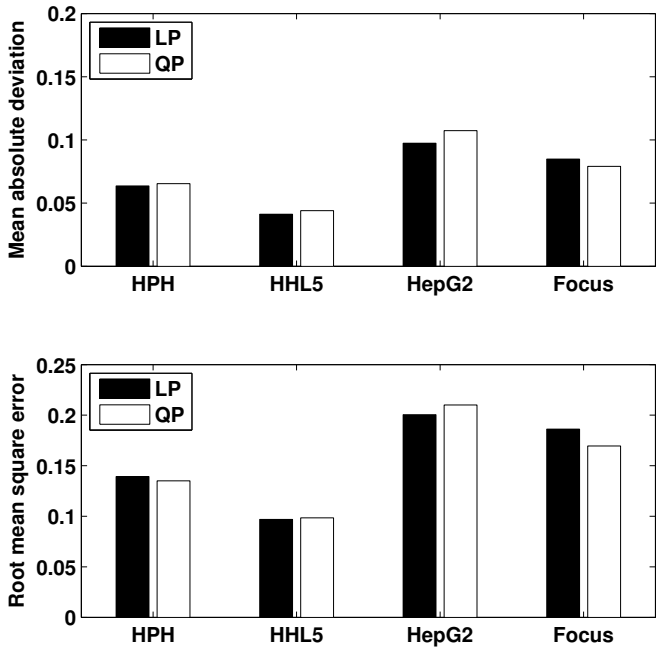


Figure 5.7: Comparison of prediction accuracy when solving the optimization with the L1 norm (linear programming, LP) and L2 norm (quadratic programming, QP).

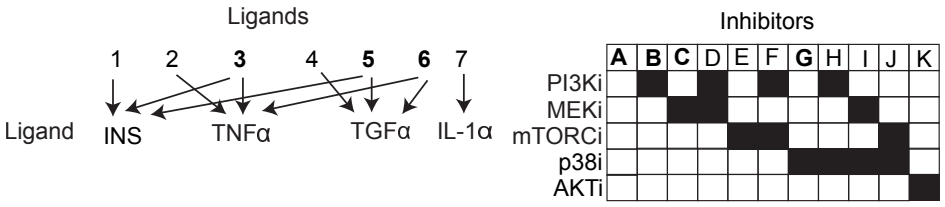


Figure 5.8: Structure of validation experiments on the HepG2 cell line. Ligand sets $1-7$ are measured combined with inhibitor sets A–K. Bold ligand sets are only combined with bold inhibitor sets. Other ligand sets are combined with all inhibitor sets.

rized in Figure 5.8. Excluding control measurements, this gives a total of 56 treatment conditions, some examples being TNFα + PI3Ki + MEKi and INS + TGFα + p38i.

The validation experimental data is limited, namely, it is only available for the HepG2 cell line, and at the single 30 minute time point. Further, the value of the fluorescence measurements cannot reliably be compared to those in the original training set. However, it is possible to determine whether a certain treatment combination resulted or did not result in the activation of a protein. Therefore we used the validation data in the following way. We normalized the validation data to the $[0, 1]$ range, taking into account control measurements, and then discretized it with the threshold set at 0.5 (with this choice, 23.5% of the data points were "active"). We used the cell type specific DBN model learned using the original training data set for the HepG2 cell line (see data in Figure 5.2). We simulated the DBN under all conditions appearing in the validation

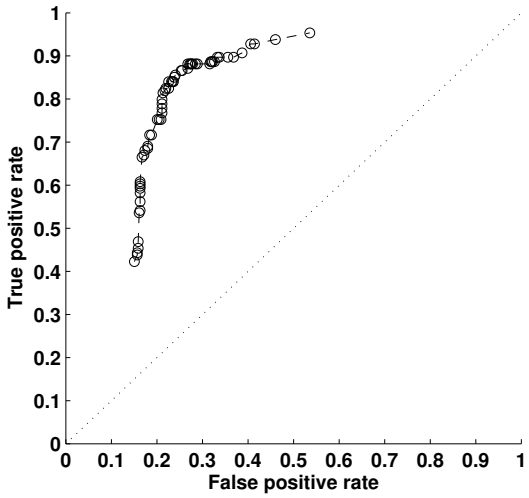


Figure 5.9: ROC curve of DBN predictions of protein phosphorylation under multiple combinations of ligands and inhibitors.

data set using the FF algorithm and recovered the state of the DBN corresponding to the 30 minute time point. The "activation" of a protein on the DBN was interpreted by requiring that the probability of the protein S taking a low value is smaller than a given threshold ($P(S = 0) \leq \gamma$, $\gamma \in (0, 1)$). This provides a binary prediction of activity, with which it is possible to calculate the true positive/negative and false positive/negative rates of the predictions with respect to the validation data. We chose a range of different thresholds between $\gamma = 0.01$ and 0.99 for the output of the DBN and drew a receiver operating characteristic (ROC) curve of the false positive and true positive rates (Figure 5.9). The ROC curve shows that DBN predictions maintain low false positive rates while reaching a high true positive rate, as the threshold is decreased.

5.5.6 Treatment evaluation

In the previous sections we showed that we can learn cell type specific DBN models for the 4 cell types of interest, and that these models have the capability to predict dynamics under treatment conditions not covered by training data.

In this section, we use the learned cell type specific models to evaluate treatment conditions and find ones that achieve specified dynamics. Our method for treatment evaluation and optimization introduced in Section 5.4 can be used with any specifications of interest expressed as PBL formulas. Here we will specify formulas based on the dynamics of protein phosphorylation in healthy cells (HPH). Our motivation in doing so is to find combinations of inhibitors, which when applied on the transformed cell lines

ID	Ligand	Property	Formula
φ_1	TNFα	cJUN stays at a low to medium level	$((\text{TNF}\alpha, v_H) \geq 1) \Rightarrow \mathbf{G}((\text{cJUN}, v_L) \geq 0.6)$
φ_2	TGFα	cJUN is activated and sustained	$((\text{TGF}\alpha, v_H) \geq 1) \Rightarrow \mathbf{F}((\text{cJUN}, v_H) \geq 0.2 \wedge \mathbf{FG}((\text{cJUN}, v_L) \leq 0.5))$
φ_3	IL-1α	cJUN is activated and sustained	$((\text{IL-1}\alpha, v_H) \geq 1) \Rightarrow \mathbf{F}((\text{cJUN}, v_H) \geq 0.2 \wedge \mathbf{FG}((\text{cJUN}, v_L) \leq 0.5))$
φ_4	TGFα	ERK is activated and sustained	$((\text{TGF}\alpha, v_H) \geq 1) \Rightarrow \mathbf{F}((\text{ERK}, v_H) \geq 0.2 \wedge \mathbf{FG}((\text{ERK}, v_L) \leq 0.5))$
φ_5	TGFα	HSP27 is transiently activated	$((\text{TGF}\alpha, v_H) \geq 1) \Rightarrow \mathbf{F}((\text{HSP27}, v_L) \leq 0.5 \wedge \mathbf{FG}((\text{HSP27}, v_L) \geq 0.8))$
φ_6	INS	GSK3s stays at a low level	$((\text{INS}, v_H) \geq 1) \Rightarrow \mathbf{G}((\text{GSK3s} = v_L) \geq 0.8)$
φ_7	IGF	GSK3s stays at a low level	$((\text{IGF}, v_H) \geq 1) \Rightarrow \mathbf{G}((\text{GSK3s} = v_L) \geq 0.8)$
φ_8	TGFα	GSK3s is transiently activated	$((\text{TGF}\alpha, v_H) \geq 1) \Rightarrow \mathbf{F}((\text{GSK3s}, v_M) \geq 0.25 \wedge \mathbf{FG}((\text{GSK3s}, v_L) \geq 0.8))$
φ_9	TNFα	CREB stays at a low level	$((\text{TNF}\alpha, v_H) \geq 1) \Rightarrow \mathbf{G}((\text{CREB} = v_L) \geq 0.8)$
φ_{10}	TGFα	CREB is transiently activated	$((\text{TGF}\alpha, v_H) \geq 1) \Rightarrow \mathbf{F}((\text{CREB}, v_M) \geq 0.25 \wedge \mathbf{FG}((\text{CREB}, v_L) \geq 0.8))$
φ_{11}	IL-1α	CREB stays at a low level	$((\text{IL-1}\alpha, v_H) \geq 1) \Rightarrow \mathbf{G}((\text{CREB} = v_L) \geq 0.8)$
φ_{12}	TNFα	p38 is transiently activated	$((\text{TNF}\alpha, v_H) \geq 1) \Rightarrow \mathbf{F}((\text{p38}, v_M) \geq 0.1 \wedge \mathbf{FG}((\text{p38}, v_L) \geq 0.8))$
φ_{13}	IL-1α	p38 is transiently activated	$((\text{IL-1}\alpha, v_H) \geq 1) \Rightarrow \mathbf{F}((\text{p38}, v_M) \geq 0.1 \wedge \mathbf{FG}((\text{p38}, v_L) \geq 0.8))$
φ_{14}	INS	AKTs activation is not sustained	$((\text{INS}, v_H) \geq 1) \Rightarrow \mathbf{FG}((\text{AKTs} = v_L) \geq 0.9)$
φ_{15}	IGF	AKTs activation is not sustained	$((\text{IGF}, v_H) \geq 1) \Rightarrow \mathbf{FG}((\text{AKTs} = v_L) \geq 0.9)$
φ_{16}	INS	p70S6K stays at a low level	$((\text{INS}, v_H) \geq 1) \Rightarrow \mathbf{G}((\text{p70S6K} = v_L) \geq 0.9)$
φ_{17}	IGF	p70S6K stays at a low level	$((\text{IGF}, v_H) \geq 1) \Rightarrow \mathbf{G}((\text{p70S6K} = v_L) \geq 0.9)$
φ_{18}	TNFα	p70S6K stays at a low level	$((\text{TNF}\alpha, v_H) \geq 1) \Rightarrow \mathbf{G}((\text{p70S6K} = v_L) \geq 0.9)$

Table 5.4: Formalized properties of protein phosphorylation dynamics on healthy liver cells.

(HHL5, HepG2, Focus), can modify some aspects of their dynamics to mimic those in healthy cells. We constructed a set of 18 properties, which each describe the dynamics of protein activity in HPH cells under some ligand stimulation. The properties we chose correspond to settings in which the dynamics in HPH cells is significantly different from other cell types. The list of properties is shown in Table 5.4.

First we verified that the properties φ_1 to φ_{18} are satisfied on the DBN model of healthy cells. We then used the models of the remaining 3 cell lines to make predictions. In this study, the choice in treatments is limited to 5 inhibitors, and each possible treatment condition is characterized by a subset of these inhibitors, which are applied on the cells (giving a total of 32 possible conditions). Despite the relatively small number of combinatorial treatments, experimentally evaluating these combinations with respect to the 18 properties would require the collection of an additional set of data roughly 3 times the size of the original data set. However, using our DBN models, verifying these

properties under all possible conditions in all cell lines is tractable (it took, on average, 35 seconds for each cell line). Therefore, in what follows, we present results based on an exhaustive evaluation of all possible inhibitor combinations.

We were interested in finding the combination of inhibitors with which the most properties are satisfied, based on the number of inhibitors used. Table 5.5 summarizes the best inhibitor combinations for each possible number of inhibitors. When no inhibitor is used, 3 properties are satisfied on the HHL5 cell line, 1 on HepG2 and 1 on Focus. When a single inhibitor is used a larger number of properties can be satisfied (8, 4 and 6) for HHL5, HepG2 and Focus, respectively. In each case, adding one more inhibitor can significantly increase the effectiveness of the treatment, and in each case the combination of PI3Ki and AKTi was found to be the best. This combination resulted in 11, 8 and 10 satisfied properties on HHL5, HepG2 and Focus, respectively.

In some cases we found that multiple different combinations can achieve the same number of satisfied properties. However, in general they will satisfy a different set of the properties. This is illustrated in Figure 5.10, where the subset of combinations corresponding to pairs of inhibitors (10 possible pairs) is shown. For each combination shaded rectangles indicate that a given property is satisfied. Figures showing the set of satisfied properties for all inhibitor combinations are given in Appendix C.

The PI3K/AKT/mTOR pathway can reduce apoptosis and promote proliferation, and is overactive in many cancers [189]. Our results specifically implicate PI3K and AKT as important targets, since inhibiting these kinases results in better resemblance to properties of healthy cells. We showed that the inhibition of either PI3K or AKT alone is less effective than their combined inhibition. Further, inhibiting PI3K and AKT is more effective than either one combined with mTOR inhibition.

Another important difference between healthy and diseased cells (expressed in property φ_5) is that heat shock protein (HSP27) response is triggered in healthy cells in response to stimulation by the growth factor TGFα, whereas the same activation is missing in diseased cell lines. The loss of heat shock response has been implicated as an important component of disease progression in liver cancer [190]. Our analysis shows that the transient activation of HSP27 in response to TGFα cannot be recovered through any combination of the studied kinase inhibitors. This implies that the lack of response could be due to the loss of an interaction between ERK and HSP27 during disease

HHL5

Number of inhibitors	Best combinations	Number of properties satisfied
0	–	3
1	PI3Ki or mTORci or AKTi	8
2	PI3Ki + AKTi	11
3	PI3Ki + AKTi + mTORci	11
4	PI3Ki + MEKi + mTORci + AKTi or PI3Ki + MEKi + p38i + AKTi or PI3Ki + mTORci + p38i + AKTi or	10
5	All	10

HepG2

Number of inhibitors	Best combinations	Number of properties satisfied
0	–	1
1	MEKi or PI3Ki	4
2	PI3Ki + AKTi	8
3	PI3Ki + MEKi + mTORci or PI3Ki + mTORci + p38i or MEKi + mTORci + AKTi	10
4	PI3Ki + MEKi + mTORci + p38i or PI3Ki + MEKi + mTORci + AKTi or PI3Ki + mTORci + p38i + AKTi or MEKi + mTORci + p38i + AKTi	10
5	All	10

Focus

Number of inhibitors	Best combinations	Number of properties satisfied
0	–	1
1	PI3Ki	6
2	PI3Ki + AKTi	10
3	PI3Ki + AKTi + mTORci	11
4	PI3Ki + AKTi + mTORci + p38i	11
5	All	10

Table 5.5: The best combinations of kinase inhibitors, shown by the number of inhibitors used, for each of 3 transformed liver cell lines.

progression, and inhibiting upstream kinases is unlikely to recover this interaction.

5.6 Summary

In this chapter we proposed a method for learning DBN models of pathway dynamics. Our method relies on solving a series of constrained optimization problems using linear programming to obtain the conditional probability parameters of the DBN. This way of learning parameters is scalable since the size of each LP problem only depends on the number of parents of a node, and not on the overall size of the model.

Next we showed that once the parameters of the DBN have been learned one can use inference algorithms to predict dynamics under various previously unseen conditions. This allows us to specify biologically relevant dynamical properties in a probabilistic temporal logic and find treatment conditions under which the properties are satisfied.

We applied our method to learn the dynamics of protein phosphorylation in signaling pathways of liver cancer cell lines. Validation experiments showed that we could reliably

predict dynamics under previously unseen combinations of ligand stimulation and kinase inhibition. We learned a DBN model for 4 cell lines ranging from healthy cells to late stage cancer cells. Then we specified 18 characteristic dynamical properties of dynamics in healthy cells and reported combinations of kinase inhibitors with which the diseased cell lines could match most of these properties.

An assumption we have made is that a prior knowledge network (PKN) is given, from which the DBN structure can be derived. Inferring model structure from limited experimental data is highly challenging and is likely to produce ambiguous results without strong prior constraints. A more realistic goal is to *refine* the structure derived from the PKN by removing unnecessary edges (edges whose existence does not contribute to explaining data). It turns out that our linear programming approach can be extended in a straightforward manner to solve this problem. Namely, one can introduce binary edge indicator variables, which are used to constrain parameters to be consistent with the existence or non-existence of an edge. Introducing a penalty for the overall number of parents of a node, a mixed integer linear programming (MILP) problem can be posed. Preliminary results show that using this approach the structure can be simplified significantly, while retaining predictive capability. The set of edges kept or removed in each cell type specific model can also reveal important differences between disease stages. We plan to further explore this line of work to gain insights about liver cancer progression.

Finally, we have considered treatment conditions as a subset of inputs that are held at a constant high level during the course of the experiment. In a sense we have explored here only *static* treatments. This is reasonable in our case study on liver cancer since the transient signaling response to stimuli is measured in a relatively short time frame (up to 360 minutes). For longer time scales, a more sophisticated *dynamic* treatment regime could be explored. Building models from experimental data representing signaling on longer time scales, and considering dynamical treatment conditions could open up many interesting possibilities. For instance, it would be possible to model the sequential application of cancer drugs, at potentially varying doses. The appropriate framework to find such treatments is likely to be control theory, and works including [191] and [192] could serve as useful pointers in this line of work.

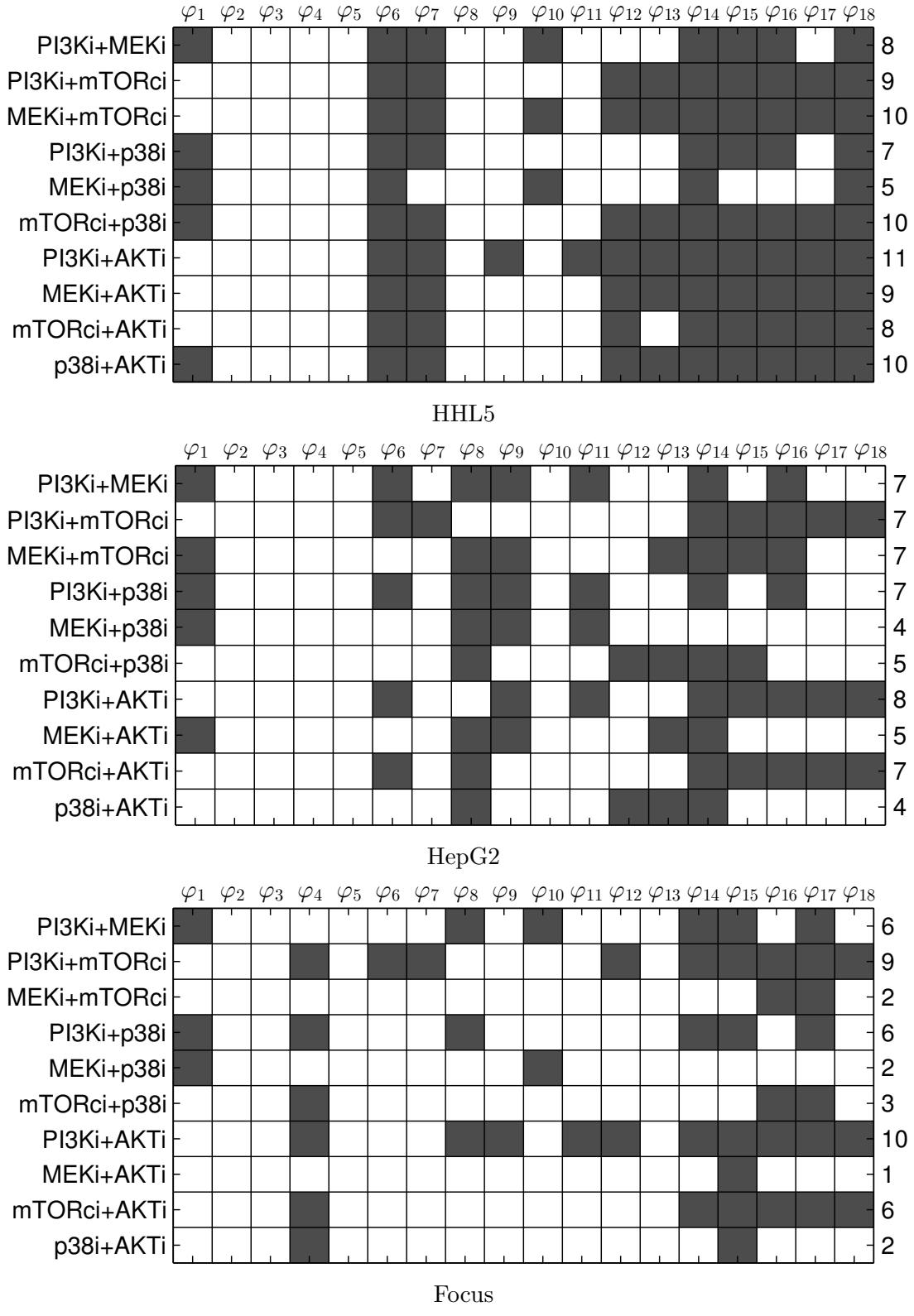


Figure 5.10: Effects of inhibitor combinations on 3 liver cancer cell lines with respect to characteristic dynamical properties of healthy cells (see Table 5.4). Shaded boxes indicate a satisfied property, and numbers on the right edge indicate the total number of satisfied properties for the given combination.

Chapter 6

Conclusion

The focus of this thesis was on modeling the dynamics of biological pathways. Our understanding of complex cellular processes including the cell cycle, programmed cell death, and the response to environmental stimuli can benefit greatly from quantitative computational models. However building such models based on prior domain knowledge and experimental data is challenging. This is partly due to many sources of uncertainty in the modeled biological system, the experimental data, and the modeling process. In the work presented here we used probabilistic approaches to deal with this uncertainty. Specifically, Chapters 3 and 4 addressed the problem of parameter uncertainty, which induces uncertainty in model predictions. Here a probabilistic approach allowed us to explicitly model and quantify the accuracy of predictions, taking into account prior knowledge and experimental data. In Chapter 5 dynamic Bayesian networks enabled us to model probabilistic relationships between species that do not physically interact with each other. This accounts for the uncertainty induced by unmodeled intermediate species and the stochasticity of dynamics. We now briefly review the main contributions of each chapter and discuss directions for future work.

In Chapter 3 we proposed kernel-enhanced particle filters to perform parameter inference on pathway models. Particle filters used before in this context suffer from particle collapse resulting in inaccurate predictions under parameter uncertainty. We showed that the basic particle filter extended with kernel steps can recover from particle collapse and give much more accurate predictions with a lower sample size. Making Bayesian computations more tractable could lead to the wider adoption of such techniques on realistic pathway models. The rich set of probabilistic analysis techniques in

a Bayesian framework will allow a faithful characterization of the uncertainty of models and the predictions made with them.

Next, we proposed a method for verifying properties of pathway models under Bayesian parameter uncertainty. This is important since certain dynamical properties can be robustly preserved even if parameter values are unconstrained, while other properties may not hold due to uncertainty. In a Bayesian probabilistic framework the posterior characterizes the distribution of possible model parameters according to their support from prior knowledge and experimental data. Sampling independently from the posterior is generally not possible, hence we proposed a Markov chain Monte Carlo scheme to collect a sequence of dependent samples. These samples can be used to decide with statistical guarantees whether the model meets a given temporal logic property with at least a certain probability.

A common point in Chapters 3 and 4 is the characterization of pathway models under Bayesian parameter uncertainty. It is increasingly recognized that predictions made using pathway models are subject to large variability depending on the amount and quality of experimental data [8, 118]. This uncertainty should be reflected in the way pathway models are published and deposited in databases. One solution would be to publish all experimental data with the model. However, this would still require subsequent users to perform computationally intensive parameter inference. It is therefore better to store and publish a representation of the parameter posterior distribution along with the model. The technical challenges involved, including how the high-dimensional posterior should be encoded efficiently and in a standardized format, are interesting questions for future research.

It would be useful to attach to a model a list of key properties that it possesses. These properties should be verifiable at all stages of model construction and dissemination [193]. For example, in a model of extrinsically triggered apoptosis, a property of interest could be "according to this model, cells die within 6 hours after treatment with 50ng/ml TRAIL". Temporal logics and model checking, as argued in Chapters 4 and 5, provide a useful framework for formalizing such statements and verifying them in a biological context. Attaching a set of such properties that hold with high probability, to models with Bayesian uncertainty, would enable their reliable dissemination and reuse. The framework presented in Chapter 4 could be the basis for such a solution, and this could

bring a new paradigm in the way pathway models are shared and updated.

The goal of Chapter 5 was to learn the dynamics of signaling pathways using dynamic Bayesian networks. We developed a framework on the DBN to find treatment conditions under which the pathway shows specified, desirable dynamics. This is especially useful when searching for combinatorial treatments, which is very expensive experimentally. We applied our DBN learning method to model signaling in 4 cell types representing a range of stages in the progression towards liver cancer. We were able to find promising combinations of kinase inhibitors, using which some aspects of the dynamics of diseased cells can be driven to mimic those of healthy cells.

The experimental data we used consists of joint measurements of the phosphorylation of multiple proteins under a range of ligand stimuli and perturbations. Similar data sets have recently been collected for several melanoma [194] and breast cancer [195] cell lines. However, dynamical models that could predict the response of different cell lines to treatments have not been built. It will be interesting to apply our DBN learning method to these and other similar data sets. More generally, our method is a step in the direction from descriptive towards dynamic and predictive models of disease specific signaling behavior.

The concept of learning cell type specific models, as we have done using our DBNs, opens up interesting possibilities in personalized treatments. If predictive models are learned for several sub-types of a disease, one will be able to make model based predictions of what treatment (or combination of treatments) is most likely to yield a desired outcome in a specific setting. Quantitative dynamical models which can make reliable predictions despite uncertainty will be elemental in reaching these goals.

Appendix A

Supplementary information for Chapter 3

A.1 Enzyme-substrate model

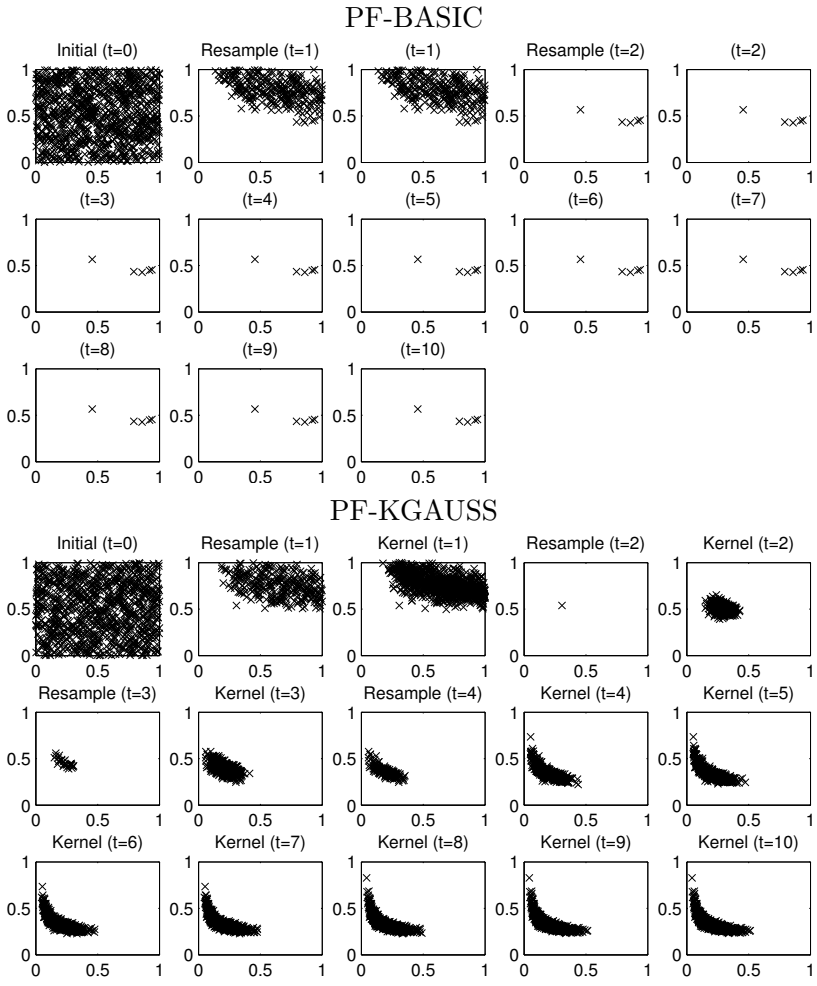


Figure A.1: Particles projected on the plane of k_1 against k_3 at each step of the filter for the enzyme-substrate model. PF-BASIC cannot converge to the high-probability region of parameter space, but PF-KGAUSS efficiently moves there by using kernel steps.

A.2 JAK-STAT model

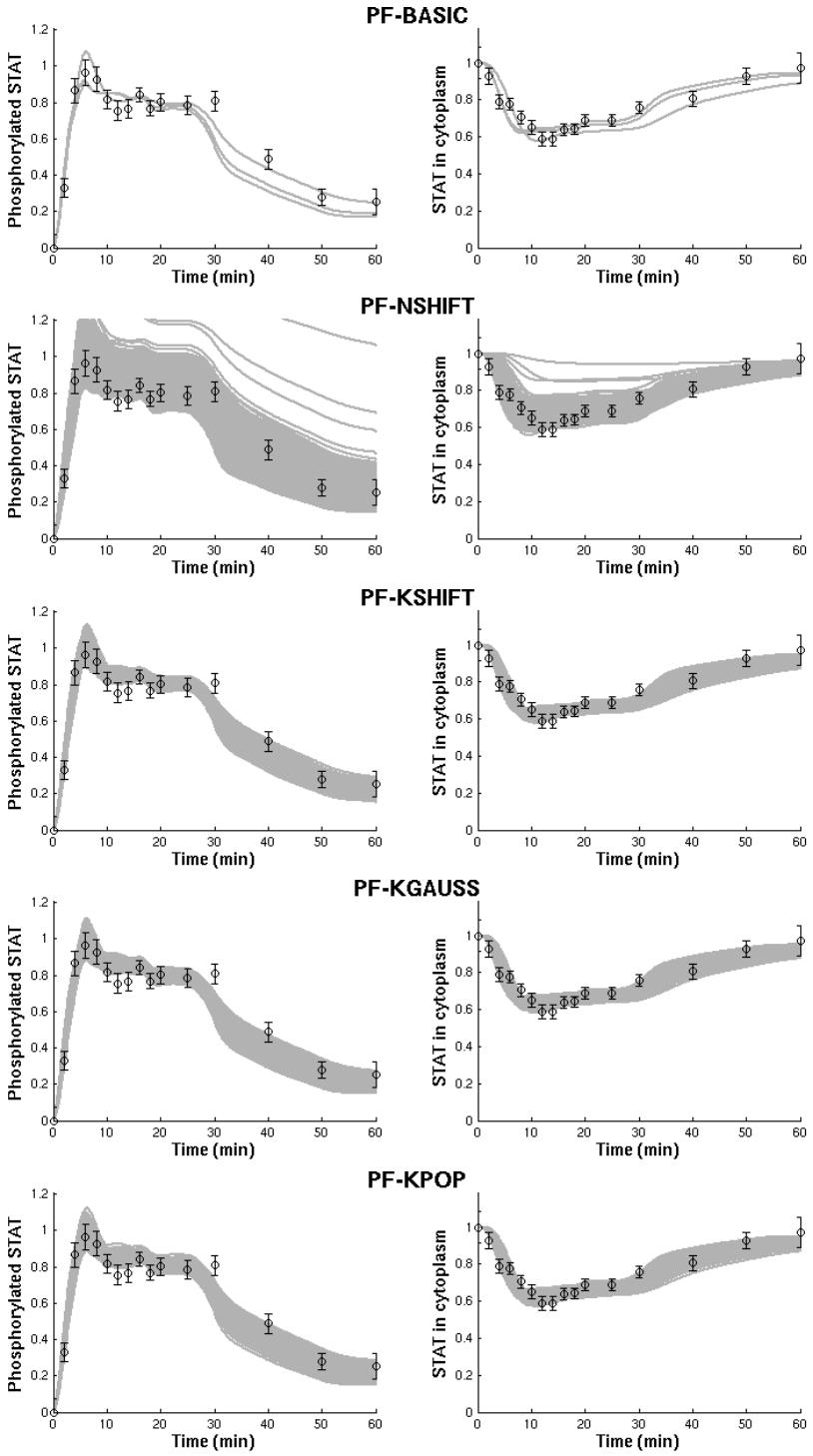


Figure A.2: Fit to experimental data with 1000 particles with different particle filter methods on the JAK-STAT pathway.

Appendix B

Supplementary information for Chapter 4

B.1 Spectral gap of the Markov chain

In Section 4.5.1 we described how the spectral gap of the Markov chain can be estimated in practice. Our Markov chains are defined on a general state space, and their transition kernel is described by a linear operator. Here we give a more precise definition of the spectral gap in this setting.

We call a Markov chain $X_1, X_2, \ldots$ on state space Ω with transition kernel $P(x, dy)$ *reversible* if there exists a probability measure π on Ω satisfying the detailed balance condition,

$$\pi(dx)P(x, dy) = \pi(dy)P(y, dx) \text{ for every } x, y \in \Omega. \tag{B.1}$$

Define $L_2(\pi)$ as the Hilbert space of complex valued measurable functions that are square integrable with respect to π, endowed with the inner product $(f, g) = \int fg^* \, \mathrm{d}\pi$. P can be then viewed as a linear operator on $L_2(\pi)$, denoted by $\boldsymbol{P}$, defined as

$$(\boldsymbol{P}f)(x) := \mathbb{E}_{P(x,\cdot)}(f),$$

and reversibility is equivalent to the self-adjointness of $\boldsymbol{P}$. The operator $\boldsymbol{P}$ acts on measures to the left, i.e. for every measurable subset A of Ω,

$$(\mu\boldsymbol{P})(A) := \int_{x\in\Omega} P(x, A)\mu(\mathrm{d}x).$$

For a Markov chain with stationary distribution π, we define the chain's *spectrum* as

$$S_2 := \{\lambda \in \mathbb{C} \setminus 0 : (\lambda \boldsymbol{I} - \boldsymbol{P})^{-1} \text{ does not exist as a bounded linear operator on } L_2(\pi)\}.$$

For reversible chains, S_2 lies on the real line. We now define the spectral gap as follows.

Definition B.1. *The* spectral gap *for reversible chains is defined as*

$$\gamma := 1 - \sup\{\lambda : \lambda \in S_2, \lambda \neq 1\} \quad \textit{if eigenvalue 1 has multiplicity 1,}$$

$$\gamma := 0 \quad \textit{otherwise.}$$

In the case of non-reversible chains, [196] defines the pseudo-spectral gap, and shows that it has similar properties as the spectral gap has for reversible chains.

B.2 EARM1.3 model

Here we show the dynamics of species appearing in the specified temporal logic properties in Section 4.6.2. The plots show the simulated trajectories with respect to the parameter posterior. The shaded area shows the 90 percentile of all trajectories (the quantiles are calculated at each time point) simulated with each parameter collected by the MCMC procedure. The dashed lines show the 50 percentile (median) of all trajectories.

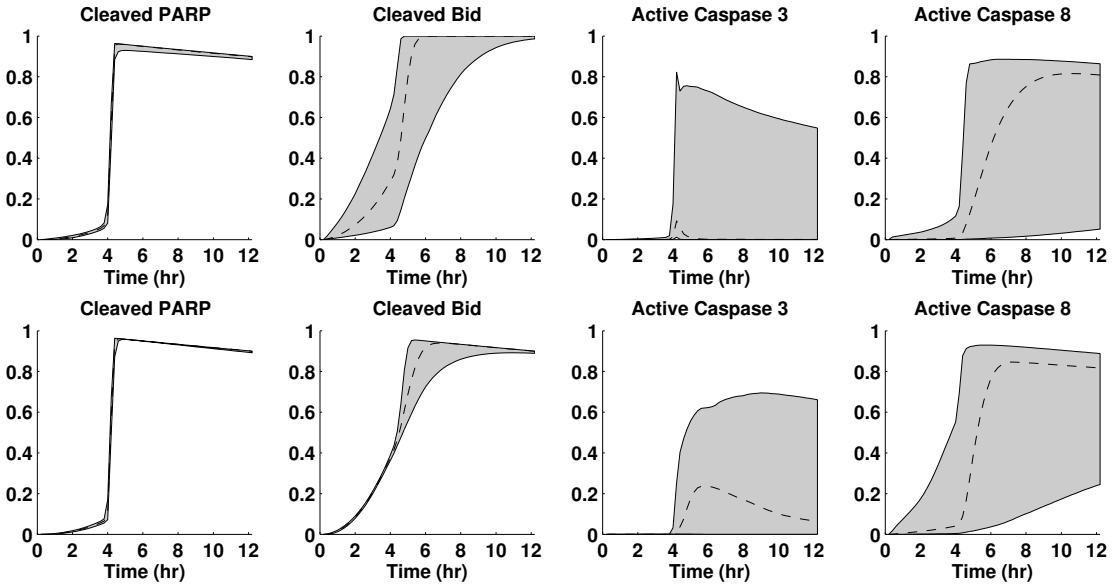


Figure B.1: Simulated trajectories (shaded area between 5 and 95 percentile) with respect to the parameter posterior when using only EC-RP data (top), and both EC-RP and IC-RP data (bottom).

Appendix C

Supplementary information for Chapter 5

C.1 DBN model of signaling in liver cancer

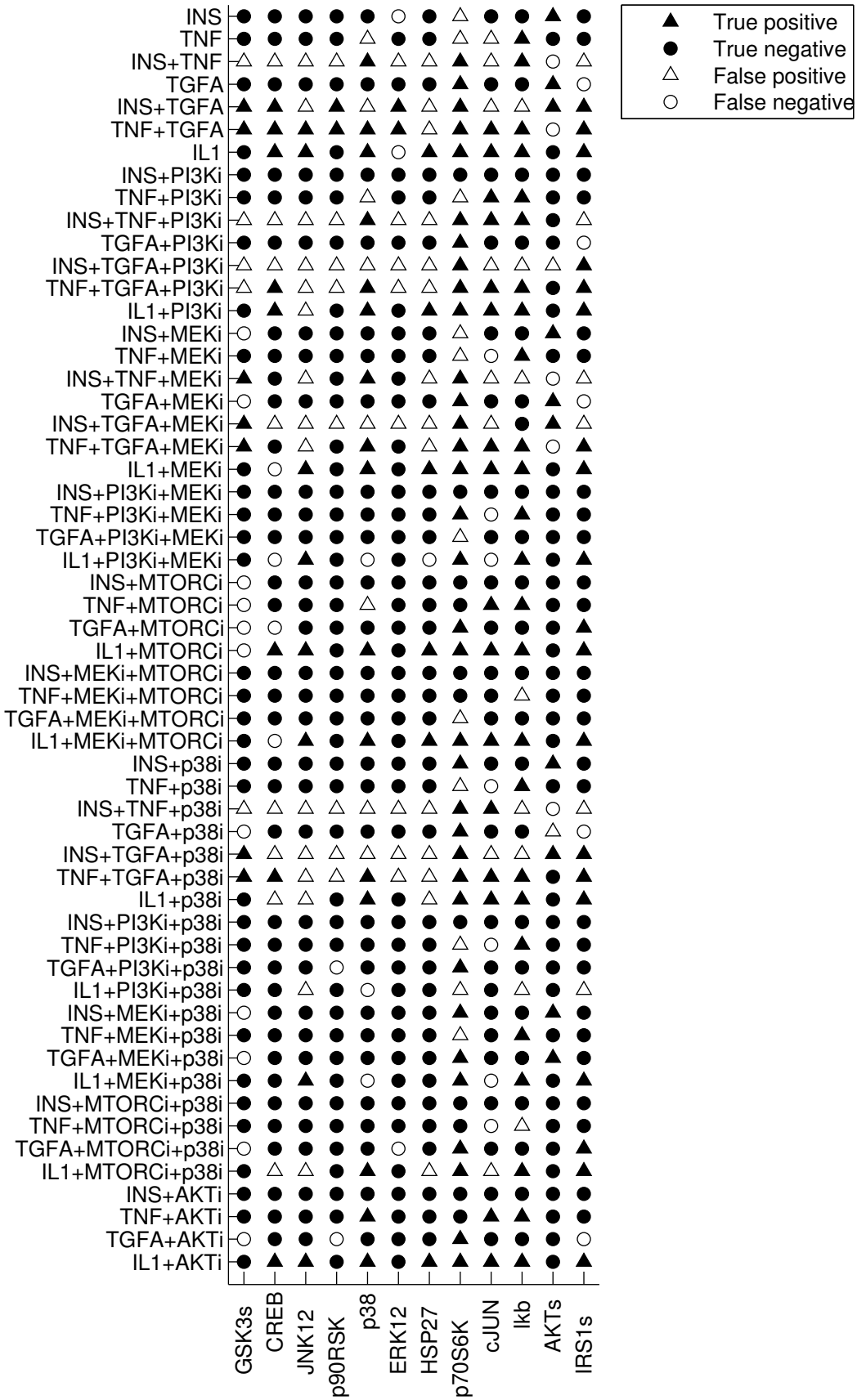


Figure C.1: Predictions by the DBN on additional experiments for the HepG2 cell line. Measurements indicate activity 30 minutes after ligand addition.

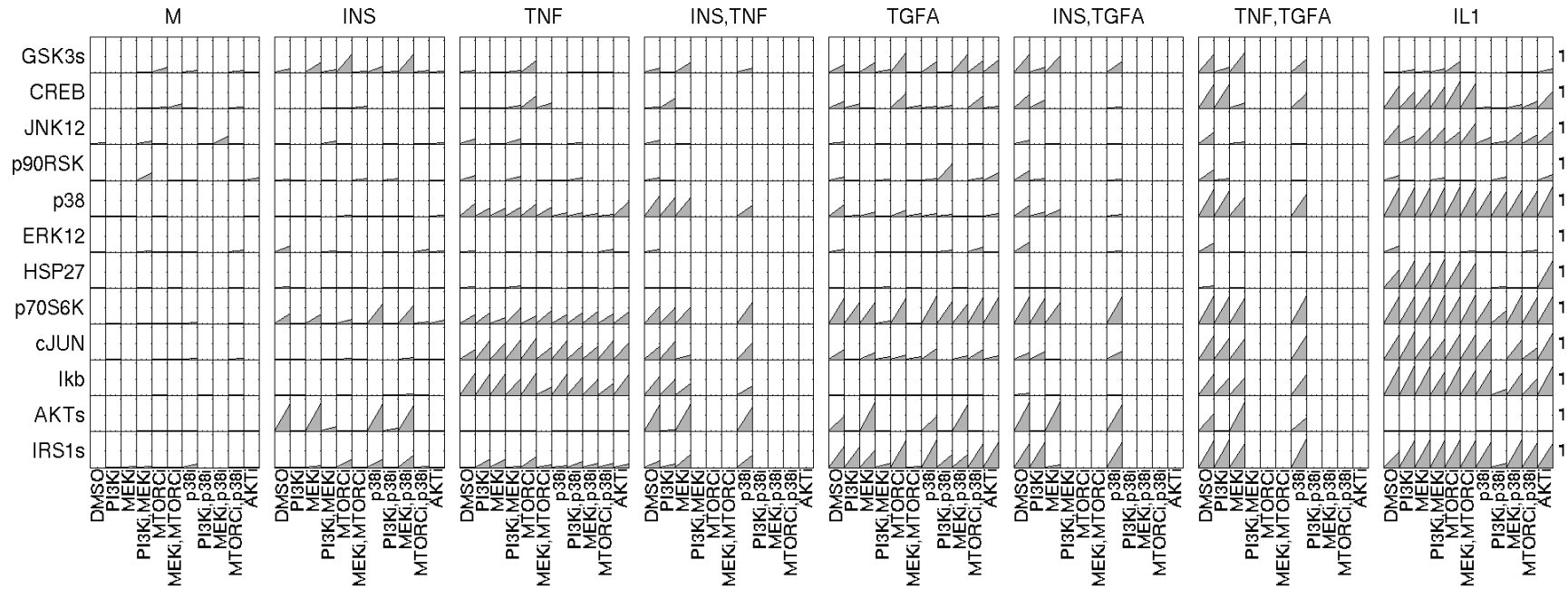


Figure C.2: Validation experiments for the HepG2 cell line, measuring phosphorylation at the 30 minute time point for 56 conditions.

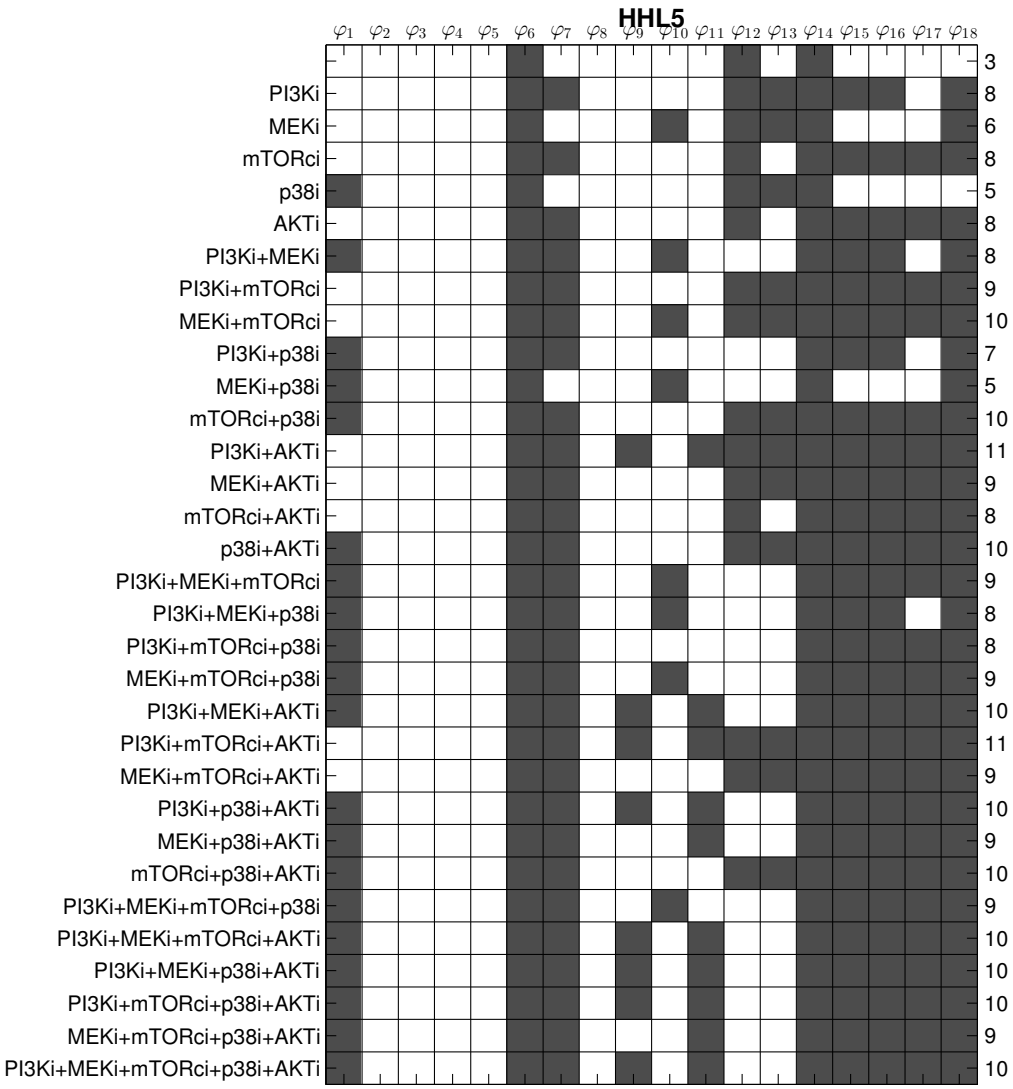


Figure C.3: Effects of all inhibitor combinations on HHL5 cells with respect to characteristic dynamical properties of healthy cells (see Table 5.4). Shaded boxes indicate a satisfied property, and numbers on the right edge indicate the total number of satisfied properties for the given combination.

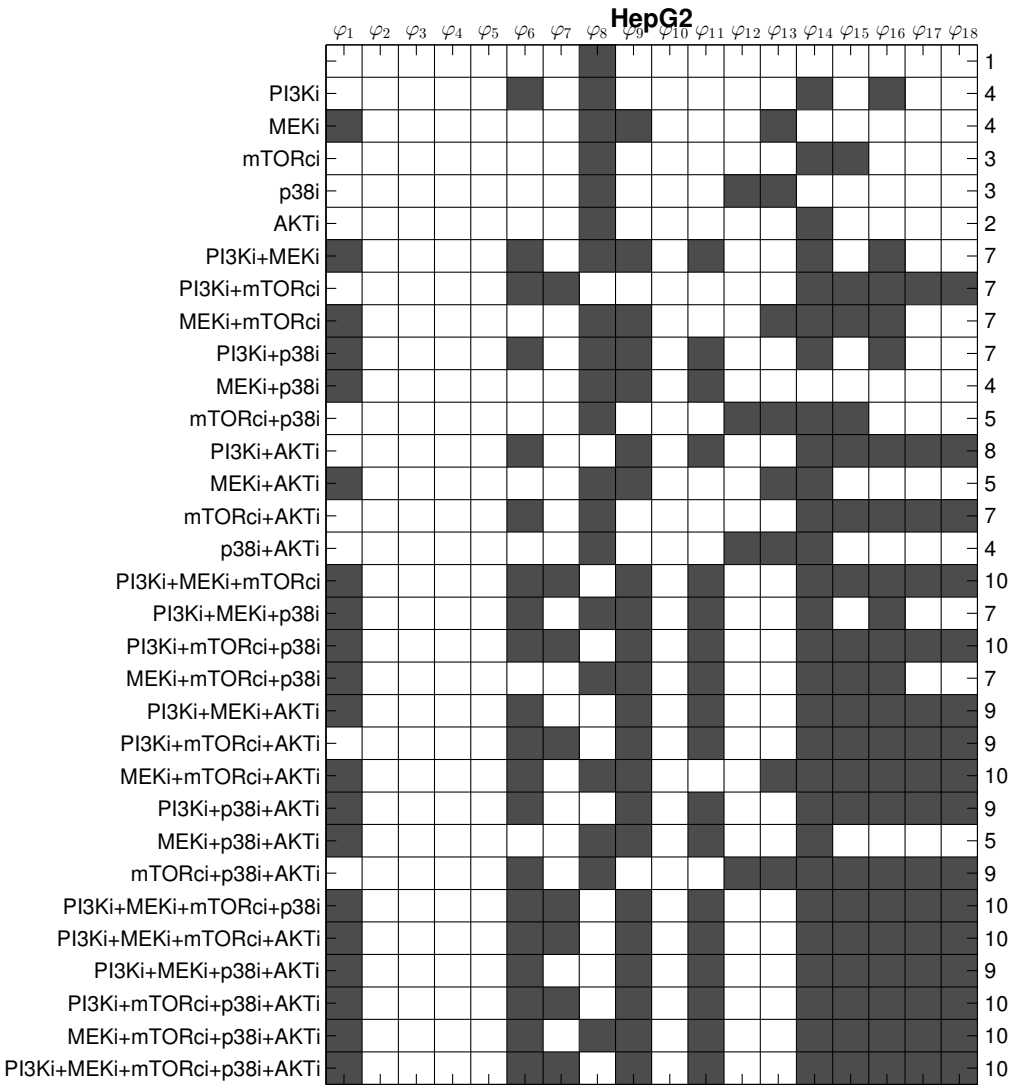


Figure C.4: Effects of all inhibitor combinations on HepG2 cells with respect to characteristic dynamical properties of healthy cells (see Table 5.4). Shaded boxes indicate a satisfied property, and numbers on the right edge indicate the total number of satisfied properties for the given combination.

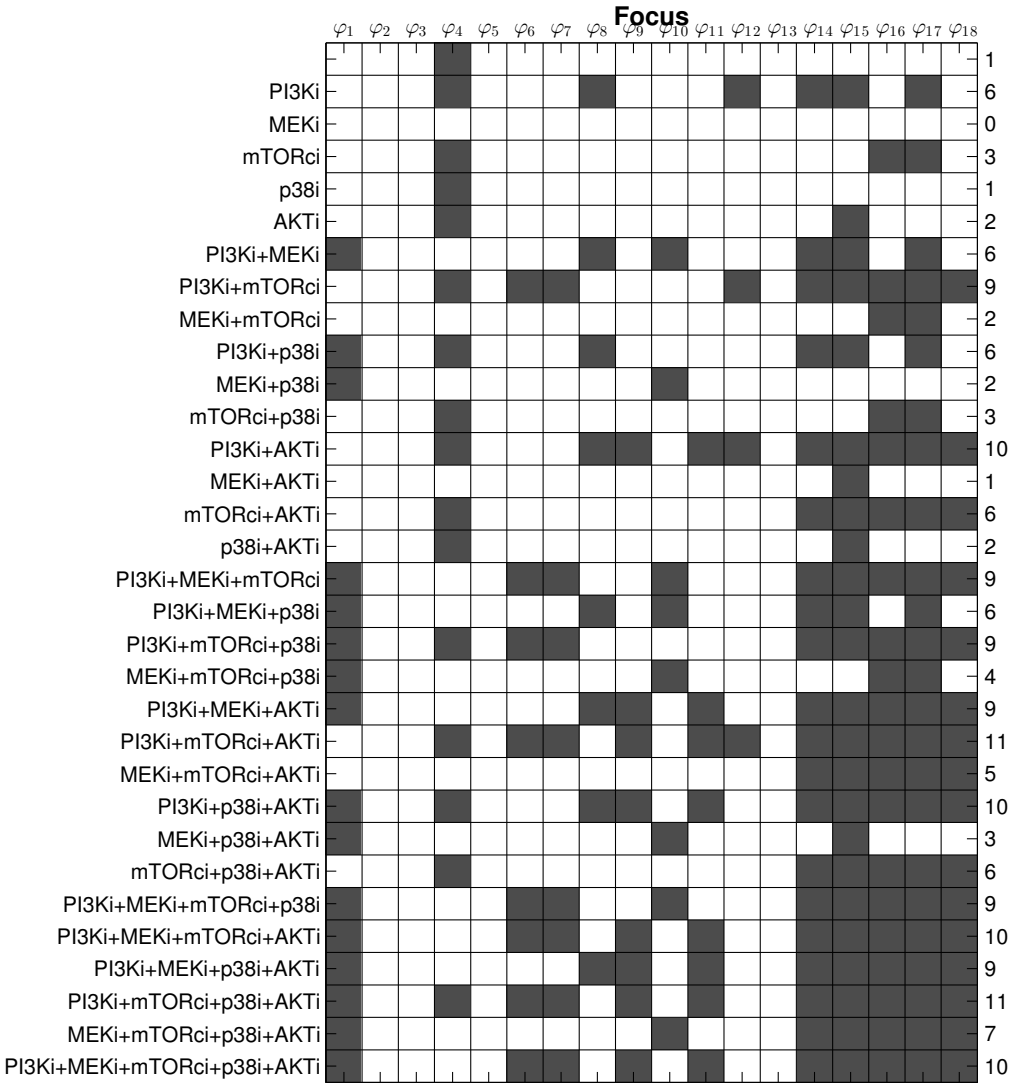


Figure C.5: Effects of all inhibitor combinations on Focus cells with respect to characteristic dynamical properties of healthy cells (see Table 5.4). Shaded boxes indicate a satisfied property, and numbers on the right edge indicate the total number of satisfied properties for the given combination.

Bibliography

[1] H. Kitano. Systems biology: a brief overview. *Science*, 295(5560):1662–1664, 2002.

[2] Jeremy E Purvis, Kyle W Karhohs, Caroline Mock, Eric Batchelor, Alexander Loewer, and Galit Lahav. p53 dynamics control cell fate. *Science*, 336(6087):1440–1444, 2012.

[3] Ioannis Lestas, Glenn Vinnicombe, and Johan Paulsson. Fundamental limits on the suppression of molecular fluctuations. *Nature*, 467(7312):174–178, 2010.

[4] B.B. Aldridge, J.M. Burke, D.A. Lauffenburger, and P.K. Sorger. Physicochemical modelling of cell signalling pathways. *Nature Cell Biology*, 8(11):1195–1203, 2006.

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

[6] Andreas Raue, Clemens Kreutz, Thomas Maiwald, Julie Bachmann, Marcel Schilling, Ursula Klingmüller, and Jens Timmer. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, 25(15):1923–1929, 2009.

[7] Kevin S Brown and James P Sethna. Statistical mechanical approaches to models with many poorly known parameters. *Physical Review E*, 68(2):021904, 2003.

[8] R.N. Gutenkunst, J.J. Waterfall, F.P. Casey, K.S. Brown, C.R. Myers, and J.P. Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLoS Computational Biology*, 3(10):e189, 2007.

[9] Mark Girolami, Ben Calderhead, and Vladislav Vyshemirsky. System identification and model ranking: The Bayesian perspective. In Neil D. Lawrence, Mark Girolami, Magnus Rattray, and Guido Sanguinetti, editors, *Learning and inference in computational systems biology*. The MIT Press, 2010.

[10] N. Friedman. Inferring cellular networks using probabilistic graphical models. *Science Signaling*, 303(5659):799–805, 2004.

[11] K. Sachs, O. Perez, D. Pe'er, D.A. Lauffenburger, and G.P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science Signalling*, 308(5721):523–529, 2005.

[12] Steven M. Hill, Yiling Lu, Jennifer Molina, Laura M. Heiser, Paul T. Spellman, Terence P. Speed, Joe W. Gray, Gordon B. Mills, and Sach Mukherjee. Bayesian inference of signaling network topology in a cancer cell line. *Bioinformatics*, 2012.

[13] C.H. Koh, M. Nagasaki, A. Saito, L. Wong, and S. Miyano. Da 1.0: parameter estimation of biological pathways using data assimilation approach. *Bioinformatics*, 26(14):1794–1796, 2010.

[14] B. Gyori and D. Paulin. Non-asymptotic confidence intervals for MCMC in practice. *arXiv preprint*, 2012.

[15] B. Liu, PS Thiagarajan, and D. Hsu. Probabilistic approximations of ODEs based bio-pathway dynamics. *Theoretical Computer Science*, 2011.

[16] Hoda Eydgahi, William W Chen, Jeremy L Muhlich, Dennis Vitkup, John N Tsitsiklis, and Peter K Sorger. Properties of cell death models calibrated and compared using Bayesian approaches. *Molecular systems biology*, 9(1), 2013.

[17] Russell N. Van Gelder. Circadian pathway, science signaling (connections map in the database of cell signaling, `http://stke.sciencemag.org/cgi/cm/stkecm;CMP_12992`).

[18] B.N. Kholodenko. Untangling the signalling wires. *Nature Cell Biology*, 9(3):247–249, 2007.

[19] J. Fisher and T.A. Henzinger. Executable cell biology. *Nature Biotechnology*, 25(11):1239–1249, 2007.

[20] Ludwig Von Bertalanffy. General system theory. *General systems*, 1(1):11–17, 1956.

[21] H. Kitano et al. Computational systems biology. *Nature*, 420(6912):206–210, 2002.

[22] W.W. Chen, B. Schoeberl, P.J. Jasper, M. Niepel, U.B. Nielsen, D.A. Lauffenburger, and P.K. Sorger. Input-output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data. *Molecular systems biology*, 5(1), 2009.

[23] K.S. Brown, C.C. Hill, G.A. Calero, C.R. Myers, K.H. Lee, J.P. Sethna, and R.A. Cerione. The statistical mechanics of complex signaling networks: nerve growth factor signaling. *Physical Biology*, 1:184, 2004.

[24] SV Sabau, S. Hashimoto, Y. Nemoto, and S. Ihara. Cell simulation for circadian rhythm based on Michaelis-Menten model. *Journal of Biological Physics*, 28(3):465–469, 2002.

[25] T. Kalmar, C. Lim, P. Hayward, S. Muñoz-Descalzo, J. Nichols, J. Garcia-Ojalvo, and A.M. Arias. Regulated fluctuations in nanog expression mediate cell fate decisions in embryonic stem cells. *PLoS Biology*, 7(7):e1000149, 2009.

[26] I.H. Segel. *Enzyme kinetics: behavior and analysis of rapid equilibrium and steady state enzyme systems*. Wiley New York:, 1975.

[27] M.W. Hirsch, S. Smale, and R.L. Devaney. *Differential equations, dynamical systems, and an introduction to chaos*. Academic Press, 2012.

[28] John C Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2008.

[29] John Denholm Lambert. *Numerical methods for ordinary differential systems: the initial value problem*. John Wiley & Sons, Inc., 1991.

[30] L Petzold and A Hindmarsh. LSODA (Livermore solver of ordinary differential equations). *Computing and Mathematics Research Division, Lawrence Livermore National Laboratory, Livermore, CA*, 24, 1997.

[31] Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

[32] J.J. Hughey, T.K. Lee, and M.W. Covert. Computational modeling of mammalian signaling networks. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 2(2):194–209, 2010.

[33] Kouichi Takahashi, Satya Nanda Vel Arjunan, and Masaru Tomita. Space in systems biology of signaling pathways–towards intracellular molecular crowding in silico. *FEBS letters*, 579(8):1783–1788, 2005.

[34] C. Lemerle, B. Di Ventura, and L. Serrano. Space as the final frontier in stochastic simulations of biological systems. *FEBS letters*, 579(8):1789–1794, 2005.

[35] D.T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188(1-3):404–425, 1992.

[36] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[37] Darren J Wilkinson. *Stochastic modelling for systems biology*. CRC press, 2011.

[38] A. Golightly and D.J. Wilkinson. Bayesian inference for stochastic kinetic models using a diffusion approximation. *Biometrics*, 61(3):781–788, 2005.

[39] A. Golightly and D.J. Wilkinson. Bayesian sequential inference for nonlinear multivariate diffusions. *Statistics and Computing*, 16(4):323–338, 2006.

[40] T.R. Kiehl, R.M. Mattheyses, and M.K. Simmons. Hybrid simulation of cellular behavior. *Bioinformatics*, 20(3):316, 2004.

[41] H. Matsuno, Y. Tanaka, H. Aoshima, A. Doi, M. Matsui, S. Miyano, et al. Biopathways representation and simulation on hybrid functional Petri net. *In Silico Biology*, 3(3):389–404, 2003.

[42] J. Saez-Rodriguez, L.G. Alexopoulos, M.S. Zhang, M.K. Morris, D.A. Lauffenburger, and P.K. Sorger. Comparing signaling networks between normal and transformed hepatocytes using discrete logical models. *Cancer research*, 71(16):5400–5411, 2011.

[43] L.G. Alexopoulos, J. Saez-Rodriguez, B.D. Cosgrove, D.A. Lauffenburger, and P.K. Sorger. Networks inferred from biochemical data reveal profound differences in toll-like receptor and inflammatory signaling between normal and transformed hepatocytes. *Molecular & Cellular Proteomics*, 9(9):1849–1865, 2010.

[44] Kevin A Janes and Michael B Yaffe. Data-driven modelling of signal-transduction networks. *Nature Reviews Molecular Cell Biology*, 7(11):820–828, 2006.

[45] Adrien Fauré, Aurélien Naldi, Claudine Chaouiya, and Denis Thieffry. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–e131, 2006.

[46] Camille Terfve, Thomas Cokelaer, David Henriques, Aidan MacNamara, Emanuel Goncalves, Melody K Morris, Martijn van Iersel, Douglas A Lauffenburger, and Julio Saez-Rodriguez. CellNOptR: a flexible toolkit to train protein signaling networks to data using multiple logic formalisms. *BMC Systems Biology*, 6(1):133, 2012.

[47] Melody K Morris, Julio Saez-Rodriguez, David C Clarke, Peter K Sorger, and Douglas A Lauffenburger. Training signaling pathway maps to biochemical data with constrained fuzzy logic: quantitative analysis of liver cell responses to inflammatory stimuli. *PLoS Computational Biology*, 7(3):e1001099, 2011.

[48] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[49] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[50] K.P. Murphy. *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, University of California, 2002.

[51] J. Saez-Rodriguez, L.G. Alexopoulos, J. Epperlein, R. Samaga, D.A. Lauffenburger, S. Klamt, and P.K. Sorger. Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Molecular systems biology*, 5(1), 2009.

[52] P. Mendes. Framework for comparative assessment of parameter estimation and inference methods in systems biology. In N.D. Lawrence, M. Girolami, and M. Rattray, editors, *Learning and inference in computational systems biology*. The MIT Press, 2010.

[53] M. Ashyraliyev, Y. Fomekong-Nanfack, J.A. Kaandorp, and J.G. Blom. Systems biology: parameter estimation for biochemical models. *FEBS Journal*, 276(4):886–902, 2009.

[54] Robert Hooke and T A_ Jeeves. "Direct search" solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961.

[55] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963.

[56] T.P. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on*, 4(3):284–294, 2000.

[57] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[58] C G Moles, P Mendes, and J R Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Research*, 13(11):2467–2474, 2003.

[59] Neil D. Lawrence, Mark Girolami, Magnus Rattray, and Guido Sanguinetti. *Learning and Inference in Computational Systems Biology*. MIT Press, 2009.

[60] Darren James Wilkinson. *Stochastic modelling for systems biology*, volume 44. CRC press, 2012.

[61] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov chain Monte Carlo in practice*. Interdisciplinary Statistics. Chapman & Hall, London, 1996.

[62] Z. Chen. Bayesian filtering: From Kalman filters to particle filters, and beyond. *Adaptive Systems Lab., McMaster University., Hamilton, Ontario, Canada. Internet: http://users. isr. ist. utl. pt/˜ jpg/tfc0607/chen_bayesian. pdf*, 2003.

[63] Hagai Attias. Inferring parameters and structure of latent variable models by variational Bayes. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 21–30. Morgan Kaufmann Publishers Inc., 1999.

[64] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael PH Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202, 2009.

[65] John J Tyson, Kathy Chen, and Bela Novak. Network dynamics and cell physiology. *Nature Reviews Molecular Cell Biology*, 2(12):908–916, 2001.

[66] Enrico Di Cera, Paul E Phillipson, and Jeffries Wyman. Limit-cycle oscillations and chaos in reaction networks subject to conservation of mass. *Proceedings of the National Academy of Sciences*, 86(1):142–146, 1989.

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

[68] Amir Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57. IEEE, 1977.

[69] Armin Biere, Alessandro Cimatti, Edmund M Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. *Advances in computers*, 58:117–148, 2003.

[70] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.

[71] M. Kwiatkowska, G. Norman, and D. Parker. Using probabilistic model checking in systems biology. *ACM SIGMETRICS Performance Evaluation Review*, 35(4):14–21, 2008.

[72] John Heath, Marta Kwiatkowska, Gethin Norman, David Parker, and Oksana Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 391(3):239–257, 2008.

[73] Grégory Batt, Calin Belta, and Ron Weiss. Temporal logic analysis of gene networks under parameter uncertainty. *Automatic Control, IEEE Transactions on*, 53(Special Issue):215–229, 2008.

[74] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: An overview. In *Runtime Verification*, pages 122–135. Springer, 2010.

[75] Håkan LS Younes and Reid G Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Information and Computation*, 204(9):1368–1409, 2006.

[76] Abraham Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[77] Edmund M. Clarke, James R. Faeder, Christopher James Langmead, Leonard A. Harris, Sumit Kumar Jha, and Axel Legay. Statistical model checking in BioLab: Applications to the automated analysis of T-cell receptor signaling pathway. In Monika Heiner and Adelinde M. Uhrmacher, editors, *CMSB*, volume 5307 of *Lecture Notes in Computer Science*, pages 231–250. Springer, 2008.

[78] P.T. Monteiro, D. Ropers, R. Mateescu, A.T. Freitas, and H. De Jong. Temporal logic patterns for querying dynamic models of cellular interaction networks. *Bioinformatics*, 24(16):227–233, 2008.

[79] Robin Donaldson and David Gilbert. A model checking approach to the parameter estimation of biochemical pathways. In *Proceedings of the 6th International Conference on Computational Methods in Systems Biology*, pages 269–287. Springer-Verlag, 2008.

[80] Gregory Batt, Michel Page, Irene Cantone, Gregor Goessler, Pedro Monteiro, and Hidde De Jong. Efficient parameter search for qualitative models of regulatory networks using symbolic model checking. *Bioinformatics*, 26(18):i603–i610, 2010.

[81] Sucheendra K. Palaniappan, Benjamin M. Gyori, Bing Liu, David Hsu, and P.S. Thiagarajan. Statistical model checking based calibration and analysis of bio-pathway models. In *CMSB'13*, pages 120–134, 2013.

[82] François Fages and Sylvain Soliman. Formal cell biology in Biocham. In *Formal Methods for Computational Systems Biology*, pages 54–80. Springer, 2008.

[83] Jiří Barnat, Luboš Brim, Ivana Černá, Sven Dražan, Jana Fabriková, Jan Láník, David Šafránek, and Hongwu Ma. Biodivine: A framework for parallel analysis of biological models. In Ralph-Johan Back, Ion Petre, and Erik de Vink, editors, Proceedings Second International Workshop on *Computational Models for Cell Processes,* Eindhoven, the Netherlands, November 3, 2009, volume 6 of *Electronic Proceedings in Theoretical Computer Science*, pages 31–45. Open Publishing Association, 2009.

[84] Chuan Hock Koh, Masao Nagasaki, Ayumu Saito, Chen Li, Limsoon Wong, and Satoru Miyano. MIRACH: efficient model checker for quantitative biological pathway models. *Bioinformatics*, 27(5):734–735, 2011.

[85] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification*, pages 585–591. Springer, 2011.

[86] Gerd Behrmann, Alexandre David, and Kim G Larsen. A tutorial on Uppaal. In *Formal methods for the design of real-time systems*, pages 200–236. Springer, 2004.

[87] Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification*, pages 167–170. Springer, 2010.

[88] Luboš Brim, Milan Češka, and David Šafránek. Model checking of biological systems. In *Formal Methods for Dynamical Systems*, pages 63–112. Springer, 2013.

[89] O. Cappé, E. Moulines, and T. Rydén. *Inference in hidden Markov models*. Springer Verlag, 2005.

[90] D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *Signal Processing, IEEE Transactions on*, 50(3):736–746, 2002.

[91] M. Nagasaki, R. Yamaguchi, R. Yoshida, S. Imoto, A. Doi, Y. Tamada, H. Matsuno, S. Miyano, and T. Higuchi. Genomic data assimilation for estimating hybrid functional Petri net from time-course gene expression data. *Genome Informatics Series*, 17(1):46, 2006.

[92] A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer Verlag, 2001.

[93] K. Nakamura, R. Yoshida, M. Nagasaki, S. Miyano, and T. Higuchi. Parameter estimation of in silico biological pathways with particle filtering towards a petascale computing. In *Pacific Symposium on Biocomputing*, volume 14, pages 227–238, 2009.

[94] Xin Liu and Mahesan Niranjan. State and parameter estimation of the heat shock response system using Kalman and particle filters. *Bioinformatics*, 28(11):1501–1507, 2012.

[95] J.S. Liu and M. West. Combined parameter and state estimation in simulation-based filtering. In N. Doucet, A. de Freitas and N. J. Gordon, editors, *Sequential Monte Carlo Methods in practice*. New York: Springer-Verlag, 2001.

[96] W.R. Gilks and C. Berzuini. Following a moving target - Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1):127–146, 2001.

[97] Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

[98] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[99] Jay H Lee and N Lawrence Ricker. Extended kalman filter based nonlinear model predictive control. *Industrial & Engineering Chemistry Research*, 33(6):1530–1541, 1994.

[100] Simon J. Julier and Jeffrey K. Uhlmann. New extension of the Kalman filter to nonlinear systems, 1997.

[101] J. Zhong and M. Brown. Joint state and parameter estimation for biochemical dynamic pathways with iterative extended Kalman filter: Comparison with dual state and parameter estimation. *Open Automation and Control Systems Journal*, 2(1):69–77, 2009.

[102] G. Lillacci and M. Khammash. Parameter estimation and model selection in computational biology. *PLoS Computational Biology*, 6(3):e1000696, 2010.

[103] M. Quach, N. Brunel, and F. d'Alché Buc. Estimating parameters and hidden variables in non-linear state-space models based on ODEs for biological networks inference. *Bioinformatics*, 23(23):3209, 2007.

[104] Drew Creal. A survey of sequential Monte Carlo methods for economics and finance. *Econometric Reviews*, 31(3):245–296, 2012.

[105] S. Thrun. Particle filters in robotics. In *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.

[106] Peter Jan Van Leeuwen. Particle filtering in geophysical systems. *Monthly Weather Review*, 137(12), 2009.

[107] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009.

[108] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *Radar and Signal Processing, IEE Proceedings F*, volume 140, pages 107–113. IET, 1993.

[109] G. Kitagawa. A self-organizing state-space model. *Journal of the American Statistical Association*, pages 1203–1215, 1998.

[110] Shinya Tasaki, Masao Nagasaki, Hiroko Kozuka-Hata, Kentaro Semba, Noriko Gotoh, Seisuke Hattori, Jun-ichiro Inoue, Tadashi Yamamoto, Satoru Miyano, Sumio Sugano, et al. Phosphoproteomics-based modeling defines the regulatory mechanism underlying aberrant EGFR signaling. *PloS One*, 5(11):e13926, 2010.

[111] D.S. Lee and N.K.K. Chia. A particle algorithm for sequential Bayesian parameter estimation and model selection. *Signal Processing, IEEE Transactions on*, 50(2):326–336, 2002.

[112] N. Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

[113] Jun S. Liu. *Monte Carlo strategies in scientific computing*. Springer Series in Statistics. Springer, New York, 2008.

[114] W Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[115] Mike West. Approximating posterior distributions by mixture. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 409–422, 1993.

[116] Min-An Chao, Chun-Yuan Chu, Chih-Hao Chao, and An-Yeu Wu. Efficient parallelized particle filter design on CUDA. In *Signal Processing Systems (SIPS), 2010 IEEE Workshop on*, pages 299–304, 2010.

[117] J Vanlier, CA Tiemann, Peter AJ Hilbers, and Natal AW van Riel. A Bayesian approach to targeted experiment design. *Bioinformatics*, 28(8):1136–1142, 2012.

[118] J Vanlier, CA Tiemann, Peter AJ Hilbers, and Natal AW van Riel. An integrated strategy for prediction uncertainty analysis. *Bioinformatics*, 28(8):1130–1135, 2012.

[119] I Swameye, TG Müller, J t Timmer, O Sandra, and U Klingmüller. Identification of nucleocytoplasmic cycling as a remote sensor in cellular signaling by data based modeling. *Proceedings of the National Academy of Sciences*, 100(3):1028–1033, 2003.

[120] J.G. Albeck, J.M. Burke, B.B. Aldridge, M. Zhang, D.A. Lauffenburger, and P.K. Sorger. Quantitative analysis of pathways controlling extrinsic apoptosis in single cells. *Molecular cell*, 30(1):11–25, 2008.

[121] Rajeev Alur, Thomas A Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *Software Engineering, IEEE Transactions on*, 22(3):181–201, 1996.

[122] Eugene Asarin, Oded Maler, and Amir Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical computer science*, 138(1):35–65, 1995.

[123] Jasmin Fisher, Nir Piterman, Alex Hajnal, and Thomas A Henzinger. Predictive modeling of signaling crosstalk during c. elegans vulval development. *PLoS Computational Biology*, 3(5):e92, 2007.

[124] John Heath, Marta Kwiatkowska, Gethin Norman, David Parker, and Oksana Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 391(3):239 – 257, 2008.

[125] Luca Bortolussi and Guido Sanguinetti. Learning and designing stochastic processes from logical constraints. In *Quantitative Evaluation of Systems*, pages 89–105. Springer, 2013.

[126] N. Metropolis. The beginning of the Monte Carlo method. *Los Alamos Sci.*, (15, Special Issue):125–130, 1987. Stanislaw Ulam 1909–1984.

[127] Patrick Flaherty, Mala L Radhakrishnan, Tuan Dinh, Robert A Rebres, Tamara I Roach, Michael I Jordan, and Adam P Arkin. A dual receptor crosstalk model of G-protein-coupled signal transduction. *PLoS Computational Biology*, 4(9):e1000185, 2008.

[128] Ben Calderhead and Mark Girolami. Estimating Bayes factors via thermodynamic integration and population MCMC. *Computational Statistics & Data Analysis*, 53(12):4028–4045, 2009.

[129] A. Gelman and D.B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992.

[130] Stephen P. Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455, 1998.

[131] Sumit Kumar Jha, Edmund M. Clarke, Christopher James Langmead, Axel Legay, André Platzer, and Paolo Zuliani. A Bayesian approach to model checking biological systems. In Pierpaolo Degano and Roberto Gorrieri, editors, *CMSB*, volume 5688 of *Lecture Notes in Computer Science*, pages 218–234. Springer, 2009.

[132] Håkan LS Younes and Reid G Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *Computer Aided Verification*, pages 223–235. Springer, 2002.

[133] Christian P. Robert and George Casella. *Monte Carlo statistical methods*. Springer Texts in Statistics. Springer-Verlag, New York, second edition, 2004.

[134] C. A. León and F. Perron. Optimal Hoeffding bounds for discrete reversible Markov chains. *Ann. Appl. Probab.*, 14(2):958–970, 2004.

[135] B. Miasojedow. Hoeffding's inequalities for geometrically ergodic Markov chains on general state space. *arXiv preprint*, 2012.

[136] C.J. Geyer. Practical Markov chain Monte Carlo. *Statistical Science*, 7(4):473–483, 1992.

[137] Benjamin Mate Gyori. Supplementary website. `http://bgyori.comp.nus.edu.sg/smc-mcmc`.

[138] John G Albeck, John M Burke, Sabrina L Spencer, Douglas A Lauffenburger, and Peter K Sorger. Modeling a snap-action, variable-delay switch controlling extrinsic cell death. *PLoS Biology*, 6(12):e299, 2008.

[139] Szymon Stoma, Alexandre Donzé, François Bertaux, Oded Maler, and Gregory Batt. STL-based analysis of TRAIL-induced apoptosis challenges the notion of type I/type II cell line classification. *PLoS Computational Biology*, 9(5):e1003056, 2013.

[140] Suzanne Gaudet, Sabrina L Spencer, William W Chen, and Peter K Sorger. Exploring the contextual sensitivity of factors that determine cell-to-cell variability in receptor-mediated apoptosis. *PLoS Computational Biology*, 8(4):e1002482, 2012.

[141] Sabrina L Spencer, Suzanne Gaudet, John G Albeck, John M Burke, and Peter K Sorger. Non-genetic origins of cell-to-cell variability in TRAIL-induced apoptosis. *Nature*, 459(7245):428–432, 2009.

[142] Bree B Aldridge, Suzanne Gaudet, Douglas A Lauffenburger, and Peter K Sorger. Lyapunov exponents and phase diagrams reveal multi-factorial control over TRAIL-induced apoptosis. *Molecular systems biology*, 7(1), 2011.

[143] Marta Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In *Formal Methods for Performance Evaluation*, pages 220–270. Springer, 2007.

[144] Andrew Golightly and Darren J Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface Focus*, 1(6):807–820, 2011.

[145] Douglas Hanahan and Robert A Weinberg. The hallmarks of cancer. *Cell*, 100(1):57–70, 2000.

[146] M.L. Blinov, J.R. Faeder, B. Goldstein, and W.S. Hlavacek. BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–3291, 2004.

[147] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 441–444, 2006.

[148] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling of cellular signalling. *CONCUR 2007*, pages 17–41, 2007.

[149] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620, 2000.

[150] M. Zou and S.D. Conzen. A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21(1):71–79, 2005.

[151] A. Bauer-Mehren, L.I. Furlong, and F. Sanz. Pathway databases and tools for their exploitation: benefits, current limitations and challenges. *Molecular systems biology*, 5(1), 2009.

[152] L.J. Jensen, J. Saric, and P. Bork. Literature mining for the biologist: from information retrieval to biological discovery. *Nature Reviews Genetics*, 7(2):119–129, 2006.

[153] R.E. Neapolitan. *Learning Bayesian networks*. Pearson Prentice Hall Upper Saddle River, NJ, 2004.

[154] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[155] A. Julius and C. Belta. Genetic regulatory network identification using monotone functions decomposition. In *18th IFAC World Congress*, Milan, Italy, 2011.

[156] T. Jaakkola, D. Sontag, A. Globerson, and M. Meila. Learning Bayesian network structure using LP relaxations. In *Proceedings of the 13th International Conference on Artifcial Intelligence and Statistics, AISTATS'10*, pages 358–365. Society for Artificial Intelligence and Statistics, 2010.

[157] Stephen Neidle. *Cancer drug design and discovery.* Academic Press, 2011.

[158] B. Liu, A. Hagiescu, S.K. Palaniappan, B. Chattopadhyay, Z. Cui, W.F. Wong, and PS Thiagarajan. Approximate probabilistic analysis of biopathway dynamics. *Bioinformatics*, 28(11):1508–1516, 2012.

[159] M.J. Lee, A.S. Ye, A.K. Gardino, A.M. Heijink, P.K. Sorger, G. MacBeath, and M.B. Yaffe. Sequential application of anticancer drugs enhances cell death by rewiring apoptotic signaling networks. *Cell*, 149(4):780–794, 2012.

[160] Chris J Needham, James R Bradford, Andrew J Bulpitt, and David R Westhead. A primer on learning in Bayesian networks for computational biology. *PLoS Computational Biology*, 3(8):e129, 2007.

[161] H. De Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[162] Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 139–147. Morgan Kaufmann Publishers Inc., 1998.

[163] Dirk Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19(17):2271–2282, 2003.

[164] Sun Yong Kim, Seiya Imoto, and Satoru Miyano. Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings in bioinformatics*, 4(3):228–235, 2003.

[165] Marco Grzegorczyk, Dirk Husmeier, Kieron D Edwards, Peter Ghazal, and Andrew J Millar. Modelling non-stationary gene regulatory processes with a non-homogeneous Bayesian network and the allocation sampler. *Bioinformatics*, 24(18):2071–2078, 2008.

[166] Zheng Li, Ping Li, Arun Krishnan, and Jingdong Liu. Large-scale dynamic gene regulatory network inference combining differential equation models with local dynamic Bayesian network analysis. *Bioinformatics*, 27(19):2686–2691, 2011.

[167] Marco Grzegorczyk and Dirk Husmeier. Non-homogeneous dynamic Bayesian networks for continuous data. *Machine Learning*, 83(3):355–419, 2011.

[168] Jonathan B Fitzgerald, Birgit Schoeberl, Ulrik B Nielsen, and Peter K Sorger. Systems biology and combination therapy in the quest for clinical efficacy. *Nature Chemical Biology*, 2(9):458–466, 2006.

[169] Alexander Mitsos, Ioannis N Melas, Paraskeuas Siminelakis, Aikaterini D Chairakaki, Julio Saez-Rodriguez, and Leonidas G Alexopoulos. Identifying drug effects via pathway alterations using an integer linear programming optimization formulation on phosphoproteomic data. *PLoS Computational Biology*, 5(12):e1000591, 2009.

[170] Ioannis N Melas, Alexander Mitsos, Dimitris E Messinis, Thomas S Weiss, and Leonidas G Alexopoulos. Combined logical and data-driven models for linking signalling pathways to cellular response. *BMC Systems Biology*, 5(1):107, 2011.

[171] Sven Nelander, Weiqing Wang, Björn Nilsson, Qing-Bai She, Christine Pratilas, Neal Rosen, Peter Gennemark, and Chris Sander. Models from experiments: combinatorial drug perturbations of cancer cells. *Molecular Systems Biology*, 4(1), 2008.

[172] Evan J Molinelli, Anil Korkut, Weiqing Wang, Martin L Miller, Nicholas P Gauthier, Xiaohong Jing, Poorvi Kaushik, Qin He, Gordon Mills, David B Solit, et al. Perturbation biology: inferring signaling networks in cellular systems. *PLoS Computational Biology*, 9(12):e1003290, 2013.

[173] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 27(1):29–34, 1999.

[174] Carl F Schaefer, Kira Anthony, Shiva Krupa, Jeffrey Buchoff, Matthew Day, Timo Hannay, and Kenneth H Buetow. PID: the pathway interaction database. *Nucleic Acids Research*, 37(suppl 1):D674–D679, 2009.

[175] Michelle N Arbeitman, Eileen EM Furlong, Farhad Imam, Eric Johnson, Brian H Null, Bruce S Baker, Mark A Krasnow, Matthew P Scott, Ronald W Davis, and Kevin P White. Gene expression during the life cycle of drosophila melanogaster. *Science*, 297(5590):2270–2275, 2002.

[176] K. Murphy and Y. Weiss. The factored frontier algorithm for approximate inference in DBNs. In *Proceedings of the Seventeenth conference on Uncertainty in Artificial Intelligence*, UAI'01, pages 378–385, 2001.

[177] M. Berkelaar, K. Eikland, P. Notebaert, et al. lpsolve: Open source (mixed-integer) linear programming system. *Eindhoven U. of Technology*, 2004.

[178] Hans Mittelmann. Benchmarks for optimization software. *See http://plato.la.asu.edu/bench.html*, 2002.

[179] Manindra Agrawal, S Akshay, Blaise Genest, and PS Thiagarajan. Approximate verification of the symbolic dynamics of Markov chains. In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science*, pages 55–64. IEEE Computer Society, 2012.

[180] YoungMin Kwon and Gul Agha. Verifying the evolution of probability distributions governed by a DTMC. *Software Engineering, IEEE Transactions on*, 37(1):126–141, 2011.

[181] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth conference on Uncertainty in Artificial Intelligence*, UAI'98, 1998.

[182] S.K. Palaniappan, S. Akshay, B. Liu, B. Genest, and PS Thiagarajan. A hybrid factored frontier algorithm for dynamic Bayesian networks with a biopathways application. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(5):1352–1365, 2012.

[183] Fred Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166, 1977.

[184] Nelson Fausto and Jean S Campbell. Mouse models of hepatocellular carcinoma. In *Seminars in liver disease*, volume 30, pages 087–098, 2010.

[185] Reginald F Clayton, Angela Rinaldi, Eve E Kandyba, Mike Edward, Christian Willberg, Paul Klenerman, and Arvind H Patel. Liver cell lines for the study of hepatocyte functions and immunological response. *Liver International*, 25(2):389–402, 2005.

[186] Stefan Wilkening, Frank Stahl, and Augustinus Bader. Comparison of primary human hepatocytes and hepatoma cell line HepG2 with regard to their biotransformation properties. *Drug Metabolism and Disposition*, 31(8):1035–1042, 2003.

[187] Lun He, Kurt J Isselbacher, Jack R Wands, Howard M Goodman, Chiaho Shih, and Andrea Quaroni. Establishment and characterization of a new human hepatocellular carcinoma cell line. *In Vitro*, 20(6):493–504, 1984.

[188] J. Saez-Rodriguez, A. Goldsipe, J. Muhlich, L.G. Alexopoulos, B. Millard, D.A. Lauffenburger, and P.K. Sorger. Flexible informatics for linking experimental data to mathematical models via DataRail. *Bioinformatics*, 24(6):840–847, 2008.

[189] Daniel Morgensztern and Howard L McLeod. PI3K/Akt/mTOR pathway as a target for cancer therapy. *Anti-cancer Drugs*, 16(8):797–803, 2005.

[190] Eisuke Yasuda, Takashi Kumada, Shinji Takai, Akira Ishisaki, Takahiro Noda, Rie Matsushima-Nishiwaki, Naoki Yoshimi, Kanefusa Kato, Hidenori Toyoda, Yuji Kaneoka, et al. Attenuated phosphorylation of heat shock protein 27 correlates with tumor progression in patients with hepatocellular carcinoma. *Biochemical and Biophysical Research Communications*, 337(1):337–342, 2005.

[191] Jannis Uhlendorf, Agnès Miermont, Thierry Delaveau, Gilles Charvin, François Fages, Samuel Bottani, Gregory Batt, and Pascal Hersen. Long-term model predictive control of gene expression at the population and single-cell levels. *Proceedings of the National Academy of Sciences*, 109(35):14271–14276, 2012.

[192] Samuel Bandara and Tobias Meyer. Design of experiments to investigate dynamic cell signaling models. In Xuedong Liu and Meredith D. Betterton, editors, *Computational Modeling of Signaling Networks*, volume 880 of *Methods in Molecular Biology*, pages 109–118. Humana Press, 2012.

[193] William S Hlavacek. How to deal with large models? *Molecular Systems Biology*, 5(1), 2009.

[194] Mohammad Fallahi-Sichani, Saman Honarnejad, Laura M Heiser, Joe W Gray, and Peter K Sorger. Metrics other than potency reveal systematic variation in responses to cancer drugs. *Nature Chemical Biology*, 2013.

[195] Mario Niepel, Marc Hafner, Emily A Pace, Mirra Chung, Diana H Chai, Lili Zhou, Jeremy L Muhlich, Birgit Schoeberl, and Peter K Sorger. Analysis of growth factor signaling in genetically diverse breast cancer lines. *BMC Biology*, 12(1):20, 2014.

[196] D. Paulin. Concentration inequalities for Markov chains by Marton couplings and spectral methods. *arXiv preprint*, 2013.