

Integrating prior knowledge to reverse-engineer simple biological systems

Short-Project Report

by

Marton Munz

under the supervision of

Prof. Jotun Hein



Systems Biology Doctoral Training Centre
The Genome Analysis and Bioinformatics Group
University of Oxford
July 2008

Abstract

A new system identification algorithm is presented and tested on synthetic datasets. The approach was efficient in inferring ordinary differential equation (ODE) models from time series data. A general mathematical framework is proposed to integrate prior biological knowledge in order to improve the correctness of the reverse-engineered models. Prior knowledge makes it possible to build accurate dynamic models even if it wouldn't be feasible using only time-course measurements. The promising results on synthetic data suggest that the method could be a useful tool in building models for systems biology.

Contents

1	Introduction	1
2	Summary of literature	3
3	Problem definition	6
4	The selected model class	8
5	Applied methods	10
5.1	Numerical integration of ODEs	10
5.1.1	Euler method	10
5.1.2	Fourth order Runge-Kutta method	11
5.2	Global optimization	11
5.2.1	Brute force or exhaustive search	12
5.2.2	Markov chain Monte Carlo	12
5.3	Iterated Grid Search	12
5.4	Bayesian inference	13
6	Theoretical studies	15
6.1	Hypergraphs and ODEs	15
6.2	Strategies to simplify the problem	17
6.2.1	Analyzing independent components	17
6.2.2	Building sparse models	17
6.2.3	Two-level optimization	18
6.3	Integrating biological knowledge	20
6.3.1	Additional comments	21
7	The algorithm	22
7.1	The main algorithm	22
7.2	Description	22
7.3	Additional comments	24

8	Results	25
8.1	Random synthetic systems	25
8.2	Inferring synthetic systems	28
8.3	Noisy data and the help of knowledge	28
8.4	Using \tilde{E} for uncertain knowledge	29
9	Conclusions	32
10	Technical notes	36

Chapter 1

Introduction

"Homeostasis", "Adaptation", "Regulation", "Response" — basic characteristic features of all living organisms. However, interestingly, none of these concepts can be defined without using the notion of "time". You also won't be able to define the concept of "life" itself, without thinking in time. Biology should be as time-dependent as Physics.

Of course we may draw static graphs of interactions to describe biological systems — but to understand them, we need to study their dynamic behavior. The static pattern of interactions in a biological network is optimized to be able to generate a dynamic pattern of molecular concentrations inside the cell.

One of the most important contributions Systems Biology may make to our understanding of living systems, is to build as detailed dynamic models for biology as possible — with the least simplification in the model type and in the number of parameters. However, biological networks contain large number of components, whilst building detailed dynamic models (like ODEs) with many variables is extremely difficult. It requires a joint optimization of a large set of properties (parameters) of the model.

Most of the dynamic models made so far in biology involve some simplification in the model type. Although these models can be extended to large systems and may explain some basic features (for example, even oversimplified Boolean networks predict attractors corresponding to different cell states), they can't explain the real time behavior of the system.

In my project I suggest an other way to improve the identifiability of accurate ODE models, even in case of incomplete and noisy time series dataset. When time-course measurement alone doesn't enable to infer the underlying ODEs, sufficient amount of knowledge about the system may be collected from different experiments/databases to help the model optimization.

Although knowledge may originate from different sources, they show different aspects of the same biological phenomenon. Therefore it makes sense to integrate all knowledge we

have about the system structure. The question is how to "put together" these "knowledge pieces" from different databases into a whole picture, and apply them as a prior belief to reduce the size and dimensionality of the system identification problem.

Important note

My results are only theoretical in the sense that so far in my research only synthetic datasets were used to test the presented new method. However, if the algorithm performs well on synthetic systems mimicking real biological systems, it is probably efficient on real datasets too. Testing the method on real time-series data could be the next step.

Chapter 2

Summary of literature

One of the most important aims of Systems Biology is to build mathematical models which can describe, predict and explain biological systems. The proteome, the transcriptome and the metabolome of a cell can be seen as complex networks, with large number of components (genes, proteins, mRNAs, metabolites etc.) and many interactions between them. High-throughput experimental techniques (such as DNA microarray experiments, mass spectrometry and two-hybrid screening) made the reconstruction of biological networks possible (**Ross et al., 2006**).

There were extensive studies on the large-scale properties of complex biological networks (**Jeong et al., 2000, 2001, Giot et al., 2003, Yook et al., 2004, Balázsi et al., 2005., Palla et al. 2005**) However these surveys consider static interaction maps, which are only snapshots of the real biological systems. For deeper understanding of cell phenotypes, it is essential to build models which capture the dynamical properties of biological networks.

During the recent years many work aimed to develop algorithms to automatically reverse-engineer biological dynamical systems from time series data. For example, studying the relaxation after pulse perturbation of the steady state concentrations of chemical species can elucidate causal connectivities between the components. This approach was successful for example in case of the glycolytic pathway (**Vance et al., 2002**). Similar method was applied to gene regulatory networks (**Tegnér et al., 2003**).

Metabolic control analysis (MCA) gives information about the system structure and function by the sensitivity analysis of network parameters. (**Fell, 1997**) Correlations analysis of time series data of concentration around a stationary state also helps to draw a wiring diagram of the system (**Arkin and Ross, 1995**). Global and local optimization techniques were used to estimate the parameters of known system structures (**Peifer and Timmer, 2007, Banga et al., 2003**).

Types of constructed dynamic models include Boolean- and Bayesian networks, Petri nets, difference and differential equation models, hybrid models etc. However, the crucial ques-

tion is, if detailed dynamic models can be extended to large scale, like genome or proteome size (**Schlitt and Brazma, 2005, 2007**). Unfortunately, the more detailed models we build, the fewer components we can incorporate, due to the increasing number of model parameters. The size and the dimensionality of the search space increases rapidly with the number of parameters.

Not surprisingly ODE models typically contain only a few dozen or less components. The largest ODE models available today include the network of segment polarity genes in the early development of *Drosophila* (**Dassow et al., 2000**) and the cell cycle model of Fission Yeast (**Novak and Tyson, 1997**). Much more components can be modeled by Bayesian networks, but they give only a rough approximation of the real dynamic behaviors.

According to **Schlitt and Brazma**, the success of building real time models for only a small part of a network highly depends on that in what extent the biological process can be decoupled from other cellular processes. Although they asked it for gene regulatory networks, the same question stands for modeling protein interaction networks or biochemical pathways: can we extend detailed dynamical models to the proteome- or metabolome scale?

Thinking of the parameter estimation task as a regression problem (as writes **Ross et al., 2006**), we can solve it only if the size of the learning set is larger than the number of variables (here: parameters). Consequently, the main problem with large detailed models, like ODEs, is that they contain too many parameters to fit properly to the data. The task gets even more difficult, when the dataset is noisy or incomplete. Noisy data may result in overfitting, particularly in case of many variables. The objective function in a high dimension search space often contain many local minima, which make the search difficult. In case of noise, the best fitting model must not be the most realistic one.

One possible way out from these problems is putting together biological knowledge from various experiments. It could decrease the size and dimensionality of the search space. In a selected region of the space, the minimum of the objective function may correspond to the real system, even in case of noise. Due to the large number of genes and limited amount of gene expression data, microarray datasets alone are not enough to infer accurate gene regulatory networks. (**Imoto et al., 2004**) For this reason, several approaches have been suggested for the application of prior biological knowledge to help the analysis of microarray datasets. They use different sources of knowledge, including promoter sequences (**Pilpel et al. 2001, Segal et al. 2002**), collected information from published literature (**Masys, 2001**) and correlation of expression values (**Bussemaker et al., 2001**).

The above mentioned strategies suggest the possibility of a general mathematical frame-

work to integrate prior knowledge from different sources. I will investigate this idea in my project. **Imoto et al.** proposed a similar approach, but it is limited to Bayesian network models and can integrate information only about known interactions. **Julius et al. (2007)** published an algorithm based on convex programming, that can incorporate knowledge about the network structure, but it can't take into account the uncertainty of our knowledge. **Todorovski and Dzeroski (2007)** used context free grammars as domain-specific knowledge to select the model class of interest. However, I've found surprisingly few papers on a general methodology of integration different types of knowledge to infer biological systems.

Generally, the field of System Identification has rich literature, due to its applicability in many areas (for some book reviews see **Ljung, 1999, Katayama, 2005, Pintelon and Schoukens, 2001, Greblicki and Pawlak, 2008**). One special topic, equation discovery, also has substantial literature covering a wide range of mathematical techniques. Systems Biology may learn a lot from applications in Electrical Engineering, Physics, Economy, Control Theory, Artificial Intelligence, Signal Processing etc.

Instead of making black box models, gray box modeling is usual in the above mentioned fields, in cases when it is possible to build prior models of the system using prior knowledge.

Chapter 3

Problem definition

The general task I try to solve is the reconstruction of biochemical networks using time series data about the concentrations of the components in the system. The following section gives the exact mathematical definition for this problem.

Suppose we have a system of n components (chemicals, genes etc.), which is a network in the sense that its components are connected by interactions (chemical reactions etc).

Let $t_k, 1 \leq k \leq N$ be a discrete series of time points. Suppose we have time-course data on the concentrations of the components at these time points:

$$\begin{aligned} &x_1(t_1), x_1(t_2), x_1(t_3), \dots x_1(t_N) \\ &x_2(t_1), x_2(t_2), x_2(t_3), \dots x_2(t_N) \\ &x_3(t_1), x_3(t_2), x_3(t_3), \dots x_3(t_N) \\ &\dots \\ &x_n(t_1), x_n(t_2), x_n(t_3), \dots x_n(t_N) \end{aligned}$$

Let T^0 be the set of these n trajectories. T_i^0 means the one-dimensional trajectory of the i^{th} component. T^0 itself is a trajectory in the n -dimensional vector space. (I will sometimes call T^0 as the template trajectory.)

We build a dynamical model of the system in the form of coupled ODEs, so we assume that T^0 is generated by an equation system of the following form:

$$dx_k = p_k(x(t))dt \tag{3.1}$$

To simplify the problem let p_k be polynomials.

There are many ways to define the similarity of the generated trajectory and the template trajectory. I will use the Euclidean distance for this purpose. Let m be a model and T^m

be the trajectory generated by this model. Then the measure of similarity is:

$$d(T^m, T^0) = \sqrt{\sum_{1 \leq i \leq n} \sum_{1 \leq k \leq N} (T_i^m(t_k) - T_i^0(t_k))^2} \quad (3.2)$$

Let M be the space of possible models. We search for the optimal $m \in M$, which minimizes the following objective function (which I usually mention as energy):

$$E(m) : m \rightarrow d(T^m, T^0) \quad (3.3)$$

Therefore we have to solve a global optimization problem, looking for the set of p_k polynomials which corresponds to the global minimum of E .

Chapter 4

The selected model class

In my work I will concentrate on biological systems composed of a set of biochemical reactions. In this section I describe the standard mathematical model used for such systems.

Reaction kinetics is the study of the rates of chemical reactions. Reaction rates are defined by the speed of concentration changes in a given chemical process. Let's consider a typical reaction:



Here capital letters stand for reactants, while lowercase letters represent stoichiometric coefficients (the degree to which a chemical species participate in the reaction). Eq. 4.1. describes a reversible transformation from two initial compounds (A and B) into two final compounds (C and D). In this example the reaction rate is:

$$r = -\frac{1}{a} \frac{d[A]}{dt} = -\frac{1}{b} \frac{d[B]}{dt} = \frac{1}{c} \frac{d[C]}{dt} = \frac{1}{d} \frac{d[D]}{dt} \quad (4.2)$$

where, for example, $[A]$ represents the concentration of reactant A. In case of more complex reaction networks, stoichiometric values are represented in a stoichiometry matrix.

The law of mass action (Guldberg and Waage, 1864-79) states that for elementary reactions (which proceed through only one transition state) the reaction rate is proportional to the product of the concentrations raised to the power of the stoichiometric coefficient of the participating reactants. The rule can be derived from collision theory (Trautz and Lewis, 1916-18) using statistical physics.

In Eq. 4.1. the forward reaction rate is:

$$r_{\rightarrow} = k_{\rightarrow} [A]^a [B]^b \quad (4.3)$$

while the backward reaction rate is:

$$r_{\leftarrow} = k_{\leftarrow}[C]^c[D]^d \quad (4.4)$$

where k_{\rightarrow} and k_{\leftarrow} are the rate constants of the reaction.

In dynamic equilibrium, the two reaction rates must be equal, therefore we get:

$$K_{eq} = \frac{k_{\rightarrow}}{k_{\leftarrow}} = \frac{[C]^c[D]^d}{[A]^a[B]^b} \quad (4.5)$$

where K_{eq} is the equilibrium constant.

Biochemical reactions are often catalyzed by enzymes, so their kinetics can't be described by only the collision of molecules. Suppose we have the following basic enzyme-catalyzed reaction:



where E is the enzyme, S is the substrate, ES is their complex and P is the product. The enzyme and the substrate forms a complex by the rate constant of k_1 , but this transformation is reversed by the rate constant of k_{-1} . The production of P is by the rate constant of k_2 . If we assume that the system is in steady state and $[S] \gg [E]$, we get the Michaelis-Menten equation for the reaction rate:

$$\frac{d[P]}{dt} = \frac{V_{max}[S]}{K_M + [S]} \quad (4.7)$$

where V_{max} is the maximal reaction rate and K_M is the Michaelis-Menten constant.

As I will describe in Section 6.1. I choose a model class which captures mass action kinetics, and which is easily extendable to Michaelis-Menten kinetics. Although I am interested in biochemical reaction networks, I make a more general model representation, which can capture different system classes, like linear additive regulation models used for gene regulatory networks.

Chapter 5

Applied methods

5.1 Numerical integration of ODEs

ODE models usually define a system by describing the time-derivative of the vector (y) of its variables:

$$y'(t) = f(t, y(t)) \quad (5.1)$$

The Initial Value Problem (IVP) involves solving an ODE system with an initial condition, which specifies a point of the unknown $y(t)$ solution. Many numerical methods were developed for IVP. (Lambert, 1991) In this section I describe two of the best-known algorithms, which belong to the class of discrete variable methods. These methods approximate the solution $y(t)$ as a discrete $y_0, y_1, y_2, \dots, y_{n-1}, y_n$ series of values with a time step of h . We start from the $y(t_0) = y_0$ initial condition.

5.1.1 Euler method

This method uses a linear approximation by taking into account only the first two terms of the Taylor expansion of y . The points of the solution are calculated iteratively by the following formula:

$$y_{n+1} = y_n + hf(t_n, y_n) \quad (5.2)$$

The error due to Euler method can be written as:

$$\frac{1}{2}h^2y''(t_0) + O(h^3) \quad (5.3)$$

For small h , the error is approximately proportional to h^2 . Since the number of time points needed to cover a given time interval is proportional to $\frac{1}{h}$, we can expect a total error proportional to h at the end of the interval. The Euler method is therefore called a first-order numerical procedure. It is less accurate and numerically more unstable than the following Runge-Kutta method.

5.1.2 Fourth order Runge-Kutta method

This method estimates the slope of the function as the weighted average of four slopes calculated at the beginning, middle and the end of the time step interval:

$$k_1 = f(t_n, y_n) \tag{5.4}$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \tag{5.5}$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \tag{5.6}$$

$$k_4 = f(t_n + h, y_n + hk_3) \tag{5.7}$$

The next point of the function is estimated as:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{5.8}$$

The error per step is on the order of h^5 , and the total accumulated error at the end of the range of integration is on the order of h^4 . The method is therefore called a fourth-order numerical procedure.

5.2 Global optimization

Let Q be the set of all candidate hypotheses in a specific problem, and let's consider the following function:

$$F : Q \rightarrow \mathbf{R} \tag{5.9}$$

The set Q , often called the search space, may be for example a vector space representing a set of unknown parameter values. The function F is called the objective function. Global optimization aims to find the global minimum/maximum of this objective function in a given Q search space.

Due to its generality, global optimization became important in a number of different fields, and many methods was proposed to solve the problem. These can be categorized as stochastic (simulated annealing and variants, genetic algorithms etc.) or deterministic (branch-and-bound techniques, dynamic programming etc.). Stochastic methods incorporate some randomness into the objective function or into the algorithm itself. Stochasticity may help to escape from local optima, but the success of such methods is not guaranteed.

Here I briefly describe the two basic search techniques I used in the system inference algorithm. (See Chapter 7. for the description of the algorithm.)

5.2.1 Brute force or exhaustive search

This trivial method means evaluating F for $\forall x \in Q$, and selecting the optimum from the results. It can be used only for search spaces of small size.

5.2.2 Markov chain Monte Carlo

A basic algorithm for stochastic optimization is the Markov chain Monte Carlo (MCMC) method using the Metropolis acceptance criteria. (Liu, 2001) The algorithm builds a random Markov chain. The next state of the chain is selected by the following rule:

$$\begin{aligned} X_{n+1} &= T(X_n) \text{ if } r \leq \min \{ 1, e^{-\beta(F(X_{n+1})-F(X_n))} \} \\ X_{n+1} &= X_n \text{ else} \end{aligned}$$

where $T(X_n)$ is the proposal function which returns a random state by perturbing the previous X_n state; r is a random value from the $[0,1]$ uniform distribution, β is an inverse temperature parameter and $F(x)$ is the function to be optimized. One has to choose a proposal function, which results in an ergodic simulation. (E.g. from any given state we can reach any other state in a finite number of steps.)

The minimum of the distribution of states generated by the Metropolis MCMC converges to the global minimum of F . However, this convergence may be extremely slow due to the freezing problem: the algorithm may be trapped by local minima. Despite of the theoretical ergodicity it causes a quasi-ergodic behavior.

Several methods were proposed to avoid this problem (Parallel Tempering: Earl and Deem 2005, Simulated Annealing: Kirkpatrick et al. 1983, Stochastic Tunneling: Wenzel and Hamacher. 1999, etc.). Although I tried these techniques in my research, I don't describe them here, because they are not part of the final search algorithm.

5.3 Iterated Grid Search

To find optimal parameter values I will use a modified version of the Iterated Grid Search algorithm (IGS), introduced by Jinhyo Kim in 1997. The basic algorithm was proposed to identify the global maximum of unimodal functions. Unimodal functions have only one global maximum and no local maxima. Although the definition of unimodality is not trivial for multivariate functions, Kim extended the algorithm to m -dimensional problems. I describe here a modified version of the original IGS algorithm.

Suppose, we have an m -dimensional function $F(x_1, x_2, \dots, x_m)$, and we know that the

global optimum point $(X_1, X_2, X_3, \dots, X_m)$ is included by the following initial region:

$$x_i^1 < X_i < x_i^2 \quad (5.10)$$

Our uncertainty about the localization of optimum is therefore described by the above mentioned $[x_i^1, x_i^2]$ intervals. Two steps are iterated:

Step 1: We place n equally distributed grid points in each interval and combine them to get n times m grid points in the whole region:

$$G_{ij} = x_i^1 + j * (x_i^2 - x_i^1) / (n + 1) \quad (5.11)$$

$$1 \leq j \leq n \quad (5.12)$$

Step 2: We evaluate the function F at each grid points, and select the maximum (or minimum) of these values. We set the surrounding m -dimensional cell as the $[x_i^1, x_i^2]$ region of uncertainty in the following iteration. Therefore the method will always refine the results until a number of iterations are done or a stopping criteria is satisfied.

This algorithm has 2 key parameters: the number of iterations and the number of grid points used. Kim proved that using only 2 grid points is enough to certainly identify the global maximum of an unimodal m -dimensional function. However, in our case, the parameter space (for a given topology) may contain several local minima, so more than 2 grid points may be necessary.

The drawback of this method is that even if we use many grid points there is no guarantee for finding the global optimum. In addition, computation time grows rapidly with the number of dimensions, therefore it's not effective for many-dimensional problems. On the other hand the algorithm enables a fast scan of a function in a multidimensional space.

5.4 Bayesian inference

Bayesian inference (Bolstad, 2007) in statistics is a way to express how our belief about a hypothesis changes due to new observations. Let x be a hypothesis and let $P(x)$ be the prior probability by which we think x is true. (Uncertainty of the hypothesis is represented by a probability.) Let $P(D | x)$ be the conditional probability of observing the data D given that x is true. $P(D | x)$ is called the Likelihood of the observation. According to the Bayes theorem the posterior probability of x is given by:

$$P(x | D) = \frac{P(D | x)P(x)}{P(D)} \quad (5.13)$$

Bayes theorem describes a statistical transformation between the uncertainty of the hypothesis before and after observing the data.

Chapter 6

Theoretical studies

6.1 Hypergraphs and ODEs

Although biological networks are usually represented by simple graphs, it is sensible to model a complex interaction system as a hypergraph. A hypergraph is a graph composed by a set of nodes and a set of hyperedges. A hyperedge is the generalization of a simple edge: it can connect not only two, but even more nodes in the graph. Each hyperedge is characterized by the (V_1, V_2) pair of the two connected sets of nodes. In case of directed hyperedges, we call V_1 the head- (or output-) set, and V_2 the tail- (or input-) set (Gallo et al., 1993).

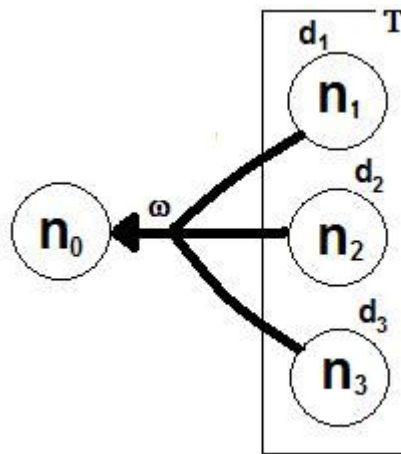


Figure 6.1: Weighted hyperedge with a tail set of three nodes

A special class of directed hypergraphs is the group of backward-hypergraphs (or B-hypergraphs), which are composed by directed hyperedges having a head set of only one element. I will call the only element of head set the target node. Backward-hypergraphs are useful in the representation of systems involving cooperative interactions. If the concentration of a component is influenced by two or more components, their relationship may be described by a hyperedge. The influenced component is represented by the target node, while the set of components affecting it cooperatively, is defined by the tail set.

I will use weighted backward-hypergraphs to describe dynamic systems. Every hyperedge is assigned by $N+1$ weights, where N is the size of the tail set. We give a weight to each nodes in the tail set (d_i) and an additional weight to the whole hyperedge (ω). Figure 6.1. shows a hyperedge with a target node (n_0) and a tail set (T) with 3 element (n_1, n_2, n_3).

It is simple to translate the hypergraph representation into the ODE representation. Suppose we have a hypergraph with n nodes. Let E be the hyperedge vector, for which E_i refers to the set of hyperedges which has the common target node i . For a given e hyperedge, let $d_e(n)$ mean the weight assigned to the node n , if n is included in the $T(e)$ tail set of e . Furthermore let $\omega(e)$ be the weight assigned to the whole e hyperedge. Then let us consider the following ODE system:

$$\frac{dx_k}{dt} = \sum_{\forall e \in E_k} \omega(e) \prod_{\forall n \in T} x_n^{d_e(n)} \quad (6.1)$$

Since the d weights are arbitrary, Eq.6.1. can describe any kind of polynomial ODE system, therefore the whole class of ODEs given in Eq.3.1. Moreover, by setting additional constraints for d and $|T|$, we can select a certain system class. For example the following constraints

$$\forall e \in E_k : \max(T(e)) = 2 \quad (6.2)$$

$$\forall t \in T(e) : d(t) = 1 \quad (6.3)$$

select the model class of mass action dynamics, whilst the following constraints

$$\forall e \in E_k : |T(e)| = 1 \quad (6.4)$$

$$\forall t \in T(e) : d(t) = 1 \quad (6.5)$$

select linear additive regulation models, often used in modeling gene regulatory systems (D'haeseleer et al, 2000).

Since Eq.6.1. is general enough, it can be used to define any system class we want to study. Because Michaelis-Menten terms are characterized by two constants (V_{max} and K_M), the representation can be easily extended to capture enzyme-catalyzed reactions by adding one more weight to the nodes in T .

To keep the generality of the above described description, I will present an algorithm, which performs a search in the space of polynomial ODEs, or equivalently in the space of the described weighted backward-hypergraphs. The algorithm will make elementary modification steps on the topology of the hypergraph, like adding or removing hyperedges, which correspond adding or removing interactions from the system.

6.2 Strategies to simplify the problem

As shown in Chapter 3. we have to solve a global optimization problem, searching for the optimal set of p_k polynomials in Eq. 3.1. However, if we want to find all p_k polynomials at the same time, it leads to an extremely difficult search problem. In my trials, I observed that the objective function has many local minima. (For explanation see Appendix 2) The MCMC simulation always ran into the so called "freezing problem", it was trapped by a local minimum, which made it impossible to identify the best fitting model.

I tried to avoid the quasi-ergodicity of the simulation applying three heuristics (Parallel Tempering, Simulated Annealing and Stochastic Tunneling). However I could never find the global minimum of the objective function. It was not only due to the roughness of the objective function, but also to the large size of the search space. To overcome these problems I developed three strategies. Two for reducing the size of the search space, and one for eliminating the local minima.

6.2.1 Analyzing independent components

It is possible to reformulate the optimization problem of Eq. 3.1. We select a component x_α . The task is to find the polynomial p_α , for which the $x_\alpha(t)$ solution of the following ODE system is optimal:

$$dx_\alpha = p_\alpha(x(t))dt \quad (6.6)$$

$$x_k = T_k^0(t) : \forall k \neq \alpha \quad (6.7)$$

In other words: we split the problem into n independent sub-problems, by treating each component individually. When we describe the dynamics of a component, we consider the dynamics of the rest of variables as known. The optimization of p_α involves fixing the other n-1 trajectories of T^0 . It's able to modify both the Euler and Runge-Kutta methods to solve Eq. 6.6-7. However Runge-Kutta method requires input trajectories of double time point resolution compared to the resolution of the resulting trajectories.

Since all components can be reverse-engineered independently, the basic task is deduced to n optimization problems. However, in each sub-problem the size of the search space is much smaller than in the original task.

6.2.2 Building sparse models

There are many way to define the complexity of a system. I use the following simple definition: Let $m \in M$ be a model. We say that $c(m)$ is the complexity of m, if $c(m)$ is the smallest positive integer number which is more or equal than the in-degree of each nodes in the corresponding hypergraph. Furthermore, let $c(x)$ be the complexity of component x, which is equal to the in-degree of the node. Therefore the complexity of a system is

the maximum of the complexities of its components.

Topologically (not considering exact parameter values) the number of possible combinations of interactions affecting a node of complexity $c(x)$ is:

$$\sum_{1 \leq i \leq c} \frac{(n + \frac{n(n-1)}{2})!}{i!(n + \frac{n(n-1)}{2} - i)!} \quad (6.8)$$

where n is the number of components and we assume first or second order (mass action) interactions.

This sum clearly shows that searching in a space of limited complexity may largely reduce the size of the search space. In order to identify the set of interactions affecting a given component, we may search in the space of interaction sets of limited size.

The reason I think it is right to use a cutoff for $c(x)$ is twofold. Firstly, biological interaction networks are often found to be sparse, which means that they contain relatively small number of interactions compared to the whole number of mathematically possible interactions between their nodes. Therefore building sparse models may not be in conflict with the real structure of systems. Secondly, limiting the number of interactions can help to avoid overfitting, the ability to perfectly fit an arbitrary system by tuning too many parameters.

Given the sparsity of biological networks, we may be interested in systems in the increasing order of complexity. For example, looking at component x , we can search for the best fitting interaction sets by choosing $c(x)=1$, then $c(x)=2$, after that $c(x)=3$ etc. This trick, like the one in 6.2.1, splits the original problem into more easily treatable subproblems.

6.2.3 Two-level optimization

A model $m \in M$ is characterized by its topology (t) and the set of parameter values (p):

$$m(t, p) \quad (6.9)$$

The optimal model is composed by the optimal topology and the optimal parameter set. It is the sensitivity for parameter values that makes the search difficult. An optimal topology with a non-optimal parameter set may result in a highly non-optimal model. If we compare two topologies, we can't decide, which one is better, unless we know the optimal parameter values for both topologies.

Doing an MCMC search both in the space of topologies and parameter values at the same time, results in an enormously big search space due to the large number of possible values for parameters. Instead, I developed a two-level optimization scheme, which involves a stochastic (MCMC) method and an iterative multidimensional grid search.

Let $\tilde{p}(t)$ be the set of parameter values which is the result of an iterative grid search optimization for a given topology t . The optimal model containing topology t is:

$$\tilde{m}(t) = m(t, \tilde{p}(t)) \quad (6.10)$$

The goal is to search in the space of possible topologies for the global minimum of the following objective function:

$$E(t) : t \rightarrow d(T^0, T^{\tilde{m}(t)}) \quad (6.11)$$

where $d(.,.)$ means the euclidean distance of the two trajectories.

Let t^* be the topology for which $E(t^*)$ is the global minimum of E . (More accurately, it is the topology with the lowest found energy.) Our prediction for the system is:

$$m(t^*, \tilde{p}(t^*)) \quad (6.12)$$

The above described method involves a two-level optimization procedure. We search in the space of topologies by an MCMC algorithm, and at each step of the MCMC, we run a grid search algorithm in order to evaluate the objective function. This hierarchical relationship of the two optimization cycles distinguishes it from the parallel optimization methods.

As presented in Section 5.3. the accuracy of the multidimensional iterative grid search algorithm depends on two key parameters (number of grid points and number of iterations). There is no need for very accurate grid search in the above described procedure. We have to set the parameters of the grid search only to reach an accuracy where the optimal topology has significantly lower energy than the other states.

6.3 Integrating biological knowledge

In case of noisy or incomplete datasets, partial knowledge about the system structure may help to reproduce the system as a whole. However, in many cases, our prior knowledge is uncertain, and we know, or at least we can estimate, the degree of its uncertainty. I represent the uncertainty of knowledge in a prior probability distribution, that can be used for Bayesian inference. The key question is how to integrate different kinds of biological knowledge from various sources into one Bayesian prior.

Suppose we have a list of statements about the structure of the system. Each statement may be true or false for the real system. Let s_i^+ be the probability by which we think the i^{th} statement is true, and $s_i^- = 1 - s_i^+$ be the probability by which we think it is false. These scoring values represent the uncertainty of our belief regarding the particular statement. We score an $m \in M$ model by checking all the statements on it, and multiplying the respective s^+ or s^- scores. So the overall $S(m)$ score of m is:

$$S(m) = \prod_i (s_i^+ I_i^+ + s_i^- I_i^-) \quad (6.13)$$

where $I_i^+ = 1$ if the i^{th} statement is true for the model, $I_i^+ = 0$, if not, and $I_i^- = 1 - I_i^+$

$S(m)$ is the joint probability of that the model is right, considering all statements.

Suppose we use the $p(T^0 | m) = \frac{1}{Z} e^{-\beta E(m)}$ Likelihood function to represent the probability of measuring T^0 given that the system is m . Although the most probable dataset is T^0 , the real trajectories may differ from T^0 due to the noise. The decrease in the probability of a trajectory depends on the $E(m)$ euclidean distance of T^m and T^0 . Using $S(m)$ as prior distribution, Bayes theorem is the following:

$$P(m | T^0) = \frac{1}{Z} e^{-\beta E(m)} S(m) = \frac{1}{Z} e^{-\beta(E(m) - \frac{1}{\beta} \ln(S(m)))} \quad (6.14)$$

Note, that finding the model with the highest posterior probability is equivalent to looking for the global minimum of an effective energy function:

$$\tilde{E}(m) = E(m) - \frac{1}{\beta} \ln(S(m)) \quad (6.15)$$

We can see, that \tilde{E} is the original objective function E modified by an additional term, which depends on the $S(m)$ score. If $S(m)=1$, the effective energy equals the original energy, but when $S(m)<1$, \tilde{E} is higher than E . The additional term may change the location of the global and local minima. In case of MCMC search, it is sensible to start from an initial state which corresponds to the maximum of the $S(m)$ prior distributions.

β is a scaling constant between the two terms in \tilde{E} : it determines the balance between the reliability of time series data and prior knowledge. The higher the β , the more we think that the maximum of $p(T^0 | m)$ alone gives the best model, and the less effect we let for $S(m)$ to alter the estimation.

Suppose we have a list of statements, which certainly have to be true for an appropriate model: $\forall i : s_i^{\dagger} = 1$. Then the posterior probability of a model may be either 0 or 1. $P(m | T^0)$ is 1, if the model satisfies all the listed statements, and it is 0, if at least one statement is false. If possible, defining a proposal function that automatically avoids models of zero probability, could speed up the simulation. In other words: deterministic usage of knowledge, as a special case of probabilistic knowledge, means searching only in the subspace of models which satisfy all requirements. It requires an appropriate proposal function, but the objective function remains $E(m)$.

6.3.1 Additional comments

- Prior knowledge on the system structure may mean for example a list of experimentally verified interactions (possibly collected from different papers). Experiments may give information about the existence and strength (rate constant) of interactions. Correlations between the time-course behavior of two components may also give an a priori hypothesis about the system topology. In gene regulatory networks, promoter sequence similarities may be used to hypothesize unknown hyperedges.
- Knowledge may also come from any known general feature of the studied system structure. For example we may assume that a subset of the components has synthesis and degradation, which depend on only the concentration of the component itself.
- Knowledge may be transferred between evolutionarily related systems. If System A and System B contain homologue components, we can also expect conserved interactions between them. Let's say, the interaction between component x and y in System A is experimentally verified, but we don't know anything about the existence of interaction between the homologue components x' and y' in System B. However we can transfer this knowledge about the interaction from System A to System B, and estimate the uncertainty of it using the degree of homology between the respective components.
- The scoring framework is even more general: We may have knowledge about any property of the system, that can be quantified and measured on a candidate model. For instance, knowledge on the stability/robustness of the system, or considerations how the system must behave if it starts from a different initial condition etc.
- When integrating knowledge of different reliability or importance, the logarithms of scores may be weighted in \tilde{E} to tune their relative influence on the results.

Chapter 7

The algorithm

In this section I present a new systems identification algorithm, that is based on the ideas mentioned in the previous sections. First, I give the exact scheme of the algorithm, and then I explain its details in the description part.

7.1 The main algorithm

```
Input:  $T^0$ 
for i = 1:n {
    for c=1: $c_{max}$  {
         $(m_1, \dots, m_k) = \text{SEARCH}(i, c)$ 
         $(m_1, \dots, m_k) = \text{FINETUNING}((m_1, \dots, m_k))$ 
         $m^*(i, c) = m_i : E(m_i) = \min\{E(m_1), E(m_2), \dots, E(m_k)\}$ 
    }
     $m_i^{**} = m^*(i, c) : E(m^*(i, c)) = \min\{m^*(i, 1), m^*(i, 2), \dots, m^*(i, c_{max})\}$ 
     $m_i^{**} = \text{REMOVE}(t, m_i^{**})$ 
}
Output =  $m^{**}$ 
```

7.2 Description

The input of the algorithm is the T^0 time series data, and the output is a prediction (m^{**}) for the ODE system underlying the dynamic process. According to Section 6.2.1., we infer the interactions of each of n components independently, so the algorithm performs an i=1:n cycle, repeating the same series of steps for each components. For a given component we infer the optimal set of interactions for different values of complexity (c). We try all c values in a range of 1 to c_{max} .

For a given node (i) and for a fixed complexity (c), we run a search algorithm. $\text{SEARCH}(i, c)$ may be a Brute force search or an MCMC search, according to the size of the search space.

SEARCH-BF (Brute Force method):

Evaluating the $E(m)$ function for all different combination of hyperedges
(The number of edges is c). Selecting the k best models.

SEARCH-MCMC (Markov chain Monte Carlo method):

```

 $m = m_0$ 
for i=1:steps {
 $m' = \text{propose}(m)$ 
  if  $E(m') < E(m) \rightarrow m=m'$ 
  else if  $\text{random} < e^{-\beta(E(m')-E(m))} \rightarrow m=m'$ 
  Save  $m$  if it is in the top  $k$  samples
}

```

The search space here is the topology space, in which every topology is parametrized by the optimal parameter values. These optimal constants are obtained from an iterative grid search algorithm. Therefore the optimization in the parameter space is embedded into each steps of the topology optimization. For more description of this two-level optimization, see Section 6.2.3.

The output of SEARCH(i,c) is the list of the most optimal models identified. We perform a fine tuning step for each models in this list. Fine-tuning involves an MCMC optimization only of the parameter values keeping the set of interactions fixed. Then we select the best model called $m^*(i,c)$ from the fine-tuned models. After repeating the above-described steps for all complexity values, we select the best m^* model (m^{**}) obtained from these trials.

In order to avoid overfitting, we remove all the hyperedges from m^{**} , whose parameter value is lower than a certain threshold (t). The remaining hyperedges are make our prediction for the interactions influencing the studied component. Repeating the same steps for each components, we can construct the final prediction for the system.

The following table shows the size of the topology space for different system sizes (number of components) and different complexities. Highlighted values indicate that Brute force optimization is feasible in a three-hour long simulation (using 4 grid points and 7 grid search iterations).

Components	c = 1	c = 2	c = 3	c = 4	c = 5	c = 6
3	6	15	20	15	6	1
4	10	45	120	210	252	210
5	15	105	455	1365	3003	5005
6	21	210	1330	5985	20349	54264
7	28	378	3276	20475	98280	376740
8	36	630	7140	58905	376992	1947792
9	45	990	14190	148995	1221759	8145060
10	55	1485	26235	341055	3478761	28989675
11	66	2145	45760	720720	8936928	90858768
12	78	3003	76076	1426425	21111090	$2.57 * 10^8$
13	91	4095	121485	2672670	46504458	$6.67 * 10^8$
14	105	5460	187460	4780230	96560646	$1.61 * 10^9$
15	120	7140	280840	8214570	$1.91 * 10^8$	$3.65 * 10^9$
16	136	9180	410040	13633830	$3.6 * 10^8$	$7.86 * 10^9$
17	153	11628	585276	21947850	$6.54 * 10^8$	$1.61 * 10^{10}$
18	171	14535	818805	34389810	$1.15 * 10^9$	$3.18 * 10^{10}$
19	190	17955	1125180	52602165	$1.96 * 10^9$	$6.03 * 10^{10}$
20	210	21945	1521520	78738660	$3.24 * 10^9$	$1.11 * 10^{11}$

7.3 Additional comments

- Euler solver was used, but it can be easily replaced by a Runge-Kutta solver.
- In SEARCH-MCMC, propose(m) stands for the proposal function which generates a new model by modifying the actual model m. It removes a hyperedge and adds a new one in its place. The complexity of the node therefore remains constant.
- The benefit of Brute Force search is that it isn't trapped by local minima, since it checks all possible models. It's disadvantage is that it is too slow, and sometimes not feasible for large systems.
- The success of the algorithm highly depends on the two parameters of the grid search method. More grid points improve the result, but it boosts up the runtime. A good balance needed, when the resolution of grid points enables to find the best topology in a reasonable time. (At the end the fine-tuning step will correct the inaccuracies of parameter values.)
- There may be several ways to improve the algorithm. Instead of optimizing it, I concentrate on the question, how prior knowledge can improve the performance of this simple method.

Chapter 8

Results

8.1 Random synthetic systems

I have created four synthetic (in numero) datasets to test my method. The testing procedure involved the following steps:

- 1. Randomly build a systems of a given complexity**
- 2. Generate a trajectory from the synthetic system**
- 3. Add Gaussian noise to the trajectory**
(add $r \in N(0, \sigma)$ independent random values to each point of the trajectory)
- 4. Treat the trajectory as a time series dataset to infer back the system**
- 5. Compare the inferred model with the original system**

The advantage of such an assessment is that we know what result to expect from the algorithm, in contrast with the case of real biological systems. I will measure the goodness of a result in terms of the number of false positive and false negative interactions. These can be counted for a whole model or for only a given component.

Constructing random systems was not a trivial task, because many system structure results in a diverging trajectory. Therefore I made an MCMC algorithm that searches in the space of possible systems using a scoring function, which give low score for bounded- and high scores for non-bounded trajectories. On bounded trajectory I mean a trajectory, which remain in a given finite range along the whole time-interval. Starting for an arbitrary initial topology, this algorithm found non-divergent systems.

To summarize my observations, only a very small subset of all possible topologies satisfies the non-divergence criteria. (It took significant time to find such systems.) This may have interesting consequences for the evolution of biological systems, where divergence may be disadvantageous or lethal. It's important to note that the topologies I constructed are entirely random, and don't incorporate typical topological features of real systems.

First I present an example, the synthetic system called M10A, containing 10 components and 19 interactions. Here are the ODEs of the system:

$$\frac{dx_1}{dt} = 0.22 * x(5) * x(8)$$

$$\frac{dx_2}{dt} = -1.19 * x(8) - 0.66 * x(2) - 0.59 * x(1) * x(7)$$

$$\frac{dx_3}{dt} = -1.03 * x(3) + 0.75 * x(7) * x(2)$$

$$\frac{dx_4}{dt} = 1.11 * x(5) * x(6) + 0.95 * x(9) * x(7)$$

$$\frac{dx_5}{dt} = -2.01 * x(9) * x(8) + 1.8 * x(5) * x(7) - 0.18 * x(4) * x(10)$$

$$\frac{dx_6}{dt} = 0.58 * x(5) * x(9)$$

$$\frac{dx_7}{dt} = -1.24 * x(10) * x(8) - 1.41 * x(10) * x(7) + 0.26 * x(10)$$

$$\frac{dx_8}{dt} = 0.69 * x(7) * x(10)$$

$$\frac{dx_9}{dt} = 2.07 * x(8) * x(5)$$

$$\frac{dx_{10}}{dt} = 3.01 * x(2) * x(10) - 0.74 * x(9)$$

Figure 8.1. shows the hypergraph topology of model M10A. Green hyperedges represent promotion, red hyperedges represent inhibition. Many component have negative feedback or self inhibition in order to avoid divergence.

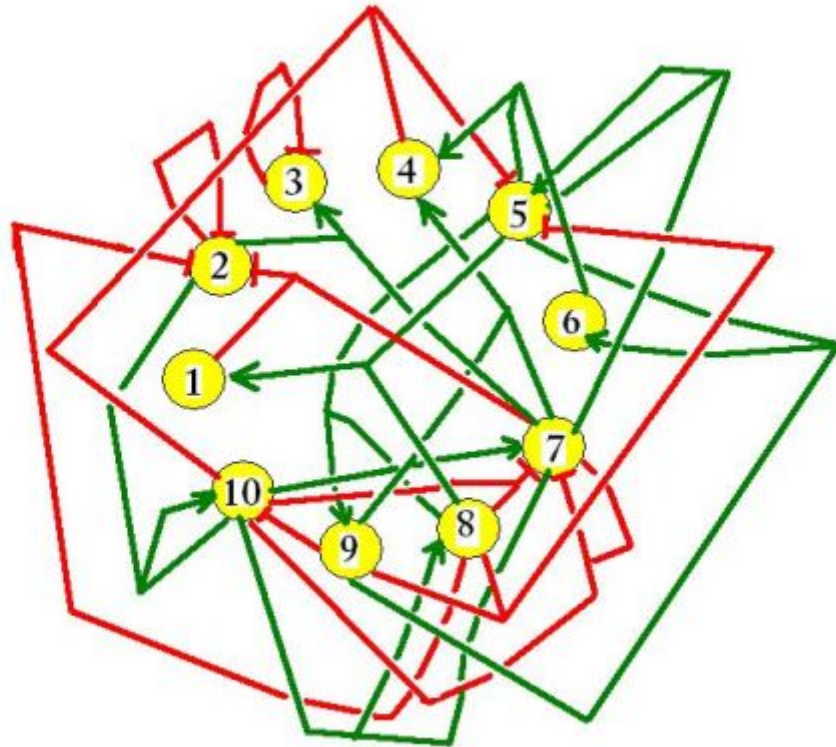


Figure 8.1: Topology of system M10A

Figure 8.2. shows the trajectories generated from this system. The initial values of each variables are all 1. The system converges into an equilibrium. Figure 8.3. shows a noisy trajectory. Gaussian noise ($\sigma = 0.05$) was added to each point of the original curve.

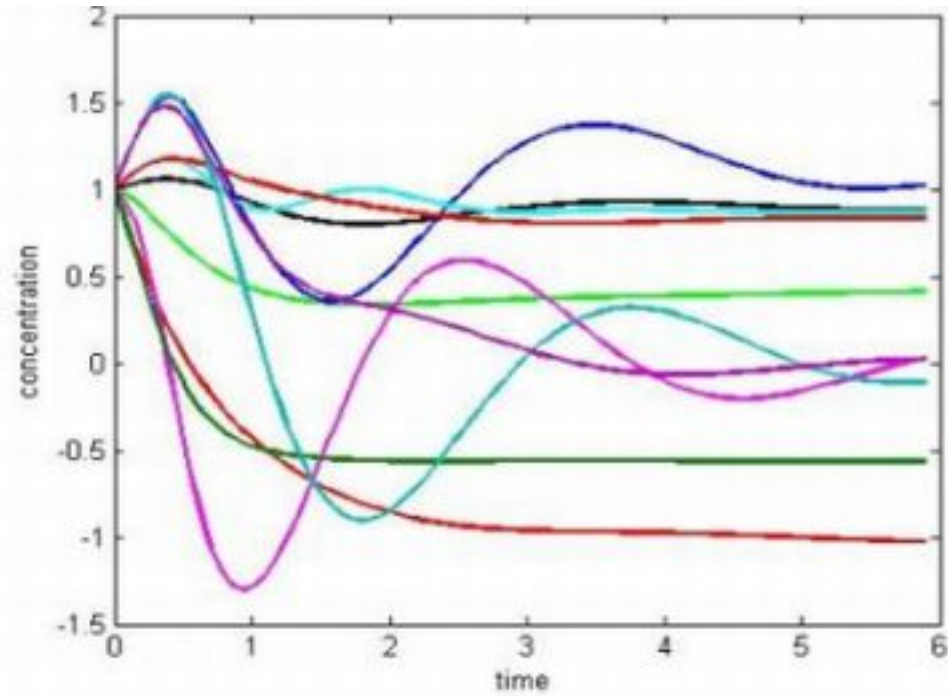


Figure 8.2: Trajectories generated from the model M10A without noise

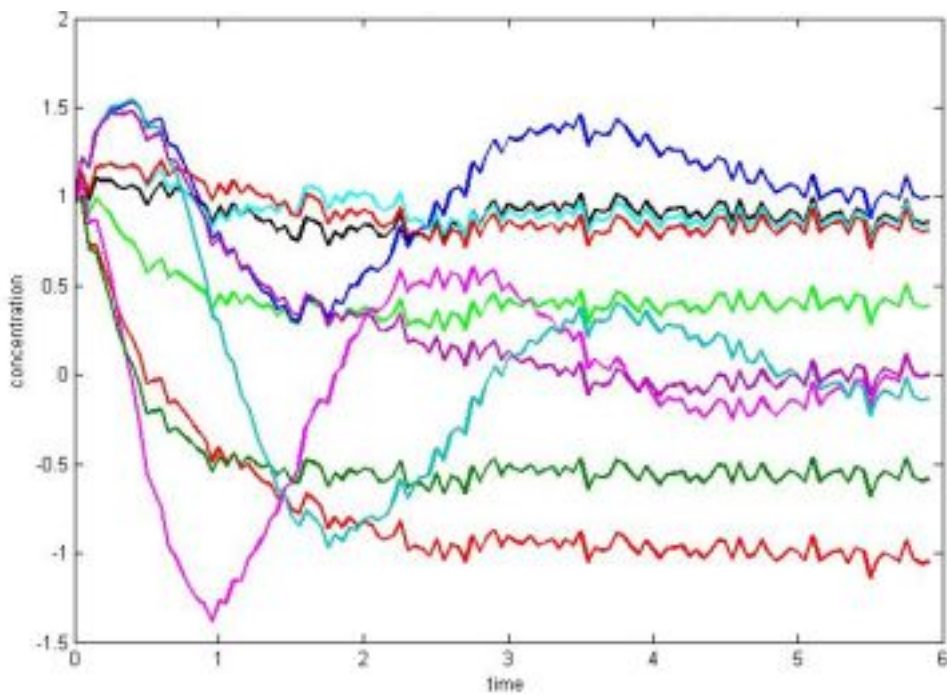


Figure 8.3: Noise ($\sigma = 0.05$) added to the trajectory of model M10A

The following table summarizes the important properties (number of components, number of interactions, number of nodes with complexity 1, 2, 3 or 4) of the four synthetic systems created:

Model ID	Components	Interactions	c(x)=1	c(x)=2	c(x)=3	c(x)=4
M6	6	13	1	3	2	0
M8	8	19	2	1	5	0
M10A	10	19	4	3	3	0
M10B	10	25	2	1	7	0

8.2 Inferring synthetic systems

The next table shows the results of inferring the above-listed 4 synthetic systems. No noise were added to the trajectories. The algorithm required slightly different settings to find the best prediction for these systems. The number of grid points was 7 and the number of grid search iterations was 5 for system M6, while systems M8, M10A and M10B required 4 grid points and 7 grid search iterations. I used Brute force search for systems M6, M8 and M10B, but an MCMC search (steps=3000, $\beta = 500$) for M10A.

In the table the goodness of the results is expressed by the sensitivity ($\frac{(True+)}{(True+)+(False-)}$) and the selectivity ($\frac{(True-)}{(True-)+(False+)}$) for interactions. The number of perfect nodes (with no False+, no False-) is also given, as well as the average relative difference (ARD) of the estimated and the original constants of each true positive interactions.

Model ID	Sensitivity	Selectivity	Perfect nodes	ARD
M6	0.92	0.99	5 (from 6)	0.03
M8	0.74	0.99	5 (from 8)	0.14
M10A	0.84	0.99	8 (from 10)	0.14
M10B	0.92	0.99	8 (from 10)	0.03

The results show that the algorithm can identify interactions with high sensitivity and very high specificity, although no prior knowledge was used in these simulations. It also estimates the constants of the system very accurately.

8.3 Noisy data and the help of knowledge

In the followings I will concentrate on Node 2 of model M10A by inferring the set of interactions affecting this component. All the following simulations involved a Brute force search with 4 grid points and 7 grid search iterations. This node has 3 interactions, so the number of false positives and false negatives may be 0, 1, 2 or 3.

When I added $\sigma = 0.05$ noise to the trajectories, only 4 per cent of the 25 simulations (performed on repeatedly generated noisy datasets) have given perfect result (false

pos.=0/false neg.=0) and 44 per cent of simulations resulted 3 false positives/3 false negatives.

To check if deterministic knowledge helps, and to *quantify* its benefit, I performed 25 Brute Force simulations on repeatedly generated noisy datasets in case of 1 or 2 known hyperedges. The score of these statements on the presence of hyperedges were $s_i^+ = 1$. Figure 8.4. presents the three histograms of the results.

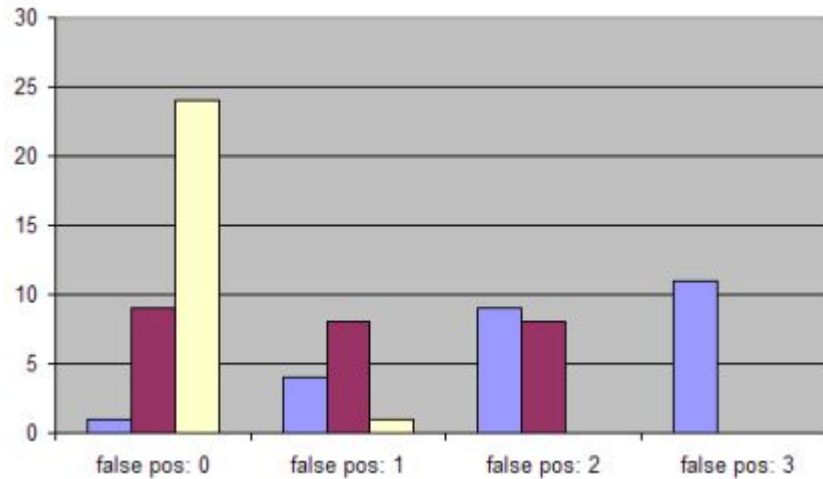


Figure 8.4: Histogram of the numbers of false positives, from 25 Brute force simulations inferring the interactions of Node 2, using noisy data ($\sigma = 0.05$). Blue histogram means no knowledge, purple means 1 known edge, white means 2 known edges from the true 3 edges.

Deterministic knowledge increased the goodness of results: when 1 edge was known, 36 per cent of the simulations resulted 0 false positive. When 2 edges were known, this ratio raised to 96 per cent. Since each simulation involved Brute Force search, the improvement of results was not caused by the fact that we started from an initial state which is close to the global minimum. Instead, deterministic knowledge restricted the search to a subspace, in which the minimum of the objective function corresponds to the real system, even if it is not the global minimum in the whole search space. The global minimum may correspond to a false model overfitted to noise.

To test how robust are these results to noise I run 6x20 simulations for different noise levels, considering 2 deterministically known interactions. In this case the number of false positives may be 0 or 1. Figure 8.5. shows the histograms of the results. We can see that as the noise level is increasing the reliability of the method is getting worse.

8.4 Using \tilde{E} for uncertain knowledge

In the followings I will concentrate on Node 10 of system M10A, that has two interactions. Suppose we have prior knowledge on one interaction, but this knowledge is uncertain. I

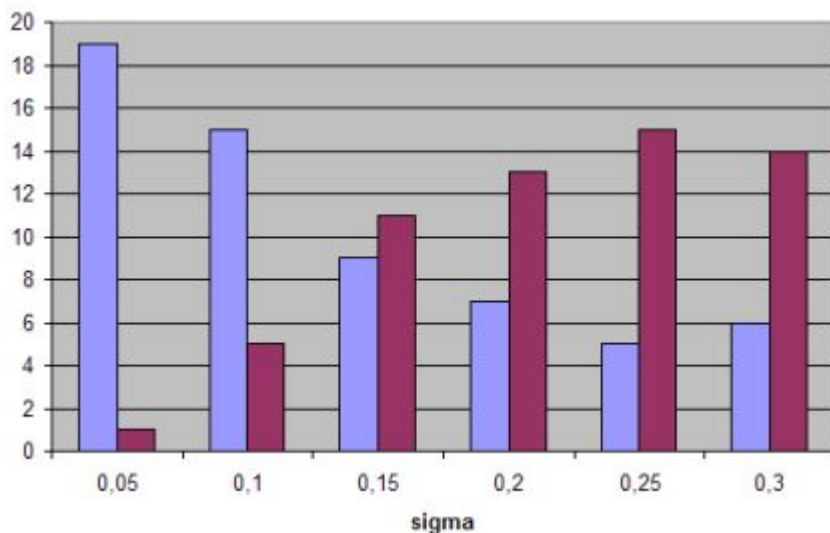


Figure 8.5: Histogram of the numbers of false positives, from 20 Brute force simulations inferring the interactions of Node 2, at increasing noise levels (from $\sigma = 0.05$ to $\sigma = 0.3$). Blue histogram means 0 false positive, purple means 1 false positive.

modeled this situation by randomly deciding the verity of prior knowledge. The probability of selecting a true interaction was $p=0.7$, while the probability of choosing a random false interaction was 0.3.

I generated and saved a noisy dataset ($\sigma = 0.2$), and every simulations were performed with it. When no knowledge was used, the algorithm could identify only one of the two interactions ((False+)=1,(False-)=1)).

100 simulations were performed with the uncertain ($p=0.7$) knowledge considered as certain. It means that the score of the hyperedge was $s^+ = 1$ (deterministic knowledge). Knowledge was randomly re-generated for each simulation. Another 100 simulations were done using the effective objective function \tilde{E} with an $s^+ = 0.7$ score for the selected interaction. Therefore in this case the uncertainty of knowledge was well-represented in the prior distribution.

Each simulation involved Brute force search, using 4 grid points and 7 grid search iterations. c_{max} was set to 2. The scaling β constant in \tilde{E} was set to 18.

Figure 8.6. shows that the use of the effective objective function has given better results in incorporating uncertain knowledge than the deterministic approach. While both resulted 29 perfect predictions ((False+)=0,(False-)=0)), the number of absolute wrong predictions ((False+)=2,(False-)=2)) was 20 versus 38 in the non-deterministic and deterministic simulations respectively. Even if we consider only the expected value (30) of the absolute wrong predictions in deterministic simulations, there is still 10 per cent

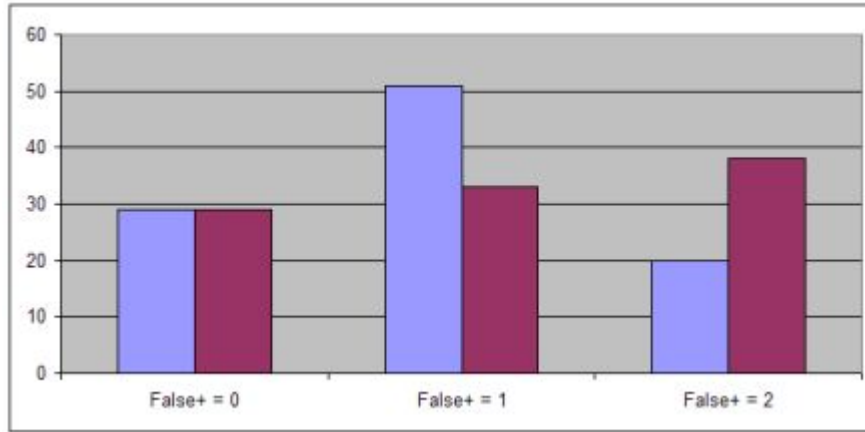


Figure 8.6: Histogram of results (number of false positives) of 2x100 Brute force simulations to incorporate uncertain ($p=0.7$) prior knowledge. Blue histogram means simulations using the effective objective function ($s^+ = 0.7$), purple histogram means simulations with deterministic knowledge ($s^+ = 1$).

difference between the two results. It suggests that the non-deterministic method may be reliable even if the prior knowledge is wrong. (This is the consequence of taking into account the uncertainty of prior knowledge.)

Many more experiments are needed however to decide if the method is useful in biological systems identification. This example was just a simple illustration for my idea.

Chapter 9

Conclusions

A new system identification algorithm was introduced in this report. Although there may be many ways to improve the method, it proved to be good enough to infer synthetic mass action systems of 10 components. The method uses time series data as input, and gives a prediction for the underlying differential equations as output.

The algorithm has limitations in the size and the complexity of systems. It can make models in a reasonable time for only sparse systems of maximum 10-20 components. Optimizing the algorithm may result in a broader spectrum of inferable systems.

The basic inference algorithm was constructed in order to be able to test if prior knowledge (in either deterministic or probabilistic way) can help to solve the reverse engineering problem.

My first results show that knowledge indeed improves the performance of the algorithm, even in case of noisy data. I proposed a general mathematical framework to integrate prior knowledge from different sources. It enables to represent the uncertainty of our knowledge in scoring functions, and integrate these scores in a prior probability distribution. It results in a modified objective function, which sums information both from time series data and from prior knowledge. The minimization of this modified objective function indeed resulted better result than using the original objective function.

Future research direction may include the optimization of the inference algorithm, making it capable to analyze incomplete datasets, designing weighting schemes for the integration of knowledge and studying how to transfer knowledge between evolutionary related systems.

Once calibrated properly on synthetic systems, the method should be tested on real biological systems using time series measurements and different sources of prior knowledge. The promising results on synthetic datasets suggest that the method may provide a useful tool in modeling biological systems.

References

- [1] Ross et al. *Determination of Complex Reaction Mechanisms: Analysis of Chemical, Biological, and Genetic Networks* 2006.
- [2] Jeong et al. *The large-scale organization of metabolic networks* 2000. Nature 407, p651 - 654
- [3] Giot et al. *A Protein Interaction Map of Drosophila melanogaster* 2003. Science Vol. 302. no.5651, pp.1727-1736
- [4] Yook et al. *Functional and topological characterization of protein interaction networks* 2004. Proteomics Vol. 4, no. 4, pp.928-942
- [5] Balázsi et al. *Topological units of environmental signal processing in the transcriptional regulatory network of Escherichia coli* 2005. PNAS Vol. 102 no. 22, p7841-7846
- [6] Palla et al. *Uncovering the overlapping community structure of complex networks in nature and society* 2005. Nature, Vol. 435, Issue 7043, pp.814-818
- [7] Vance et al. *Determination of causal connectivities of species in reaction networks* 2002. PNAS Vol. 99. no. 9., p5816-5821
- [8] Tegnér et al. *Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling* 2003. PNAS vol.100 no.10 p5944-5949
- [9] Fell *Understanding the Control of Metabolism* 1997.
- [10] Arkin and Ross *Statistical Construction of Chemical Reaction Mechanisms from Measured Time-Series* 1995. J. Phys. Chem. Vol. 99, p970-979
- [11] Peifer and Timmer *Parameter estimation in ordinary differential equations for biochemical processes using the method of multiple shooting* 2007. IET Syst Biol., March;1(2):78-88
- [12] Banga et al. *Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods* 2003. Genome Res. 13:2467-2474
- [13] Schlitt and Brazma *Modelling gene networks at different organization level* 2005. FEBS Letters Volume 579, Issue 8, p1859-1866

- [14] Schlitt and Brazma *Current approaches to gene regulatory network modelling* 2007. BMC Bioinformatics, 8(Suppl 6): S9
- [15] Dassow et al. *The segment polarity network is a robust developmental module* 2000. Nature, Vol. 406, Issue 6792, pp.188-192
- [16] Novák and Tyson *Modeling the control of DNA replication in fission yeast* 1997. Proc. Natl. Acad. Sci. USA 94: pp.9147-9152.
- [17] Imoto et al. *Combining Microarrays and Biological Knowledge for Estimating Gene Networks via Bayesian Networks* 2004. J Bioinform Comput Biol. Mar;2(1):77-98.
- [18] Pilpel et al. *Identifying regulatory networks by combinatorial analysis of promoter elements* 2001. Nature Genetics Vol. 29, no2, pp. 153-159
- [19] Segal et al. *From promoter sequence to expression: a probabilistic framework* 2002. Proc. of the 6th ann. internat. conference on Computational Biology
- [20] Masys *Linking microarray data to the literature* 2001. Nature Genetics Vol. 28(1):9-10
- [21] Bussemaker et al. *Regulatory element detection using correlation with expression* 2001. Nature Genetics Vol. 27, pp.167-174
- [22] Julius et al. *Genetic Network Identification Using Convex Programming* 2007. (submitted)
- [23] Todorovski et al. *Integrating domain knowledge in equation discovery* 2007. Computational Discovery of Scientific Knowledge, Vol. 4660/2007 p69-97
- [24] Ljung *System Identification: Theory for the User* 1999.
- [25] Katayama *Subspace Methods for System Identification* 2005.
- [26] Pintelon and Schoukens *System identification: a frequency domain approach* 2001.
- [27] Greblicki and Pawlak *Non-Parametric System Identification* 2008.
- [28] Lambert *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem* 2001.
- [29] Liu *Monte Carlo Strategies in Scientific Computing* 1991.
- [30] Earl and Deem *Parallel Tempering: Theory, Applications, and New Perspectives* 2005. Phys. Chem. Chem. Phys., 7, pp.3910-3916,
- [31] Kirkpatrick et al. *Optimization by Simulated Annealing* 1983. Science Vol. 220. no.4598, pp.671-680

- [32] Wenzel and Hamacher *Stochastic Tunneling Approach for Global Minimization of Complex Potential Energy Landscapes* 1999. Physical Review Letters, Vol. 82, Issue 15, pp.3003-3007
- [33] Kim *Iterated Grid Search Algorithm on Unimodal Criteria* 1997. PhD Dissertation (Virginia Polytechnic Institute and State University)
- [34] Bolstad *Introduction to Bayesian Statistics* 2007.
- [35] Gallo et al. *Directed hypergraphs and applications* 1993. Discrete applied mathematics Vol 42, Issue 2-3, Special issue: combinatorial structures and algorithms p177-201
- [36] D'haeseleer et al. *Genetic Network Inference: From Co-Expression Clustering to Reverse Engineering* 2000. Bioinformatics Vol. 16 no. 8 2000 p707-726
- [37] Reka Albert *Scale-free networks in cell biology* 2005. Journal of Cell Science 118, pp4947-4957
- [38] Gutenkunst et al. *Universally Sloppy Parameter Sensitivities in Systems Biology Models* 2007. PLoS Comput Biol 3(10): e189.
- [39] Wolkenhauer *Systems biology: Dynamic pathway modelling* 2005.
- [40] Palsson *Systems biology: Properties of reconstructed networks* 2006. Bioinformatics Vol. 16 no. 8 2000 p707-726
- [41] Wilkinson *Bayesian methods in bioinformatics and computational systems biology* 2007. Briefings in Bioinformatics Vol 8 No 2 pp109-116

Chapter 10

Technical notes

The system identification algorithm introduced in this report was implemented in a computer program written in Java SE 6 in the Eclipse IDE 3.3.2 development environment. MATLAB R2007a was used to visualize the results and to produce figures for the report. Simulations were run on a 2.2 GHz processor.

Appendix 1

Since the presented algorithm is based on the Iterative Grid Search method, its success highly depends on the shape of the objective function in the parameter space of a given topology (set of interactions). If the function has only one global minimum and no local minima, the algorithm will converge to the global minimum point. However, if the objective function has several local minima, we may need to use more grid points, and there is still no guarantee for finding the global minimum.

I present the 2D map of the objective function in the parameter space for Node 2 of system M10A. This component has 3 interactions. Figure 1 shows the case when the parameter of the first interaction was considered constant, and the other two parameters varied in a range of $[-4,4]$. (100x100 points, 0 represents -4, 100 represents 4) The color shows the value of the objective function for the given parameter combination. Figure 2 and 3 show the landscape of the objective function when the parameter of the second and the third interaction was constant. Figure 4-9 present the same for Node 5 and Node 7.

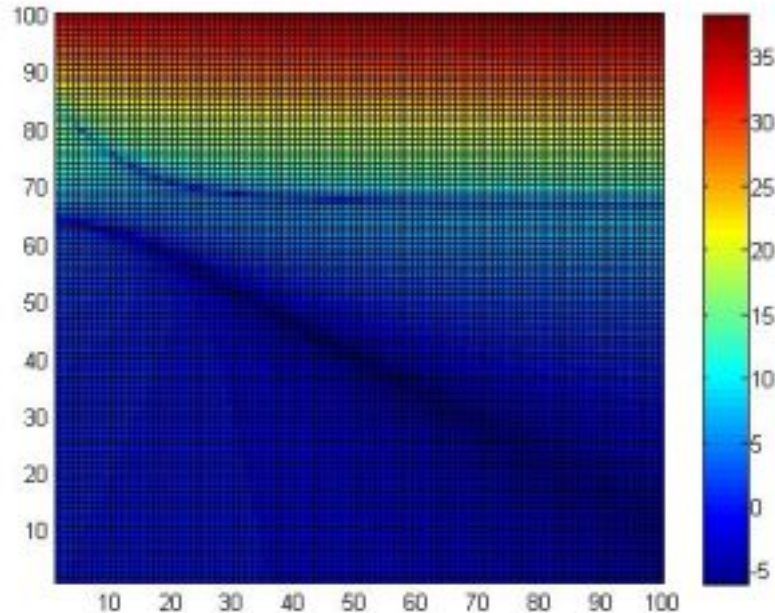


Figure 1: System M10A, Node 2, x-axis: 2^{nd} parameter, y-axis: 3^{rd} parameter

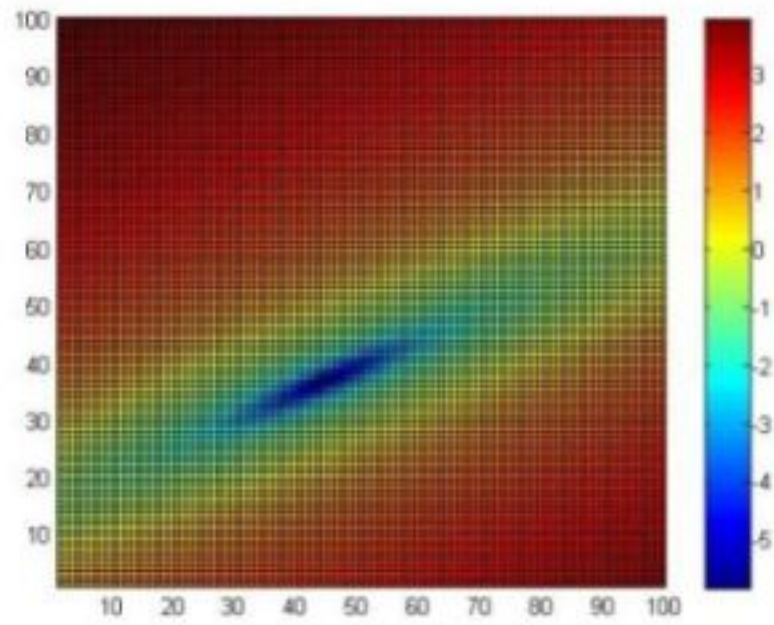


Figure 2: System M10A, Node 2, x-axis: 1st parameter, y-axis: 3rd parameter

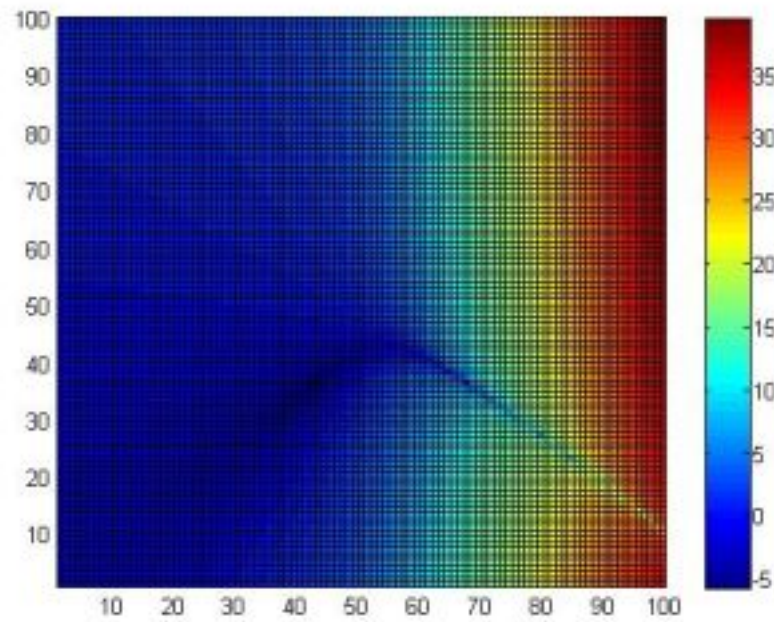


Figure 3: System M10A, Node 2, x-axis: 1st parameter, y-axis: 2nd parameter

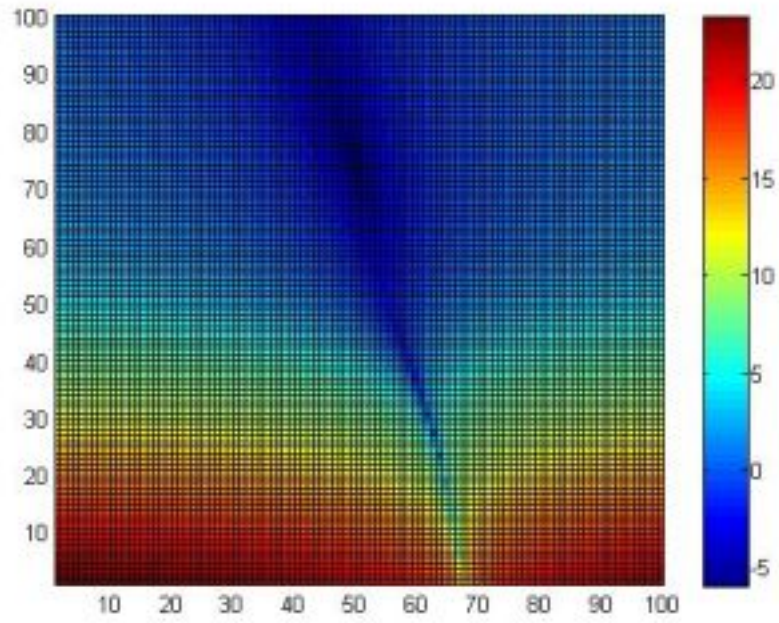


Figure 4: System M10A, Node 5, x-axis: 2nd parameter, y-axis: 3rd parameter

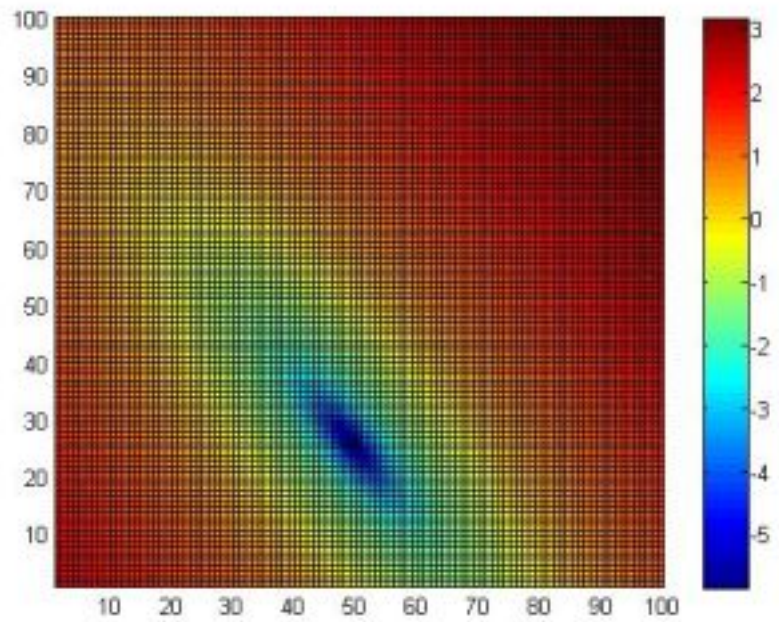


Figure 5: System M10A, Node 5, x-axis: 1st parameter, y-axis: 3rd parameter

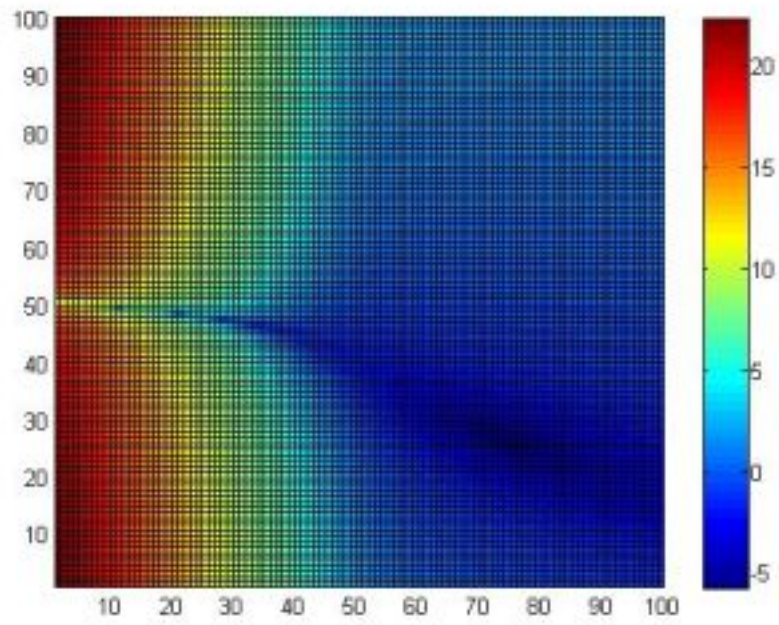


Figure 6: System M10A, Node 5, x-axis: 1st parameter, y-axis: 2nd parameter

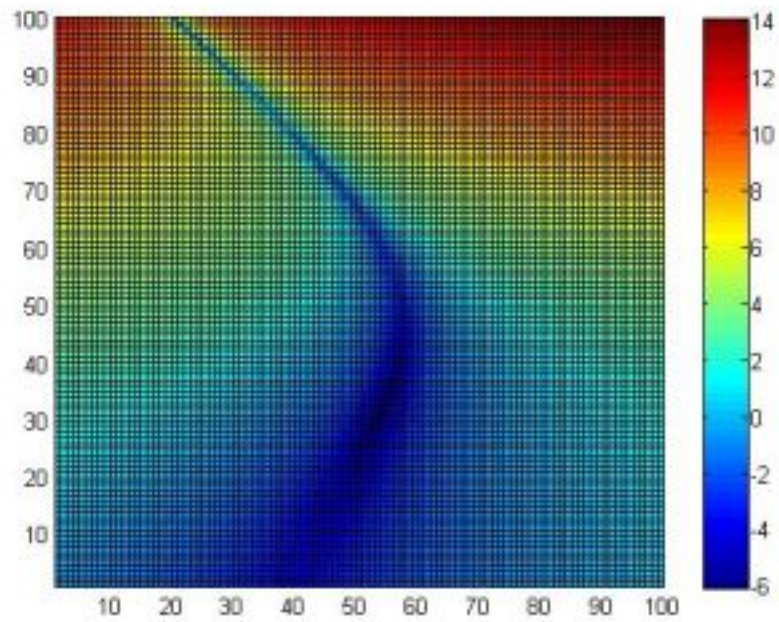


Figure 7: System M10A, Node 7, x-axis: 2nd parameter, y-axis: 3rd parameter

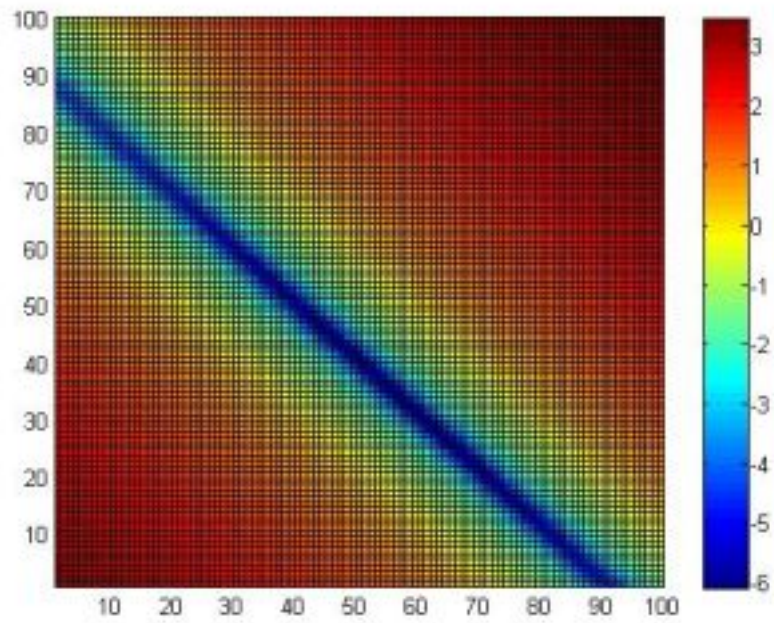


Figure 8: System M10A, Node 7, x-axis: 1st parameter, y-axis: 3rd parameter

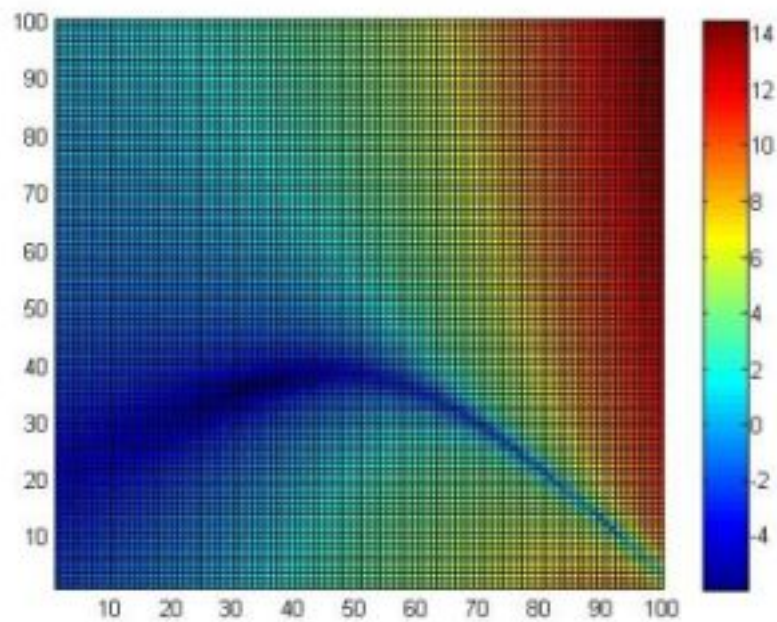


Figure 9: System M10A, Node 7, x-axis: 1st parameter, y-axis: 2nd parameter

Appendix 2

The underlying system may not be unique

Initial Value Problem (IVP) aims to find the $X(t)$ solution of the following ODEs:

$$dX(t) = f(X(t))dt \quad (1)$$

where the initial value $X(0) = x$ is specified.

(Here X is an n -dimensional vector and f is a $\mathbf{R}^n \rightarrow \mathbf{R}^n$ function.) Function $f(X(t))$ gives the tangent of the trajectory at the point $X(t)$.

According to the Picard-Lindelöf theorem, the solution of this problem exists and is unique, in case of f is a Lipschitz function. Effective numerical methods have been developed for approximating the solution of the Initial Value Problem. (See in Section 5.1.)

However, the solution of an ODE system determines the values of tangent only at the points of the trajectory, and not at all point in the n -dimensional space. Therefore values of function f are also defined only along the known trajectory, and not globally. Let's define a new Lipschitz function, called g , for which $g = f$ at the points of the trajectory, but $g \neq f$ at some other points of the space. Now we can consider the following ODE:

$$dY(t) = g(Y(t))dt \quad (2)$$

Using initial condition $Y(0) = x$, we get the same trajectory for $Y(t)$ as for $X(t)$. However, starting the two systems from an other initial condition, which doesn't belong to the trajectory, $X(t)$ and $Y(t)$ may differ. We can conclude that it is possible to construct different ODE systems which lead to the same trajectory for a given initial condition. The opposite of the Picard-Lindelöf theorem is not true: a trajectory doesn't necessarily determines a unique ODE system. Therefore we may find false (non-biological) systems which fit the data perfectly.

Sub-optimal trajectories

The number of ODE systems, which generate the same trajectory, is the number of functions, that have the same values along the trajectory. Let's measure the goodness of an ODE system by the similarity of its trajectory to a template trajectory according to the Euclidean distance. The number of equation systems, whose trajectory is better than a certain threshold c can be written in the following way:

$$N(c, T_0) = \{m \in M \mid d(T_m, T_0) \leq c\} \quad (3)$$

where M is the whole set of possible ODEs, T_m is the trajectory of m , T_0 is the template trajectory, and d stands for the Euclidean distance.

Let $T(c, T_0)$ be the set of trajectories in the space which are closer to T_0 than c .

$$T(c, T_0) = \{T_m \mid d(T_m, T_0) \leq c\} \quad (4)$$

Then we can write:

$$N(c, T_0) = \sum_{t \in T(c, T_0)} N(0, t) \quad (5)$$

This sum grows rapidly if we increase the value of c . It therefore suggests that there may be a great number of ODE systems, which lead to sub-optimally fitting solutions. These systems correspond to some local minima of the $m \rightarrow d(T_m, T_0)$ $m \in M$ function.

The case of incomplete data

Suppose that we want to reverse-engineer a dynamic system with n components, but we have time-series data of only $m \leq n$ variables. It means that we have measured a trajectory in a lower-dimension subspace of the original space. The measured T_0 trajectory is a result of a $P : \mathbf{R}^n \rightarrow \mathbf{R}^m$ projection of the original T solution of the underlying ODE system. Let $T^*(c, T_0)$ be the set of trajectories in the m -dimensional subspace which are closer to T_0 than c . Here we use d^* distance measure, which is the m -dimensional restriction of d .

$$T^*(c, T_0) = \{T_m \mid d^*(T_m, T_0) \leq c\} \quad (6)$$

Let $F(T)$ be the set of all trajectories in the original n -dimensional space, which gives T , if projected to the m -dimensional subspace:

$$F(T) = \{t \in \mathbf{R}^n \mid P(t) = T\} \quad (7)$$

Then the total number of trajectories whose projection are within a distance of c to T_0 :

$$G(c, T_0) = \sum_{t \in T^*(c, T_0)} F(t) \quad (8)$$

An ODE system with an initial condition can generate only one solution (Picard-Lindelöf), so the number of different ODEs which lead to solutions being within a distance of c to T_0 , is at least $G(c, T_0)$. Since we measure the goodness of a system by the m -dimensional Euclidean distance, this multiplicity causes deep minima in the function, besides the minimum corresponding to the real solution.

Summary

- The uniqueness of the solution in the reverse-engineering problem is not guaranteed. However, if we search only in a specified class of ODE system, a solution may be unique within the class.
- Even if there were a unique solution, a number of false solution may also exist due to the large number of sub-optimal trajectories.
- If we know the trajectories of only a subset of variables, this uncertainty may result in additional false solution.
- The effects listed above may result in a number of local (or global) minima of the objective function.