ODEion - a software module for structural identification of ordinary differential equations

Peter Gennemark[1] and Dag Wedelin[2]

Abstract

In the systems biology field, algorithms for structural identification of ordinary differential equations (ODEs) have mainly focused on fixed model spaces like S-systems and/or on methods that require sufficiently good data so that derivatives can be accurately estimated. There is therefore a lack of methods and software that can handle more general models and realistic data.

We present ODEion, a software module for structural identification of ODEs. Main characteristic features of the software are:

- The model space is defined by arbitrary user-defined functions that can be non-linear in both variables and parameters, such as for example chemical rate reactions.
- ODEion implements computationally efficient algorithms that have been shown to efficiently handle sparse and noisy data. It can run a range of realistic problems that previously required a supercomputer.
- ODEion is easy to use and provides SBML output.

We describe the mathematical problem, the ODEion system itself, and provide several examples of how the system can be used.

Availability: www.odeidentification.org

Keywords: system identification, ordinary differential equations

[1]Mathematical sciences, University of Gothenburg; Department of Mathematics, Uppsala University; peter.gennemark@astrazeneca.com

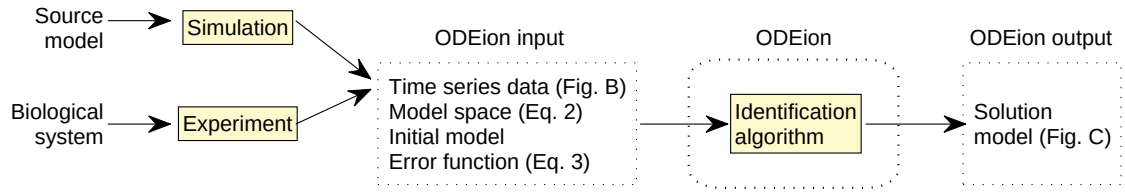
[2]Department of Computer Science and Engineering, Chalmers University of Technology; dag@chalmers.se

1 Introduction

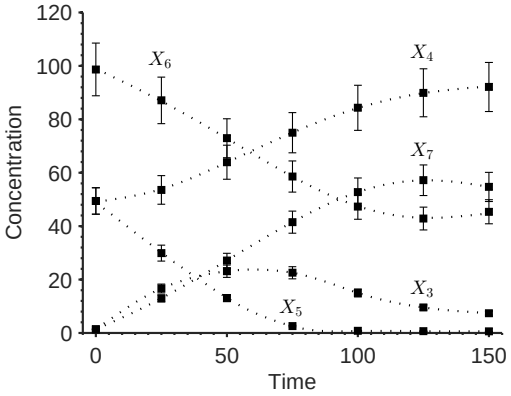
Identification of both the structure (the form of the equations) and the parameters of ordinary differential equations (ODEs) is a fundamental problem, and there has in recent years been a significant increase in the number of reported methods approaching this problem in the biological literature.[1] However, it is also a challenging problem, and existing research typically considers restricted model spaces, and algorithms with very high computational requirements.

We present ODEion, a computational software module for structural identification of ODEs in a more general framework, see Fig. 1A. The aim is to identify a dynamic model as a system of ODEs, from sparse and noisy time-series data.

A. ODE SYSTEM IDENTIFICATION



B. TIME SERIES DATA



C. THE SOLUTION MODEL

$$X_3'(t) = \frac{1.0X_4(t)}{X_4(t)+5.0} - \frac{5.0X_3(t)}{(X_3(t)+5.0)(1+X_1(t)/1.0)} + \frac{1.0X_5(t)}{X_5(t)+5.0} - \frac{5.0X_3(t)}{(X_3(t)+5.0)(1+X_2(t)/1.0)}$$

$$\vdots$$

$$\vdots$$

$$X_7'(t) = -\frac{5.0X_7(t)}{(X_7(t)+5.0)(1+\frac{X_3(t)}{1.0})} + \frac{1.0X_6(t)}{X_6(t)+5.0}$$

$$X_3(0) = 1.25$$

$$\vdots$$

$$X_7(0) = 1.36$$

Figure 1: Identification of ODE models. ***A****. Identification can be seen as the opposite to simulation - time-series data is input and a solution model is output. To enable identification, some additional information about the model space and the error function is required, see Section 2 for details. For evaluation purposes, simulated data from a known source model can be used, allowing the solution model to be compared to the source model.* ***B****. Example of time-series data for one experiment.* ***C****. The output of a successful identification algorithm is an ODE model that minimizes the error function.*

For each variable X_i, the generic form of the ODE reads

$$X_i'(t) = \sum_{j=1}^{N_i} f_{ij}(X, \theta, t) \tag{1}$$

where X is the state vector of the system, θ is a vector of parameters, t is time, f_{ij} are arbitrary functions that can be non-linear in both variables and parameters, and N_i represents the number of terms on the right-hand side of equation i. The problem we consider differs from a standard regression problem in two fundamental ways. We need to find a best subset of terms in the ODEs, taken from a very large set of possible terms implicitly defined by the user. Then, to make the most of sparse and noisy data, we cannot simply estimate the derivative in the left hand side of Eq. 1 directly from data, but the model fit needs to be evaluated by simulating the ODEs.

The main contribution of ODEion is to make the highly efficient identification algorithms previously described and extensively evaluated[1, 2, 3, 4], available in the form of a software module. This integrates our previous work, and enables anyone to use and evaluate the developed methods. This is useful for direct applications, and for comparisons with other approaches, which is normally not possible when algorithms are just informally described. Compared to our original research implementations, which could only be used by ourselves in a specific test setting, we have added the significant new ability to allow arbitrary user-defined functions as input. We have invested considerable additional effort for stability in this more general case, and in making the ODEion software easy to use.

Related software include well known statistical software such as R, SAS and SPSS, which typically have tools for non-linear and stepwise regression. We also mention the Eureqa system[5, 6, 7] and Inferelator[8] for finding mathematical relationships in data. Potentially, such systems could be used for similar tasks, especially if the derivatives can be sufficiently well estimated from data. This is however not the typical situation in systems biology where data is sparse and noisy, and these tools have not been designed to provide the functionality or performance of our software module. Furthermore, the software ModelMaGe presents a semi-automatic top-down approach where the solution model is searched for by removing interactions from a pre-defined super-model containing all plausible interactions.[9] This approach can be very helpful, but is fundamentally different from our fully automatic bottom-up approach. Finally, some experimental software codes can be requested from researchers in the field, mainly applicable to specialized models such as S-systems and GMA systems, see Gennemark and Wedelin (2009)[1] for detailed references. A framework for dynamic flux estimation was used for identification of a metabolic system with chemical rate reactions[10], but did not investigate how to best handle noisy data by avoiding direct estimation of the derivatives.

To our knowledge, ODEion is unique in the sense that no other software exists for identification of ODEs when the model space is general (not restricted to e.g. S-systems) and when time-series data is sparse and noisy (excluding methods that infer parameters from estimated derivatives of time series data). Compared to existing algorithms for S-systems our approach is significantly more efficient - no supercomputing is required (see Section 3 for a benchmark summary).

2 Software input and output

To fully specify the identification problem we require the following inputs:[1]

- **Time-series data**. This can be for one or several experiments. In the case of several experiments, they may differ with respect to initial values and/or input functions

(Fig. 1B).

- **Model space and constraints**. The model space, $\mathcal{M}$, defines the allowed forms of the function(s) on the right-hand side of the ODEs, $\forall_{i,j} : \ f_{i,j} \in \mathcal{M}$. These functions are defined by the user and $\mathcal{M}$ may differ between the state variables. For example, for one of the state variables, $\mathcal{M}$ can de defined as

$$\mathcal{M} = \left\{ \begin{array}{l} \overbrace{\theta_{11} X_k(t)}^{\mathcal{M}_1}, \ \overbrace{\theta_{21} X_k(t) X_l(t)}^{\mathcal{M}_2}, \ \overbrace{\frac{\theta_{31} X_k(t)}{\theta_{32} + X_k(t)}}^{\mathcal{M}_3}, \ \overbrace{\frac{\theta_{41} X_k(t)}{(\theta_{42} + X_k(t)) \left(1 + \frac{X_l(t)}{\theta_{43}}\right)}}^{\mathcal{M}_4} \end{array} \right\}, \quad (2)$$

$$\text{where } k, l \in [1, 2 \ldots n], k \neq l, \quad (3)$$

$$\text{where } \theta_{.1} \in [0, 50], \quad \theta_{.2}, \theta_{.3} \in [0.1, 50], \quad (4)$$

  and where n is the number of variables in the system. Here, the upper expression specifies the allowed forms of the functions: $\mathcal{M}_1$ represents a unimolecular linear reaction, $\mathcal{M}_2$ a bimolecular linear reaction, $\mathcal{M}_3$ a Michaelis-Menten reaction, and $\mathcal{M}_4$ represents a Michaelis-Menten reaction with non-competitive inhibition. The middle expression defines which variables are allowed in the functions, and the lower expression defines the parameter ranges.

  Finally, lower and upper bounds for the initial data-point in each time-series can also be defined.

- **Initial model**. The initial model corresponds to prior knowledge of the structure of the system and is included as reactions (terms) from the model space on the right hand side of the ODEs. Also terms from outside the model space can be included in the initial model. If no prior structural information is available, all ODEs are initially set to 0.

- **Error function**. The error function is defined as

$$-L(\hat{X} | \, \theta) + h\left(\lambda, |\theta|, |\hat{X}|\right). \quad (5)$$

  The first term is the negative log-likelihood of the experimental data ($\hat{X}$ denotes experimental data, θ is the vector of parameters), and the second term penalizes structural complexity of the model (h is an arbitrary function, λ is a local parameter in the function h, $|\theta|$ is the number of model parameters, and $|\hat{X}|$ is the number of data points). This makes it possible to use AIC and BIC/MDL.

It is important to note that by specifying the model space and the error function as part of the problem, the problem is unambiguously defined as an optimization problem, giving a clear conceptual separation between modelling and solving identification problems.

The output is a solution model, i.e. a specific ODE of the form Eq. 1 for each variable (Fig. 1C).

3 Overview of the software

The software is available at www.odeidentification.org, together with detailed documentation and sample scripts for a variety of tasks. It is distributed under the GNU general public license. The Java Runtime Environment and a Fortran compiler are required. All source code is provided.

The software is implemented in a form common for specialized computational software, and is easy to use as a standalone program, with input and output in human-readable text files[1, 3], or in other formats such as SBML. It can be integrated in a workflow or other system as desired.

The software offers four main functionalities:

- Verify syntax and consistency of a problem. Potential errors and reasonable solutions to these are reported.
- Visualize data from an experiment of a problem.
- Generate a Fortran program designed to solve a specific problem. The resulting program is self-contained, and can be compiled and run immediately, or saved for future execution.
- Generate a Systems Biology Markup Language (SBML) file of the solution model, allowing visualization, simulation and further analysis on a range of software.[11] Output is also given in a simple human readable format.

Input data manipulation is implemented in Java for flexibility, and time-critical code is generated in Fortran to allow for best performance and access to efficient numerical libraries.

The Fortran program implements the actual identification algorithm, which consists mainly of a search algorithm for the structure, and a parameter estimation algorithm optimized for this context, see Gennemark and Wedelin (2007)[2] for full detail. A heuristic algorithm performs a bottom-up search of the model space for a model that minimizes the error function. It is based on a local search that incrementally adds terms (one f_{ij} at a time) to individual ODEs from the model space, locally (and occasionally globally) estimating the parameters for the different models considered. The search algorithm also removes terms that no longer appear significant. For example, the solution model in Fig. 1C was incrementally built from an initial empty model structure (each ODE was equal to zero), to the resulting model with up to four terms on the right hand sides of the equations.

To provide a complete candidate model that can be evaluated with the error function, the search algorithm in turn uses an accurate and efficient parameter estimation algorithm. For sparse and noisy data it is not adequate to base parameter estimation on straightforward estimation of derivatives directly from data. For significantly better accuracy, models are evaluated by simulating their output which is compared with the given time-series data.

In addition, the parameter estimation is very time-critical, since parameters need to be estimated for a very large number of candidate models, so it is a significant design challenge to provide both the desired accuracy and speed. The main approach for speed is to match the bottom-up search and compute locally as much as possible, resorting to full global

computation only when necessary. This local computation is an important factor in the competitiveness of the implemented methods. It follows that the total number of variables is not the most critical factor with respect to how the implemented methods scale with problem size. Rather, it is the structure and nature of the equations themselves, and the number of variables in each subexpression that are the major factors with respect to execution time. On a lower level, our parameter estimation uses well known methods and implementations such as `dn2gb` (see http://www.netlib.org).

The performance of the *algorithms* have previously been evaluated and found competitive on a large set of benchmark problems, both with models based on chemical rate reactions, S-systems and GMA systems.[1, 3, 4] The new ODEion implementation has been extensively tested on the same benchmark set. Fig. 2 illustrates the current performance of ODEion for problems where results of other methods have been reported (see Table 1 for further details about these problems). It is to be noted that ODEion often is several orders of magnitude faster.

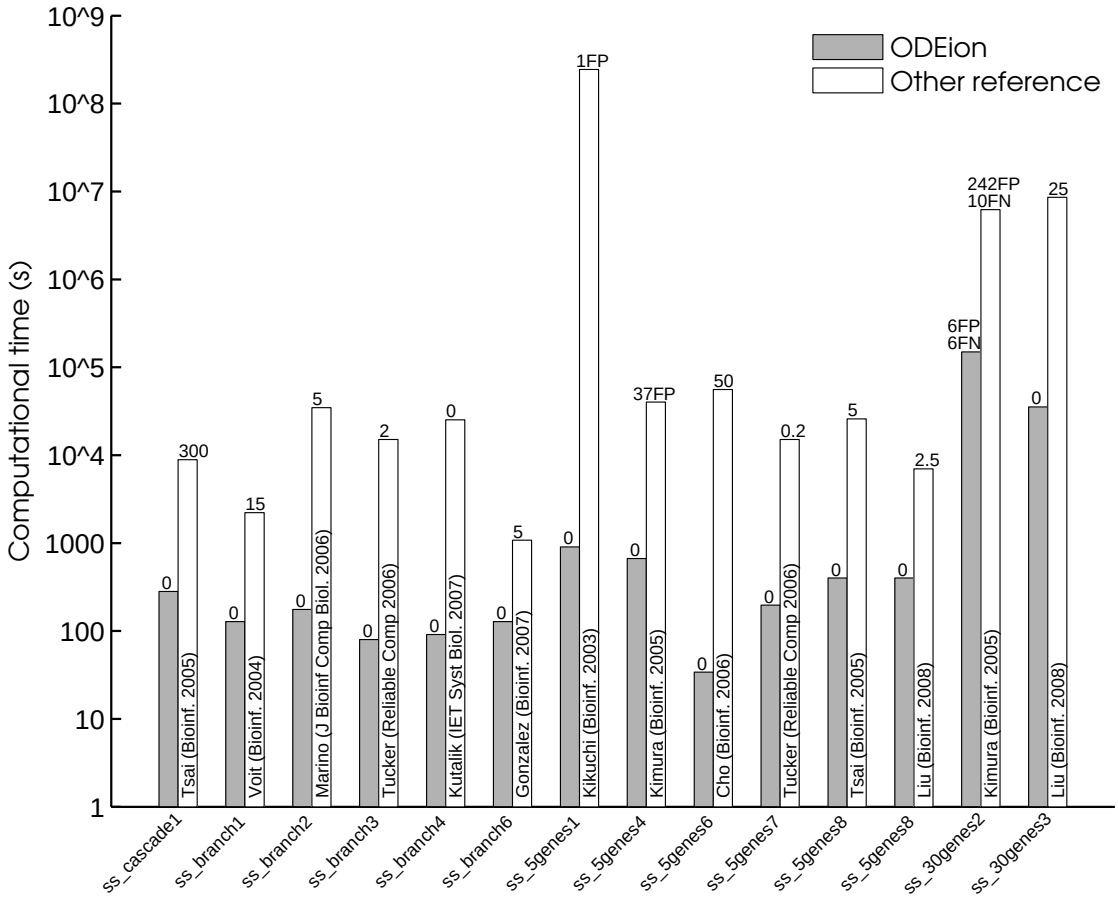


Figure 2: Comparison between ODEion and other methods on problems where results of other methods have been reported (specification of all problems and complete references are given on www.odeidentification.org). The bars correspond to the computational time (logarithmic scale) in seconds, scaled to a single 1GHz processor. If the correct structure was found, parameter accuracy is given on top of each bar as the relative error (%) of the parameter with greatest relative error. If the true structure was not found, FP and FN indicate the number of false positive and negative interactions, respectively.

Table 1: Main characteristics of the problems considered in Fig. 2. Here, #Var refers to the number of variables in the system, #React refers to the number of reactions (interactions) between variables in the source model, and #Param refers to the number of parameters in the source model.

Problem(s)	#Var	#React	#Param	Description
ss_cascade1	3	5	14	Source model with two negative feedbacks acting on the same variable.
ss_branch1–4,6 (5 problems)	4	6	18	Source model with one negative feedback, and one very weak interaction. Problems differ with respect to amount of data, sampling times, and prior structural information.
ss_5genes1,4,6,7 (4 problems)	5	8	23	Source model with four negative feedbacks (two acting on the same variable). Sparse sampling in the time region of interest. Problems differ with respect to amount of data, sampling times, and prior structural information.
ss_5genes8	8	13	28	Larger model space compared to ss_5genes1,4,6,7.
ss_30genes2	30	38	128	Large source model and model space. Several weak interactions combined with noisy data.
ss_30genes3	30	38	128	Large source model and model space.

4 Examples

In order to illustrate the use of ODEion, and explain how ODEion can be used to explore identifiability issues in practice, we consider three examples. The purpose of the examples is to show the kind of information one can easily obtain from ODEion, and these are not intended as independent research in their own right.

4.1 A metabolic network

The metabolic pathway depicted in Fig. 3 has previously been studied by Arkin and Ross (1994, 1995)[12, 13] and by Gennemark and Wedelin (2007).[2] The system has two input variables X_1 and X_2 and five dependent variables X_3–X_7. The variables X_3–X_5 correspond to fructose interconversion (fructose 6-phosphate with fructose 1,6-bisphosphate and fructose 2,6-bisphosphate, respectively), and the variables X_6–X_7 represent a phosphorylation/dephosphorylation cycle commonly found in many systems. The input variables correspond to regulators of phosphofructokinase 1 and 2, e.g. phosphoenolpyruvic acid, citrate, and glucagon. The reactions v_1–v_6 are modelled by Michaelis-Menten kinetics with non-competitive inhibition and are catalyzed by different enzymes which are assumed present at constant levels.[13] Fig. 1C writes out complete ODEs of X_3 and X_7.

Based on previously reported problems on this system, metabol1-3[1], we created a set of new problems with varying amount and quality of time-series data but with the same model space. To generate time-series data is a straightforward simulation task and example scripts that simulate and write data in the ODEion file format are included in the ODEion download.

In order to generate a problem for ODEion we also need a specification of the model space

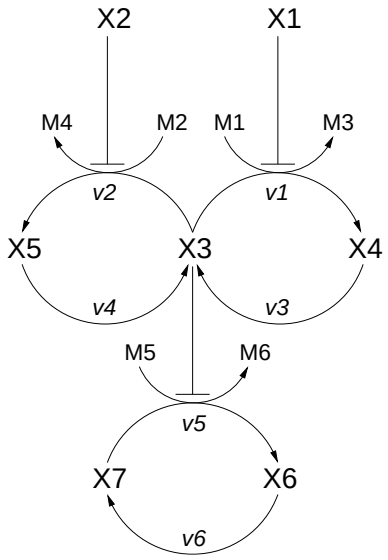


Figure 3: The metabolic system representing a biochemical NAND gate.

in terms of possible reactions for each variable, and some other details in accordance with Section 2. This is written in a self-documenting format which is easy to read both by the human and the computer. An extract of the format is given below:

```
// VARIABLES
variable_1 has name = x1 is inputVariable
variable_2 has name = x2 is inputVariable
variable_3 has name = x3 is dependent
...

// REACTION TYPES
reaction_1
has name = biMolecularMassAction
has localVariableName_1 = X1
has localVariableName_2 = X2
has localParameterName_1 = k1
has equation = k1*X1*X2

reaction_2
has name = michaelisMenten
has localVariableName_1 = X1
has localParameterName_1 = k1
has localParameterName_2 = k2
has equation = k1*X1/(k2+X1)
...

// MODEL SPACE OF VARIABLE 3
possibleReaction_3 of variable_3
has type = biMolecularMassAction
```

```
has spaceOfVariable X1 = memberOfSet_2
has spaceOfVariable X2 = memberOfSet_1
has rangeOfParameter k1 = range_1
...

// EXPERIMENT 1
sample_1 of experiment_1
has time = 0.00
has variable_ = 3.00 2.00 ...
has sdev of variable_ = 0.00 0.00 ...
...
```

This particular extract describes first some variables, and then the definitions of two reaction terms (exemplifying how arbitrary reactions are input to ODEion). This is followed by a possible reaction in the model space. Finally, a sample from the first experiment is given. Complete files for the identification problems (metabol1–3 and modifications thereof) based on this system – and associated scripts – are available in the ODEion download package, using a model space for all variables as in Eq. 2.[1, 2] A detailed user's manual makes it easy to define and execute own problems.

We next ran all problems in ODEion and monitored the number of false positive and false negatives interactions obtained, respectively, see Table 2. Given information about measurement precison, such data is valuable when planning a study: what results can be expected for systems of a particular kind and with various experimental designs? Since a complete dynamic model is obtained by ODEion, more advanced measures than simply reporting false positives/negatives can be considered, e.g. simulation based measures. We note that false interactions are usually weak, so if simulation is the main purpose a model can work well even if the structure is not perfect.

Table 2: Number of false positive and false negative interactions, respectively, obtained by ODEion when identifying test system metabol with various amount and quality of data (7 sample points for each variable and experiments as in Fig. 1B). Noise is added from a Gaussian distribution with standard deviation given as a certain percentage of each experimental value. The best result is taken from several runs with various random seeds.

Number of experiments	Noise level=0 %	Noise level=10 %	Noise level=20 %
12	0/0	0/0	2/1
10	0/0	0/0	2/1
8	0/0	4/2	4/2
6	3/2	5/2	4/2
4	3/2	5/3	12/9

In general, when the true system is unknown and several systems are plausible, the computational analysis can be repeated for a range of hypothetical systems for increased robustness. Naturally, if bias in data is an issue such data can be generated and analyzed in the same way.

4.2 A genetic network

In the previous example, prior structural information was not considered. Here, we will use a genetic network to illustrate identification with different amounts of data and different prior information. The system was introduced by Maki et al. (2001)[14] and applied in Kimura et al. (2005)[15] and Kutalik et al. (2007)[16]. It represents a genetic network with 30 variables, see Fig. 4A for the structure of the network.

This model is defined as a so called S-system model. The S-system formalism is based on approximating kinetic laws with multivariate power-law functions.[17, 18] A model consists of n non-linear ODEs and the generic form of equation i reads

$$X_i'(t) = \alpha_i \prod_{j=1..n} X_j(t)^{g_{ij}} - \beta_i \prod_{j=1..n} X_j(t)^{h_{ij}} \tag{6}$$

where X is a vector (length n) of dependent variables, α and β are vectors (length n) of non-negative rate constants and g and h are matrices ($n \times n$) of kinetic orders, that can be negative as well as positive. We note that Eq. 6 fits the generic model space of ODEion given in Eq. 1.

We will start by considering the problem ss_30genes2[1], which includes 20 experiments with varying initial conditions and with noise from a Gaussian distribution with standard deviaton given as 20 % of each experimental value. This problem contains no prior information about the structure, all ODEs are assumed equal to zero.

Assuming that some interactions are known from literature, we can define a prior structure as depicted in Fig. 4B, and create a new problem using syntax like this (see the ODEion user manual for details):

```
// MODEL SPACE OF G
 g has defaultLowerBound =  -3.
 g has defaultUpperBound =   3.
 :
 g_5_1 has lowerBound = 0.0001
 g_6_1 has lowerBound = 0.0001
 :
```

This means that we force some interactions to be present in the model. Rather than fixing parameter values for these interactions, we choose to only specify wide bounds for the parameters, so all parameters in the model are still estimated by the system. The prior information is therefore in this case qualitative and not quantitative.

We also created problems with fewer experiments and ran all problems in ODEion, see Table 3 for summary data. We first note that it is not unexpected to obtain false positives/negatives for problems with noisy data, and that the number of falsely identified interactions increases as the number of experiments gets smaller. We can also see, in Table 3, how prior structural information significantly improves inference of the topology.

The results contribute to better understanding of the expected magnitude of the gain as well as the dependence of the gain on various amounts of data. Again, this is valuable when designing experiments in a learning-confirming cycle. This kind of analysis with repeated

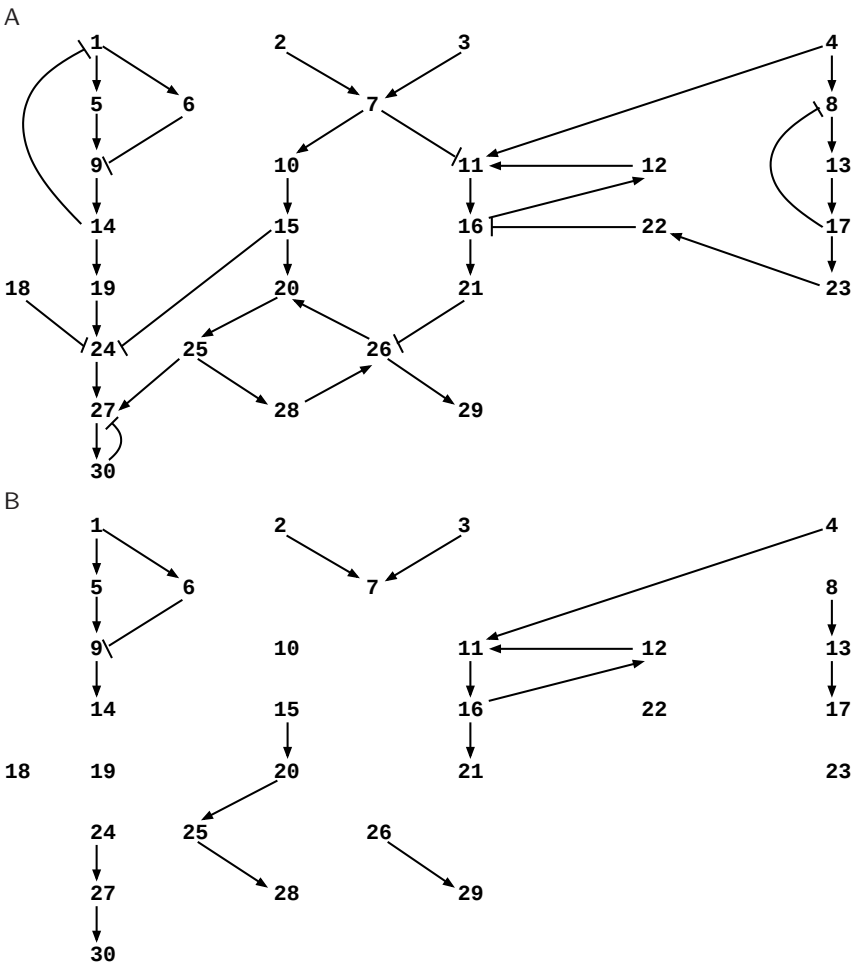


Figure 4: ***A****. The true structure of the genetic network, ss_30genes.* ***B****. The structural prior information in cases where it was assumed. In both figures, positive regulations are indicated by arrows, whereas negative regulations are indicated by blunt arrows.*

Table 3: Number of false positive and false negative interactions, respectively, obtained by ODEion when identifying test system ss30genes with various prior information and amount of data. The best result is taken from several runs with various random seeds.

Number of experiments	No structural prior information	Structural prior information (Fig. 4B)
20	6/6	5/3
15	10/14	8/10
10	13/22	8/16

identification of several relatively large problems (tens of variables) was previously not feasible without supercomputing (compare execution times in Fig. 2). Using ODEion the complete analysis can be run on a single computer in a couple of days. This is not possible with any other software that we are aware of.

4.3 Inferring a model for ethanol fermentation

In a final example, we consider a problem based on real data from a batch fermentation process presented in Wang et al (2001) and Liu and Wang (2008).[19, 20] The goal is to infer an S-system model using two experiments with varying start concentrations of glucose, see Fig. 5A and B. ODEion infers the following model:

$$\begin{aligned} B'(t) &= 2.17 - 1.25 \times B(t)^{-0.422}, \\ G'(t) &= 0.0584 \times B(t)^{2.40} - 2.40 \times B(t)^{1.12} \times G(t)^{0.0306}, \\ E'(t) &= 4.98 \times B(t)^{0.292} - 3.21, \\ L'(t) &= 0.440 \times L(t)^{0.559} - 0.000349 \times B(t)^{2.80}, \\ B(0) &= \{0.287, 0.270\}, \\ G(0) &= \{98.4, 151\}, \\ E(0) &= \{0.106 \times 10^{-2}, 0.164 \times 10^{-5}\}, \\ L(0) &= \{0.990 \times 10^{-3}, 0.512 \times 10^{-8}\}, \end{aligned} \tag{7}$$

where B, G, E and L refer to biomass, glucose, ethanol, and glycerol, respectively, and where initial conditions are given for experiments 1 and 2, respectively. Simulated trajectories of the model fit data reasonably well as depicted in Fig. 5A and B. As Wang et al., we validate the inferred model by predicting a third experiment not part of the training set, see Fig. 5C for data and predicted trajectories. Clearly, the model predicts an unrealistic glucose trajectory, which increases after 14 hours.

The S-system inferred by Liu and Wang has a different structure than the model inferred by ODEion. However, the two models behave similarly when visually comparing the simulated trajectories on experiment 1 and 2. A quantitative comparison was not feasible since initial conditions are not specified in the Liu and Wang paper (an attempt to take initial conditions from data was not successful).

The inferred S-system structure is partly hard to interpret mechanistically, and gives unrealistic predictions. To further analyze this inference problem in a more mechanistic way, we use a model space, $\mathcal{M}$, composed of reactions of Monod type [21, 22] as

$$\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4\},$$

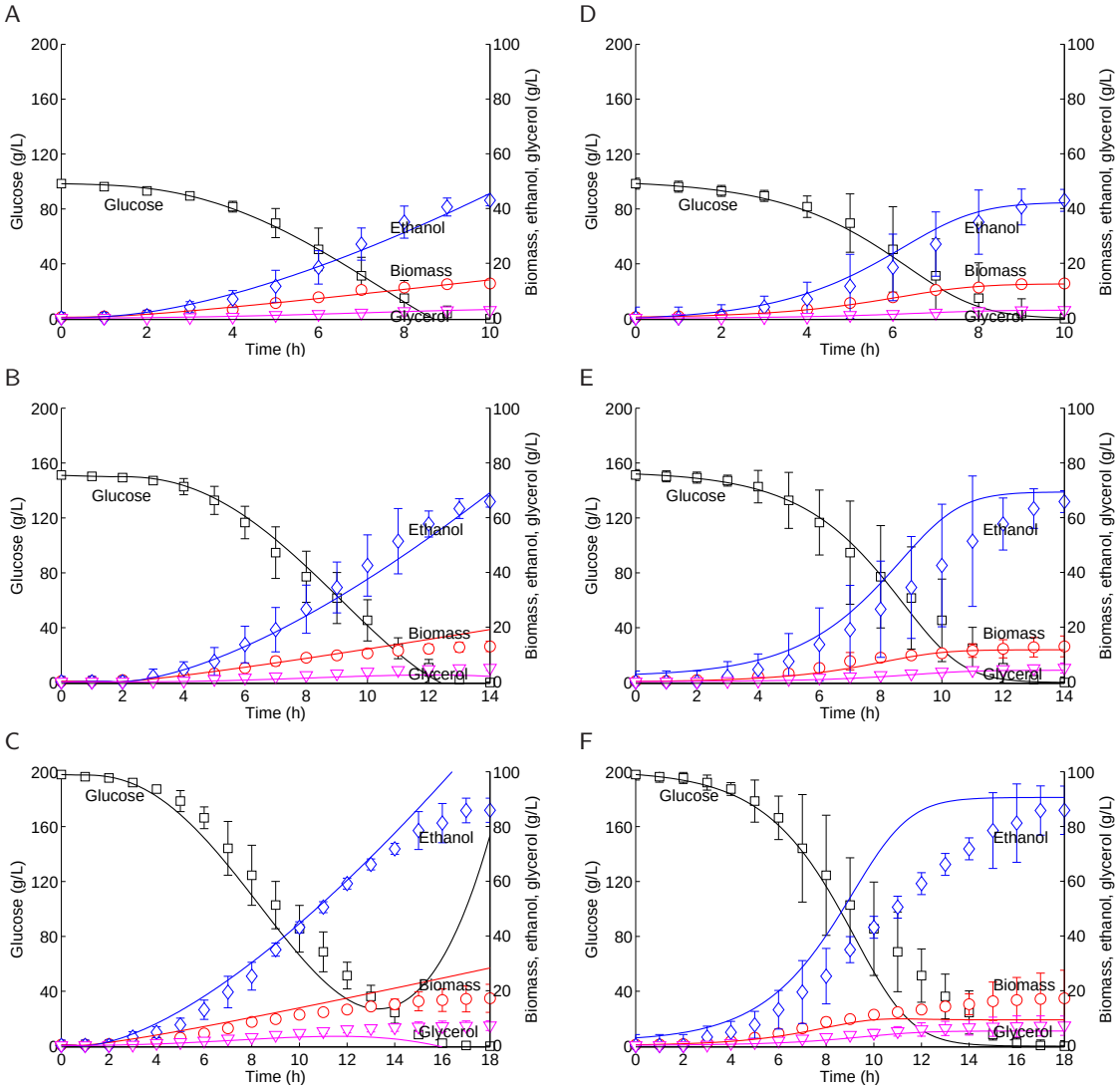


Figure 5: Data from a batch fermentation process and simulated trajectories of mathematical models representing this system. A and B: data for experiments 1 and 2, respectively, and simulated trajectories of an S-system model inferred from these two experiments (problem ss_ethanolferm3). C: data for experiment 3, used as validation data, and prediction of this third experiment using the S-system model. D and E: data for experiments 1 and 2, respectively, and simulated trajectories of a reaction kinetics model inferred from these two experiment (problem ethanolferm3). C: data for experiment 3, used as validation data, and prediction of this third experiment using the reaction kinetics model.

$$
\begin{aligned}
\mathcal{M}_1 &= \mu \frac{G(t)}{K_S + G(t)} B(t), \\
\mathcal{M}_2 &= \mu \frac{G(t)}{K_S + G(t)} \left(1 - \frac{E(t) - P_i}{P_m - P_i}\right) B(t), \\
\mathcal{M}_3 &= \mu \frac{G(t)}{K_S + G(t)} \frac{K_i}{K_i - G(t)} B(t), \\
\mathcal{M}_4 &= \mu \frac{G(t)}{K_S + G(t)} \left(1 - \frac{E(t) - P_i}{P_m - P_i}\right) \frac{K_i}{K_i - G(t)} B(t), \\
\mu &\in [-100, 100], \quad K_S, K_i \in [0.1, 200], \quad P_i \in [0.1, 100], \quad P_m \in [80, 100].
\end{aligned} \tag{8}
$$

Here, glucose is considered the growth limiting substrate. $\mathcal{M}_1$ introduces μ which corresponds to the maximum overall specific rate (of growth for biomass; of substrate utilization for glucose, and of production for ethanol and glycerol). Furthermore, K_S is a substrate limitation constant. In $\mathcal{M}_2$, an etahnol inhibition factor is added, where P_m is maximum ethanol concentration, and P_i is the ethanol inhibition constant. Alternatively, in $\mathcal{M}_3$, a substrate inhibition factor is added, where K_i the substrate inhibition constant. $\mathcal{M}_4$ combines substrate inhibition and ethanol inhibition. In constrast to the S-system model space, the parameters are interpretable, and we can assign parameter bounds from data (e.g. K_S should be in the observed data range for glucose).

The model space $\mathcal{M}$ is applied to all variables. It is straightforward to replace the S-system model space with $\mathcal{M}$ and run the new problem in ODEion. The model space is coded in the input file in analogy to the metabolic network previously described (Section 4.1). The inferred model was:

$$
\begin{aligned}
B'(t) &= 2.04 \frac{G(t)}{84.8 + G(t)} \left(1 - \frac{E(t) - 8.60}{80.0 - 8.60}\right) \frac{81.5}{81.5 + G(t)} B(t), \\
G'(t) &= -6.38 \frac{G(t)}{53.6 + G(t)} B(t), \\
E'(t) &= 3.11 \frac{G(t)}{66.5 + G(t)} B(t), \\
L'(t) &= 0.122 \frac{G(t)}{22.8 + G(t)} B(t), \\
B(0) &= \{0.356, 0.234\}, \\
G(0) &= \{98.5, 153\}, \\
E(0) &= \{0.654 \times 10^{-4}, 0.00\}, \\
L(0) &= \{0.747 \times 10^{-9}, 0.161\}.
\end{aligned} \tag{9}
$$

See Fig. 5D–E for the fitted curves. The inferred model is then used to predict a third experiment not part of the training set, see Fig. 5F. The mechanistic approach does not extrapolate perfectly in this case. However, in the mechanistic approach, the predicted concentration time profile of glucose approaches zero at the end of the experiment, while the corresponding S-system trajectory makes a bend upwards after about 14 hours as previously mentioned. Therefore, the prediction of the mechanistic model can be seen as more sound than the S-system prediction. The result may also be helpful in reconsidering the model space, in an iterative fashion. In general, ODEion allows for flexible exploration of both S-systems and chemical rate reaction models.

5 Discussion

Mathematical models of phenomena in nature are commonly defined as systems of ordinary differential equations. Such models can be viewed to have a structure specifying the form of the equations – typically chosen manually – and parameters which can be automatically estimated from data.

ODEion gives access to very efficient identification algorithms, which go beyond parameter estimation and allow algorithmic modelling support also with structural uncertainty, enabling the user to represent uncertainty in both structure and parameters. ODEion considers many different structures and attempts to find the best model given the data and an error criterion (model selection). ODEion is general in the sense that the user can define a wide range of functions in the model space. The software is therefore not restricted to any particular model type or application.

Concerning limitations, these exist both with respect to the nature of the problem itself, and with respect to the used methods and their implementation. Output from a software of this kind must therefore be interpreted with care, see Gennemark and Wedelin (2007)[2] for a longer discussion. A fundamental difficulty in the identification of biological systems is that data is sparse, highlighting the issue that there is no guarantee that available data is sufficient in a given case, or that a system is uniquely identifiable with a particular type of data. From a practical perspective, this can be handled by providing more data, and/or alternative data from a modified system, or by breaking it up in parts. As illustrated in the examples, ODEion itself can be used to explore and give hints about what data is required for successfully identifying a particular kind of system, and generally to investigate ways to successfully formulate and solve problems in a particular area. If required, it is possible to further analyse solution models by other methods e.g. by the observability test and software suggested in Sedoglavic (2002).[23]

Then, due to the combinatorial complexity and the heuristic nature of the search, the software does not guarantee that the best solution to a given problem is found, and it is difficult to give exact bounds on execution time. However, thanks to the main algorithmic strategies of simulation and local computation, ODEion has been shown to successfully identify many relatively large and realistic problems, using biologically realistic amounts of data, and with reasonable computational effort.[1]

The ODEion software is easy to use as it is, as we have shown in the examples. The examples illustrate how ODEion can be used to solve and analyze identification problems, providing information that would be very hard to obtain without ODEion. We assess this to be useful not only in research, but also for educational purposes. Additionally, since source code is provided, the software can be freely adapted and integrated.

References

[1] Gennemark P, Wedelin D, Benchmarks for identification of ordinary differential equations from time series data, *Bioinformatics* **25**(6):780-6, 2009.

[2] Gennemark P, Wedelin D, Efficient algorithms for ordinary differential equation model identification of biological systems, *IET Syst Biol.* **1**:120-9, 2007.

[3] Wedelin D, Gennemark P, Web site with benchmarks for ODE identification problems, http://www.odeidentification.org, 2008.

[4] Kattas GD, Gennemark P, Wedelin D, Structural identification of GMA models: algorithm and model comparison, *Proceedings of the 8th International Conference on Computational Methods in Systems Biology*, ACM New York, NY, USA, ISBN: 978-1-4503-0068-1, 2010.

[5] Bongard J, Lipson H, Automated reverse engineering of nonlinear dynamical systems, *Proc Natl Acad Sci USA* **104**(24):9943-8, 2007.

[6] Schmidt M, Lipson H, Distilling free-form natural laws from experimental data, *Science* **324**(5923):81-85, 2009.

[7] Schmidt MD, Vallabhajosyula RR, Jenkins JW, Hood JE, Soni AS, Wikswo JP, Lipson H, Automated refinement and inference of analytical models for metabolic networks, *Phys Biol* **8**(5):055011, 2011.

[8] Bonneau R, Reiss DJ, Shannon P, Hood L, Baliga NS, Thorsson V, The Inferelator: a procedure for learning parsimonious regulatory networks from systems-biology datasets de novo, *Genome Biol* **7**(5):R36, 2006.

[9] Schaber J, Flöttmann M, Li J, Tiger CF, Hohmann S, Klipp E, Automated ensemble modeling with modelMaGe: analyzing feedback mechanisms in the Sho1 branch of the HOG pathway, *PLoS One* **6**(3):e14791, 2011.

[10] Chou IC, Voit EO, Estimation of dynamic flux profiles from metabolic time series data, *BMC Syst Biol* **6**:84, 2012.

[11] Hucka M, Finney A, Sauro HM, et al., The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models *Bioinformatics* **19**(4):524-531, 2003.

[12] Arkin AP, Ross J, Computational functions in biochemical reaction networks, *Biophys J* **67**:560-578, 1994.

[13] Arkin AP, Ross J, Statistical construction of chemical reaction mechanisms from measured time-series, *J Phys Chem* **99**:970-979, 1995.

[14] Maki Y, Tominaga D, Okamoto M, Watanabe S, Eguchi Y, Development of a system for the inference of large scale genetic networks, *Pac Symp Biocomput* **446-58**, 2001.

[15] Kimura S, Ide K, Kashihara A, Kano M, Hatakeyama M, Masui R, Nakagawa N, Yokoyama S, Kuramitsu S, Konagaya A, Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm, *Bioinformatics* **21**:1154-63, 2005.

[16] Kutalik Z, Tucker W, Moulton V, S-system parameter estimation for noisy metabolic profiles using newton-flow analysis, *IET Syst Biol* **1**:174-80, 2007.

[17] Savageau MA, *Biochemical systems analysis: a study of function and design in molecular biology*, Addison-Wesley, Reading, Mass, 1976.

[18] Voit EO, *Computational analysis of biochemical systems. A practical guide for biochemists and molecular biologists*, Cambridge University Press, Cambridge, 2000.

[19] Wang FS, Su TL, Jang HJ, Hybrid differential evolution for problems of kinetic parameter estimation and dynamic optimization of an ethanol fermentation process, *Ind Eng Chem Res* **40**:2876-85, 2001.

[20] Liu PK, Wang FS, Inference of biochemical network models in S-system using multiobjective optimization approach, *Bioinformatics* **24**:1085-92, 2008.

[21] Monod, J. *Recherches sur la Croissance des Cultures Bactriennes*, Hermann, Paris, 1941.

[22] Leksawasdi N, Joachimsthal EL, Rogers PL, Mathematical modelling of ethanol production from glucose/xylose mixtures by recombinant Zymomonas mobilis, *Biotech Letters* **23**:1087-93, 2001.

[23] Sedoglavic A, A probabilistic algorithm to test local algebraic observability in polynomial time, *J Symb Comp* **33**:735-755, 2002.