Available online at www.sciencedirect.com
ScienceDirect
Mathematical Biosciences 208 (2007) 607–620
Mathematical Biosciences
www.elsevier.com/locate/mbs

Estimating parameters for generalized mass action models using constraint propagation

Warwick Tucker [a], Zoltán Kutalik [b], Vincent Moulton [c,*]

[a] Department of Mathematics, Uppsala University, Box 480, Uppsala, Sweden
[b] Department of Medical Genetics, University of Lausanne, CH-1005 Lausanne, Switzerland
[c] School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, UK

Received 5 April 2006; received in revised form 13 November 2006; accepted 28 November 2006
Available online 8 December 2006

Abstract

As modern molecular biology moves towards the analysis of biological systems as opposed to their individual components, the need for appropriate mathematical and computational techniques for understanding the dynamics and structure of such systems is becoming more pressing. For example, the modeling of biochemical systems using ordinary differential equations (ODEs) based on high-throughput, time-dense profiles is becoming more common-place, which is necessitating the development of improved techniques to estimate model parameters from such data. Due to the high dimensionality of this estimation problem, straight-forward optimization strategies rarely produce correct parameter values, and hence current methods tend to utilize genetic/evolutionary algorithms to perform non-linear parameter fitting. Here, we describe a completely deterministic approach, which is based on interval analysis. This allows us to examine entire sets of parameters, and thus to exhaust the global search within a finite number of steps. In particular, we show how our method may be applied to a generic class of ODEs used for modeling biochemical systems called Generalized Mass Action Models (GMAs). In addition, we show that for GMAs our method is amenable to the technique in interval arithmetic called constraint propagation, which allows great improvement of its efficiency. To illustrate the applicability of our method we apply it to some networks of biochemical reactions appearing in the literature, showing in particular that, in addition to estimating

* Corresponding author.
E-mail addresses: warwick@math.uu.se (W. Tucker), zoltan.kutalik@unil.ch (Z. Kutalik), vincent.moulton@cmp.uea.ac.uk (V. Moulton).

0025-5564/$ - see front matter
doi:10.1016/j.mbs.2006.11.009

system parameters in the absence of noise, our method may also be used to recover the topology of these networks.

Keywords: S-systems; GMA systems; Metabolic modelling; Parameter estimation; Biochemical systems; Interval analysis; Constraint propagation

1. Introduction

The modeling and simulation of biochemical systems is currently receiving a great deal of attention. This is in part due to the fact that techniques such as mass spectrometry and nuclear magnetic resonance allow the gathering of time-dense profiles consisting of simultaneous measurements of metabolites and proteins within the same cell, cell system or organism [25]. The dynamics of these systems are commonly modeled using systems of ordinary differential equations that involve several parameters corresponding to, e.g., kinetic rates of underlying chemical reactions [24]. In the past, the lack of experimental data made the estimation of large numbers of parameters from biochemical measurements virtually impossible. The dramatic increase in the availability and quality of high-throughput biological data, however, is starting to make parameter estimation in principle feasible [25], although it still remains a difficult mathematical and computational problem.

In mathematical terms, one form of the parameter estimation problem can be stated as follows: assume that we are given a system of ordinary differential equations $\dot{x} = f(x;p)$, where the right-hand side (the vector field) depends on a (multi-dimensional) parameter p. In biochemical modeling, the entries of the vector x usually correspond to the concentration of some reactants. We then aim to search for a particular parameter or parameters $p^\star$ within a search space $\mathbb{P}$ such that the solutions of the system $\dot{x} = f(x;p^\star)$ match a given data set consisting of time samples of the two vectors x and $\dot{x}$ in some pre-specified manner.

In the setting of biochemical modeling various methods have been proposed for parameter estimation (see e.g., [24, Chapter 5], [10] for overviews) including, more recently, numerical optimization methods [12], genetic algorithms [11], evolutionary optimization [22], and neural networks [25]. These methods share the advantages and drawbacks of all global optimization approaches. In particular, they often require good initial estimates for parameters, and they are prone to getting trapped in local optima.

In [23] we describe a new approach for parameter estimation based on the theory of *interval analysis* that can circumvent these two problems. In particular, given time samples of the vector x and corresponding estimates of $\dot{x}$ (derived using, e.g., methods described in [25]) yielding a data set $\{x(t_i);\dot{x}(t_i)\}_{i=1}^{M}$ of concentrations of reactants and the rate of change of these concentrations at times t_i, we present a method that aims to capture information regarding the solution set

$$\Gamma_f(\mathbb{P}) = \Gamma_f(\mathbb{P};\{x^{(i)};\dot{x}^{(i)}\}_{i=1}^{M}) = \{p \in \mathbb{P} : \dot{x}^{(i)} = f(x^{(i)};p) \quad i = 1,\ldots,M\}$$

for a given search region $\mathbb{P}$ in parameter space. This set consists of all parameters that are *consistent* with the data set.

In essence, our method in [23] relies on the observation that the solution set $\Gamma_f(\mathbb{P})$ is the intersection of $\mathbb{P}$ with the (formal) inverse of the constraints $\dot{x}(t_i) = f(x(t_i); p)$ with respect to the parameter p. Without any prior information regarding the vector field f, the solution set may have a very complicated structure. Nevertheless, extending the vector field f to a set-valued function enables us to determine when $\Gamma_f(\mathbb{P})$ is empty. Thus, given a compact, global search region $\mathbb{P}$ we can form a finite partition $\mathbb{P} = \mathbb{P}_1 \cup \ldots \cup \mathbb{P}_n$, and discard all subsets $\mathbb{P}_i$ for which $\Gamma_f(\mathbb{P}_i) = \emptyset$. The union of the remaining subsets forms a covering of the solution set. By adaptively refining the partition, we obtain an accurate covering in an effective manner.

In this paper, we extend this method in two ways. Although our technique can (in principle) be applied to *any* system of finitely parameterized ordinary differential equations, in [23] we apply it to a special class of such systems called S-systems [18–20,24]. Here we extend our method to the more general class of biochemical models called *Generalized Mass Action models* (GMAs). Moreover, we show that this class of systems is amenable to *constraint propagation* – a technique commonly used in interval analysis. In particular, complementing the exclusion test described in the previous paragraph, we can re-interpret the differential equation $\dot{x} = f(x; p)$ as a set of constraints that any solution must satisfy. These additional constraints allow us to discard portions of the partition elements $\mathbb{P}_i$, resulting in a contraction operator on the global search region $\mathbb{P}$.

The rest of the paper is organized as follows. In Section 2 we give a short description of GMAs and S-systems. In Section 3, we describe the underlying methods that we use, in particular providing a brief overview of interval analysis and constraint propagation. In Section 4 we describe how these methods can be applied to obtain parameter estimates for GMAs, and an algorithm for parameter estimation incorporating these techniques. In Section 5 we illustrate the applicability of our algorithm to noise-free data by applying it to two examples of GMAs presented in [24,25]. In Section 6 we make some concluding remarks.

2. Generalized mass action models and S-systems

In this section we briefly review GMAs and S-systems. A more detailed account of these concepts may be found in, e.g., [21,24].

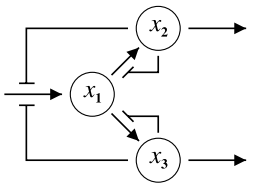
A *generalized mass action model* is a system of ordinary differential equations on the form

$$\dot{x}_i = \sum_{j=1}^{N_i} a_{ij} \prod_{k=1}^{d} x_k^{g_{ijk}} \quad (i = 1, \ldots, d). \tag{1}$$

Each variable x_i (which assumes only positive values) represents the concentration of some reactant, and $\dot{x}_i$ denotes the time derivative of x_i. The parameters a_{ij} are known as *rate constants*, whereas the parameters g_{ijk} are referred to as the *kinetic orders*. The interactions corresponding to the system are usually summarized in the form of a network. For example, a GMA-system and the associated network that is studied in [24, pp. 81–85] is presented in Fig. 1 (See also Section 5.1).

Each component of an GMA-system is made up of positive and negative terms, each corresponding to individual reactions giving rise to the production and consumption of the substance x_i, respectively. To illustrate the significance of a generic term, consider

$$a_{12} x_1^{g_{121}} x_3^{g_{123}} x_4^{g_{124}} = 3.5 x_1^{0.5} x_3^{-0.3} x_4^{2}.$$



$$\dot{x}_1 = 0.4x_2^{-1}x_3^{-1} - 3x_1^{0.5}x_2^{-0.1} - 2x_1^{0.75}x_3^{-0.2}$$
$$\dot{x}_2 = 3x_1^{0.5}x_2^{-0.1} - 1.5x_2^{0.5}$$
$$\dot{x}_3 = 2x_1^{0.75}x_3^{-0.2} - 5x_3^{0.5}$$

Fig. 1. A branched pathway with feedback inhibitions (left), together with the corresponding GMA-system (right).

This is the second term of the first component ($\dot{x}_1$), and describes a reaction contributing to the production of x_1, since the rate constant $a_{12} = 3.5$ is positive. Furthermore, it tells us that this reaction directly involves x_1, x_3, and x_4. Finally, we are informed that whereas x_1 and x_4 *promote* the production of x_1 (their corresponding kinetic orders are positive), x_3 *inhibits* the very same production ($g_{123} = -0.3$ is negative).

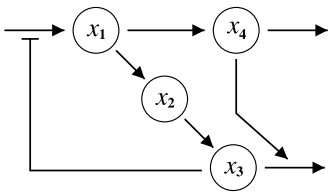
In regards to GMA models, the parameter estimation can be divided into two sub-tasks: First, one may determine the network *topology*. This is a qualitative property of the system that describes whether a substance decreases or enhances the rate of a reaction. The focus of this stage is to determine which parameters are non-zero. Second, one is interested in the *rate* at which the synthesis/degradation occurs. This quantitative information corresponds to finding approximate values of the non-zero parameters.

S-systems are a special class of GMAs having two terms per component; one corresponding to the production, and one corresponding to the degradation of the reactant. An S-system can be considered as a condensed version of a GMA-system, obtained by aggregating individual reactions into the net processes of synthesis and degradation – see [21,24] for a detailed account of this procedure. In particular, such a system consists of a system of ordinary differential equations of the form

$$\dot{x}_i = \alpha_i \prod_{k=1}^{d} x_k^{g_{ik}} - \beta_i \prod_{k=1}^{d} x_k^{h_{ik}} \quad (i = 1, \ldots, d),$$

with non-negative rate constants α_i and β_i and real-valued kinetic orders g_{ij} and h_{ij}. An S-system and its associated network that is described in [25] and that we study later in Section 5.2, is presented in Fig. 2.

One important difference between S-systems and GMA-systems is that, given the dimension d of the system, an S-system has a bounded number $2d(d+1)$ of possible parameters to be



$$\dot{x}_1 = 12x_3^{-0.8} - 10x_1^{0.5}$$
$$\dot{x}_2 = 8x_1^{0.5} - 3x_2^{0.75}$$
$$\dot{x}_3 = 3x_2^{0.75} - 5x_3^{0.5}x_4^{0.2}$$
$$\dot{x}_4 = 2x_1^{0.5} - 6x_4^{0.8}.$$

Fig. 2. A branched pathway with activations and inhibitions (left), together with the corresponding S-system (right).

reconstructed. This is not so in the case of GMA-systems. Therefore, for large systems, one must be given (more or less) the topology of an GMA-system in advance.

3. Methods

3.1. Interval analysis

Here, we will briefly describe the fundamentals of interval analysis. For a concise reference on this topic, see, e.g., [1,13,14].

Let $\mathbb{IR}$ denote the set of closed intervals. For any element $\mathbb{A} \in \mathbb{IR}$, we adopt the notation $\mathbb{A} = [\underline{\mathbb{A}}, \overline{\mathbb{A}}]$. Thus '$a \in \mathbb{A}$' means 'the point a belongs to the interval $\mathbb{A}$'. If $\bigstar$ is one of the operators $+, -, \times, \div$, we define the arithmetic on elements of $\mathbb{IR}$ by

$$\mathbb{A}\bigstar\mathbb{B} = \{a\bigstar b : a \in \mathbb{A}, b \in \mathbb{B}\},$$

except that $\mathbb{A} \div \mathbb{B}$ is undefined if $0 \in \mathbb{B}$. Working exclusively with closed intervals, we can describe the resulting interval in terms of the endpoints of the operands:

$$\begin{aligned}
\mathbb{A} + \mathbb{B} &= [\underline{\mathbb{A}} + \underline{\mathbb{B}}, \overline{\mathbb{A}} + \overline{\mathbb{B}}],\\
\mathbb{A} - \mathbb{B} &= [\underline{\mathbb{A}} - \overline{\mathbb{B}}, \overline{\mathbb{A}} - \underline{\mathbb{B}}],\\
\mathbb{A} \times \mathbb{B} &= [\min(\underline{\mathbb{A}}\underline{\mathbb{B}}, \underline{\mathbb{A}}\overline{\mathbb{B}}, \overline{\mathbb{A}}\underline{\mathbb{B}}, \overline{\mathbb{A}}\overline{\mathbb{B}}), \max(\underline{\mathbb{A}}\underline{\mathbb{B}}, \underline{\mathbb{A}}\overline{\mathbb{B}}, \overline{\mathbb{A}}\underline{\mathbb{B}}, \overline{\mathbb{A}}\overline{\mathbb{B}})],\\
\mathbb{A} \div \mathbb{B} &= \mathbb{A} \times [1/\overline{\mathbb{B}}, 1/\underline{\mathbb{B}}], \quad \text{if } 0 \notin \mathbb{B}.
\end{aligned} \tag{2}$$

When computing with finite precision, directed rounding must also be taken into account (see e.g., [13,14]).

A key feature of interval arithmetic is that it is *inclusion monotonic*, i.e., if $\mathbb{A} \subseteq \mathbb{X}$, and $\mathbb{B} \subseteq \mathbb{Y}$, then

$$\mathbb{A}\bigstar\mathbb{B} \subseteq \mathbb{X}\bigstar\mathbb{Y}, \tag{3}$$

where we require that $0 \notin \mathbb{Y}$ for division.

One of the main reasons for passing to the interval realm is that we want a simple way of enclosing the *range* $R(f;D) = \{f(x) : x \in D\}$ of a real-valued function $f : D \to \mathbb{R}$. Except for the most trivial cases, mathematics provides few tools to describe this set.

We begin by extending the real functions to *interval functions*. By this, we mean functions that take and return intervals rather than real numbers. Interval arithmetic (2) provides the theory of extending rational functions, i.e., functions on the form $f(x) = p(x)/q(x)$, where p and q are polynomials. Simply substituting all occurrences of the real variable x with the interval variable $\mathbb{X}$ (and the real arithmetic operators with their interval counterparts) produces a rational interval function $F(\mathbb{X})$, called the *natural interval extension* of f. As long as no singularities are encountered, we have the inclusion

$$R(f;\mathbb{X}) \subseteq F(\mathbb{X}), \tag{4}$$

by property (3). In fact, this type of range enclosure can be achieved for any reasonable function.

Example 3.1. Let $f(x) = 7/(3 + x)$, and consider the domain $x \in [0, 100]$. By the inclusion property (4), we have the following range enclosure:

$$R(f;[0,100]) \subseteq F([0,100]) = \frac{7}{3+[0,100]} = \frac{7}{[3,103]} = \left[\frac{7}{103}, \frac{7}{3}\right].$$

In this particular case, the enclosure is *sharp*, i.e., we have $R(f;[0,100]) = F([0,100])$.

Higher-dimensional functions $f : \mathbb{R}^n \to \mathbb{R}$ can be extended to an interval function $F : \mathbb{IR}^n \to \mathbb{IR}$ in a similar manner. The function argument is then an *interval-vector* $\mathbb{X} = (\mathbb{X}_1, \ldots, \mathbb{X}_n)$, which we also refer to as a *box*.

Example 3.2. Consider the first component of a two-dimensional S-system:

$$\dot{x}_1 = \alpha_1 x_1^{g_{11}} x_2^{g_{12}} - \beta_1 x_1^{h_{11}} x_2^{h_{12}},$$

together with the parameter domains $\alpha_1 \in [0,4]$, $\beta_1 \in [1,3]$, and $g_{1j}, h_{1j} \in [-1,1]$, $j = 1,2$. By the inclusion property (4), we have the following range enclosure for the vector field, evaluated at the data point $(x_1, x_2) = (1,2)$:

$$\begin{aligned}\dot{x}_1 \subseteq & [0,4]1^{[-1,1]}2^{[-1,1]} - [1,3]1^{[-1,1]}2^{[-1,1]} \\ = & [0,4]\left[\frac{1}{2},2\right] - [1,3]\left[\frac{1}{2},2\right] = [0,8] - \left[\frac{1}{2},6\right] = \left[-6, 7\frac{1}{2}\right],\end{aligned}$$

which coincides with the exact range of slopes.

Interval analysis can be extended to an *exception-free* system by redefining the interval extension process. Without going into details, this makes *all* operations well-defined (including division by zero), see [26]. This extension is heavily used in our examples, where we are repeatedly computing logarithms of sets containing negative numbers.

There exist several open source programming packages for interval analysis [3,7,16], as well as commercial products such as [5].

3.2. Constraint propagation

The basic idea behind constraint propagation is to view a single equation $y = f(x)$ as a set of constraints that must be satisfied by the solution(s) of the original equation. Viewed as set-valued functions, these constraints often act as *contractors*, i.e., propagating the original search domain through the constraints often produces a smaller domain.

Example 3.3. As a first example, consider the equation $y = x_1^3 + x_2$, which can be recast as $x_1 = \sqrt[3]{y - x_2}$ and $x_2 = y - x_1^3$. Given a search domain $\mathbb{X}_1 \times \mathbb{X}_2$, we can impose the constraints $x_1 \in \mathbb{X}_1 \cap \sqrt[3]{y - \mathbb{X}_2}$ and $x_2 \in \mathbb{X}_2 \cap (y - \mathbb{X}_1^3)$ on all solutions. Taking $y = 2$, and looking for a solution with $(x_1, x_2) \in [0,1] \times [0,1]$, the constraints produce

$$\begin{aligned}x_1 \in & \mathbb{X}_1 \cap \sqrt[3]{y - \mathbb{X}_2} = [0,1] \cap \sqrt[3]{2 - [0,1]} = [0,1] \cap \sqrt[3]{[1,2]} = [0,1] \cap [1, \sqrt[3]{2}] = \{1\}, \\ x_2 \in & \mathbb{X}_2 \cap (y - \mathbb{X}_1^3) = [0,1] \cap (2 - [0,1]^3) = [0,1] \cap (2 - [0,1]) = [0,1] \cap [1,2] = \{1\}\end{aligned}$$

which actually happens to give the (unique) solution within the domain.

In general, we are not so fortunate as in this example, but the constraints do often contract the components of the search domain.

Example 3.4. As a second example, consider again the first component of a two-dimensional S-system:

$$\dot{x}_1 = \alpha_1 x_1^{g_{11}} x_2^{g_{12}} - \beta_1 x_1^{h_{11}} x_2^{h_{12}}.$$

This equality can be recast as, e.g., a constraint for the parameter g_{12}:

$$g_{12} = \frac{1}{\log x_2} \log\left(\frac{\dot{x}_1 + \beta_1 x_1^{h_{11}} x_2^{h_{12}}}{\alpha_1 x_1^{g_{11}}}\right).$$

Given the same parameter domains and data point as in Example 3.2, and given the slope estimate $\dot{x}_1 = 5$, this constraint produces

$$\begin{aligned} g_{12} \in & [g_{12}] \cap \frac{1}{\log x_2} \log\left(\frac{\dot{x}_1 + [\beta_1] x_1^{[h_{11}]} x_2^{[h_{12}]}}{[\alpha_1] x_1^{[g_{11}]}}\right) = [-1,1] \cap \frac{1}{\log 2} \log\left(\frac{5 + [1,3] 1^{[-1,1]} 2^{[-1,1]}}{[0,4] 1^{[-1,1]}}\right) \\ = & [-1,1] \cap \frac{1}{\log 2} \log\left(\frac{5 + [1,3][\tfrac{1}{2},2]}{[0,4]}\right) = [-1,1] \cap \frac{1}{\log 2} \log\left(\frac{[5\tfrac{1}{2},11]}{[0,4]}\right) \\ = & [-1,1] \cap \frac{1}{\log 2} \log\left[\frac{11}{8}, +\infty\right] = [-1,1] \cap \left[\frac{\log \frac{11}{8}}{\log 2}, +\infty\right] = \left[\frac{\log 11}{\log 2} - 3, 1\right] \approx [0.4594, 1.0]. \end{aligned}$$

Here the constraint contracted the domain for g_{12} by a factor $\approx$3.7, despite involving a (well-defined) division by zero. Analogous procedures can be carried out for the remaining parameters.

By iterating the contractions, in combination with a partitioning scheme, the solution set is rapidly obtained. Furthermore, if we, somewhere in the process, encounter an empty intersection, then we have proved that no solution exists within the given domain. For a thorough treatment of constraint propagation techniques, see [8] and references therein.

4. Parameter estimation

In this section, we shall show that constraint propagation is well-suited to parameter estimation in GMA-systems. Recall that a component of such a system can be expressed as

$$\dot{x} = \sum_{i=1}^{N} a_i \prod_{j=1}^{d} x_j^{g_{ij}}.$$

As described in [23,25], we assume that the GMA-system in question can be *decoupled* into d independent sets of algebraic equations. This assumes that we are given the values of the variables $x_1, \ldots, x_d$ *and* the slope $\dot{x}$ at various time points, i.e., a data set of the form $\{x(t_i); \dot{x}(t_i)\}_{i=1}^{M}$. An alternative approach, which does not rely on decoupling, is to work directly on the full set

of ODEs. Some progress in this direction has been made in [9,17], but only for very small systems.

Our task is to use the data set to estimate the parameters a_i and g_{ij}, given their associated search domains $\mathbb{A}_i$ and $\mathbb{G}_{ij}$, respectively. In this situation, the set-valued constraints for the rate constants become

$$a_k \in \mathbb{A}_k \cap \left(\left(\dot{x} - \sum_{\substack{i=1 \\ i\neq k}}^{N} \mathbb{A}_i \prod_{j=1}^{d} x_j^{\mathbb{G}_{ij}} \right) \Big/ \prod_{j=1}^{d} x_j^{\mathbb{G}_{kj}} \right), \tag{5}$$

whereas, for the kinetic orders, we have (compare with Example 3.4)

$$g_{k,\ell} \in \mathbb{G}_{k,\ell} \cap \log \left(\left(\dot{x} - \sum_{\substack{i=1 \\ i\neq k}}^{N} \mathbb{A}_i \prod_{j=1}^{d} x_j^{\mathbb{G}_{ij}} \right) \Big/ \left(\mathbb{A}_k \prod_{\substack{j=1 \\ j\neq \ell}}^{d} x_j^{\mathbb{G}_{kj}} \right) \right) \Big/ \log x_\ell. \tag{6}$$

In order to keep the formulas to a minimum, we introduce the following short-hand notations:

$$\mathcal{A}_k(\dot{x}, x, \mathbb{A}, \mathbb{G}) = \left(\dot{x} - \sum_{\substack{i=1 \\ i\neq k}}^{N} \mathbb{A}_i \prod_{j=1}^{d} x_j^{\mathbb{G}_{ij}} \right) \Big/ \prod_{j=1}^{d} x_j^{\mathbb{G}_{kj}},$$

$$\mathcal{G}_{k\ell}(\dot{x}, x, \mathbb{A}, \mathbb{G}) = \log \left(\left(\dot{x} - \sum_{\substack{i=1 \\ i\neq k}}^{N} \mathbb{A}_i \prod_{j=1}^{d} x_j^{\mathbb{G}_{ij}} \right) \Big/ \left(\mathbb{A}_k \prod_{\substack{j=1 \\ j\neq \ell}}^{d} x_j^{\mathbb{G}_{kj}} \right) \right) \Big/ \log x_\ell.$$

Note that $\mathcal{A}_k$ does not depend on the interval parameter $\mathbb{A}_k$, and similarly, $\mathcal{G}_{k\ell}$ does not depend on $\mathbb{G}_{k\ell}$. The fact that we can completely factor out each parameter improves the efficiency of the constraints, but it is not necessary for the method to work, see [8].

Given the data $\{x(t_i); \dot{x}(t_i)\}_{i=1}^{M}$ we have M constraints per parameter to propagate through:

$$\begin{aligned} \alpha_k &\in \mathbb{A}_k \cap \mathcal{A}_k(\dot{x}(t_i), x(t_i), \mathbb{A}, \mathbb{G}) \\ g_{k,\ell} &\in \mathbb{G}_{k,\ell} \cap \mathcal{G}_{k\ell}(\dot{x}(t_i), x(t_i), \mathbb{A}, \mathbb{G}) \end{aligned} \quad (i = 1, \ldots, M). \tag{7}$$

There are several possibilities to apply these constraints; we choose to loop through all parameters at each time-step. This procedure is implemented in Algorithm 4.1, where we write a and g in place of the boxes $\mathbb{A}$ and $\mathbb{G}$, respectively. Line 5 in the code corresponds to Eq. (5), and line 9 corresponds to Eq. (6). Note that we check for consistency immediately after each constraint is applied (lines 6 and 10). Here, consistency simply means that the contracted domain is still non-empty. If this is not the case, we immediately terminate the process, returning the boolean value `false`. If, on the other hand, all constraints are consistent with the data, the procedure returns the value `true`. As a side effect, it also modifies the parameter boxes a and g according to any contractions incurred by the constraints.

Algorithm 4.1.

```
1   bool constraintPropagate(a, g, xDot, x)    // Modifies parameter boxes a and g.
2   {
3     for (int i = 1; i <= M; i++)    // Loop through M sample points.
4       for (int k = 1; k <= N; k++){   // Loop through N rate constants.
5         a(k) = intersection(a(k), A(k, xDot(i), x(i), a, g));
6         if ( isEmpty(a(k)) )
7           return false;
8         for (int l = 1; l <= d; l++){   // Loop through d kinetic orders.
9           g(k,l) = intersection(g(k,l), G(k, l, xDot(i), x(i), a, g));
10          if ( isEmpty(g(k,l)) )
11            return false;
12        }
13      }
14    return true;
15  }
```

4.1. The main algorithm

Given the data $\{x(t_i); \dot{x}(t_i)\}_{i=1}^{M}$ generated from some target system, the search for feasible parameter values is divided into d component-wise searches. In what follows we will focus on a single such search.

Each search takes place within a global parameter region $\mathbb{P}$, which is initialized as a box whose bounds are determined by biochemical knowledge, see e.g., [21,24]. In principle, our method can take a search region of any size, and still produce a valid result; the computational cost is discussed in Section 5.4. As a first step, we initialize a list `parameterList` with the unique element $\mathbb{P}$. This list is then passed on to the main loop of our search algorithm, listed in Algorithm 4.2:

Algorithm 4.2.

```
1  while ( isEmpty(parameterList) = = false ) {
2    parameter = getCurrent(parameterList); // Here, parameter = (a, g)
3    if ( constraintPropagate(parameter, dataSet) ) // and dataSet = (xDot, x).
4      if ( Diameter(parameter) < Tol )
5        store(parameter, resultList);
6      else
7        splitAndStore(parameter, parameterList);
8  }
```

Within this loop, each member of `parameterList` is tested via the constraints (5) and (6) (on line 3). If these constraints produce a non-empty result, there are two possibilities: either the diameter of the parameter box is smaller than some pre-assigned tolerance `Tol`, in which case the box is stored in a second list `resultList` (line 5); otherwise the parameter box is bisected along its

widest component,[1] and the two resulting sub-boxes are returned to `parameterList` (line 7) for further investigation. If, however, the constraints produce an empty set, the current parameter box is excluded from the remaining search (we go from line 3 directly to line 8). When the search terminates (the argument of line 1 evaluates to `false`), `resultList` contains all sub-boxes of size $\approx$`Tol` satisfying the constraints. If this list is empty, we have established that the corresponding network topology is not consistent with our data at this level of resolution. If, on the other hand, the list has elements $\mathbb{P}_1, \ldots, \mathbb{P}_K$, we can form a *guaranteed* enclosure $\mathbb{P}^\star$ of the solution set $\Gamma_f(\mathbb{P})$ by simply taking the union of all elements of the list: $\mathbb{P}^\star = \cup_{i=1}^{K} \mathbb{P}_i$.

It is not unusual to have access to sample data from several trajectories, that is, trajectories emanating from different initial points $x^{(1)}(t_1), \ldots, x^{(R)}(t_1)$. We can then augment the constraints (7) to take this additional information into account:

$$\begin{array}{l} \alpha_k \in \mathbb{A}_k \cap \mathcal{A}_k(\dot{x}^{(j)}(t_i), x^{(j)}(t_i), \mathbb{A}, \mathbb{G}) \\ g_{k,\ell} \in \mathbb{G}_{k,\ell} \cap \mathcal{G}_{k\ell}(\dot{x}^{(j)}(t_i), x^{(j)}(t_i), \mathbb{A}, \mathbb{G}). \end{array} \quad (i = 1, \ldots, M; \quad j = 1, \ldots, R) \tag{8}$$

This additional data improves our method, seeing that it becomes easier to contract/discard parameter regions. In our experience we have found that it is often wiser to extend the sample data by adding samples from new trajectories (increasing R), rather than increasing the number of samples points (M) on already existing trajectories.

5. Computational results

To illustrate the applicability of our method we apply it to two low-dimensional GMA-systems (one of which is an S-system) considered in [24,25]. As in [25] (and [11,15,22] where similar low-dimensional examples are considered), to examine the performance of our method under optimal conditions we use noise-free data.

Time-series data $\{x(t_i)\}_{i=1}^{M}$ was generated via the `MATLAB ode45` solver. In addition, to decouple the systems we supplemented the data with slopes $\{\dot{x}(t_i)\}_{i=1}^{M}$ obtained by evaluating the differential equations at the data points. It should be pointed out that the sample times $t_1, \ldots, t_M$ are non-uniformly distributed. We chose a logarithmic distribution of the sample times, in order to capture the higher initial reaction rates.

The actual parameter estimation was carried out by a `C++` program, utilizing the PROFIL/BIAS interval package [16]. The computations were performed on a single 1200 MHz Intel Pentium M processor using 384MB of RAM.

5.1. A 3-dimensional GMA-system

We first consider the GMA-system (also studied in [15]) presented in Fig. 1.

Here, the *topology* is assumed to be known, i.e., we know which parameters appear as non-zero quantities in the differential equation. Furthermore, we also use information regarding *dependencies*. More precisely, we know that the second term of the first component matches the first term

[1] This is the simplest, but by no means the most effective subdivision strategy.

of the second component, and that the third term of the first component matches the first term of the third component. Thus, all in all, we are to determine the values of 13 distinct parameters, arranged in two 5-dimensional problems (the parameters of the second and third components, respectively), and one 3-dimensional problem (the three leading parameters of the first component).

For the computations, we used 10 sets of initial conditions, and each trajectory was sampled at 20 points in time. The search region for each kinetic order g_{ij} and rate constant a_i was formed by embedding each true parameter value in an interval with a radius proportional to the modulus of the true value, e.g., $\mathbb{A}_i = [(1-\rho)a_i, (1+\rho)a_i]$. The relative radii (100ρ) considered were $\{0, 10, 50, 100, 200, 300, 400, 500\}$. The stopping tolerance was set to 1×10^{-4} for the second and third component of the GMA-system. This produced parameter enclosures no wider than 4.61×10^{-4}. These were then inserted into the first component of the GMA-system, after which the remaining three parameters were solved for using the tolerance 1×10^{-3}. This setup produced the correct parameter values rounded to three significant digits.

In Table 1, we present the timings for the estimation, as well as the number of parameter boxes examined during the search.

We were surprised to note that the 3-dimensional subproblem (estimating the three leading parameters of the first component) was immediate. By this, we mean that the search only required one single pass through the constraints before satisfying the stopping tolerance. This resembles the situation presented in Example 3.3, illustrating the potential strength of using constraints as contraction mappings.

5.2. A 4-dimensional GMA-system

Our second example presented in Fig. 2 falls into the category of S-systems.

Again, the *topology* is assumed to be known, i.e., we know which parameters appear as non-zero quantities in the differential equation. Contrary to the previous example, however, we no longer use any information regarding *dependencies*. That is, we do *not* use the fact that the second term of the second component matches the first term of the third component, and so on. Thus, all in all, we are to determine the values of 17 distinct parameters, arranged in three 4-dimensional problems (the parameters of the first, second and fourth components, respectively), and one 5-dimensional problem (the parameters of the third component).

For the computations, we used 5 sets of initial conditions, and each trajectory was sampled at 20 points in time. The search region for each kinetic order g_{ij} and rate constant a_i was formed by embedding each true parameter value in an interval with a radius proportional to the modulus of the true value, e.g., $\mathbb{A}_i = [(1-\rho)a_i, (1+\rho)a_i]$. The relative radii (100ρ) considered were again $\{0, 10, 50, 100, 200, 300, 400, 500\}$. The stopping tolerance was set to 1×10^{-3}, which in all cases produced the correct parameter values rounded to three significant digits. In Table 2, we present

Table 1
The computational effort for the GMA-system

Relative radius (%)	0	10	50	100	200	300	400	500
CPU time (seconds)	0.02	3.99	5.08	5.68	6.88	8.50	10.77	13.63
Examined boxes	3	827	1067	1185	1463	1835	2341	3063

Table 2
The computational effort for the S-system

Relative radius (%)	0	10	50	100	200	300	400	500
Time (in seconds)	0.00	2.22	2.94	3.24	3.55	3.83	4.03	4.36
Examined boxes	4	734	988	1072	1190	1292	1376	1494

the timings for the estimation, as well as the number of parameter boxes examined during the search.

In [23,25], the search region for each of the kinetic orders g_{ij} was set to $[-1, +1]$, whereas the rate constants α_i were sought for within the domain $[0, 20]$. Using the same data set and stopping tolerance as above required a total running time of 4.09 s, assuming the correct topology, but using no prior information regarding dependencies. During the search a total of 1364 parameter boxes were examined. The estimated parameter values were, again, correct to at least three significant digits.

5.3. Determining the topology

To further test our method, we also examined its ability to determine the topology of the S-system just described. As a consequence of our set-valued approach, the proposed method discards incorrect topologies very rapidly. Therefore, we can search through increasingly dense topologies until we find the 'sparsest match'. Starting with topologies of cardinality one (i.e., those having only one non-zero kinetic parameter), it took 55 s to find the solution, and an additional 14 s to complete the search through the remaining topologies of the same cardinality as the solution. This should be compared to the method presented in [23], which took 3.5 h on a slightly more challenging problem (using the exact same hardware). The timings ($\approx$15 min) reported in [25] were based on a more powerful computer and correspond to a run-time of roughly 1 h. Furthermore, the parameter values obtained in [25] were only correct to one significant digit.

5.4. Computational complexity

Given the topology of an GMA-system, the worst-case computational complexity of our method is proportional to dr^k, where d is the dimension of the system, k is the maximal number of non-zero parameters in a single component, and r is the size of the largest search domain for a single parameter. Thus, if $k \ll d$, then an increase in the dimension only incurs a linear increase in computational cost. For many biochemical systems, this is indeed often the case: only a very limited number of components react directly with each other. With regards to the size of the search domain, the observed performance of our method is not as bad as one could expect: doubling the width of a single parameter domain seldom doubles the computational cost. This is most likely due to the efficient exclusion/contraction of subdomains.

If the topology is unknown, but a template GMA-system with say m potentially non-zero parameters per component is given, then the upper bound becomes $d\binom{m}{k}r^k$. In the special case of the S-systems, we always have $m \leqslant 2(d+1)$, and the upper bound becomes $d\binom{2(d+1)}{k}r^k$.

Since we generally assume no information regarding dependencies, all d searches can be performed in parallel, which then reduces the complexity bound by a factor d. In the situation where

we do have access to the systems's dependencies, this can be used to considerably reduce the computational cost. For a serial computation of a d-dimensional system, one would simply insert the already computed parameter enclosures into the remaining components before reconstruction. For a parallel computation, there would have to be some interprocess communication, where the current enclosures for the same parameters are intersected.

6. Concluding comments

We have presented a novel method for estimating parameters using interval analysis in combination with constraint propagation. In particular, we have applied it to estimate parameters in GMA-systems and obtained results that improve upon previously described parameter estimation methods in this setting. We stress that the proposed method is quite general, and can (in principle) be applied to *any* system of finitely parameterized differential equations. It can also be used as a pre-processing stage for any other solver, seeing that it simply contracts the global search space.

Our method differs in a fundamental way from the main-stream parameter estimation methods in that we solve the problem by a pruning scheme based on a contraction principle, rather than recasting the estimation as a global minimization problem. One advantage with this approach is that, with a finite dataset, there is no reason to expect a unique solution. In fact, several disjoint regions in parameter space could be the natural answer. As our method simply contracts/discards portions of the parameter space under scrutiny, according to their consistency with the underlying data, it can handle this scenario with only small modifications in the I/O routines.

The transition to set-valued vector fields also allows us to dismiss unrealistic network topologies. In particular, this allows us to detect when the model we are trying to fit to the provided data is not appropriate; with a sufficiently small stopping tolerance, our method will then discard *all* parameter values.

To demonstrate the performance of our new method we have chosen some rather low-dimensional, noise-free examples, which are somewhat unrealistic. Our point, however, is to first investigate our proposed method under optimal conditions, in much the same way that the performance of the methods introduced in [11,15,22,25] is examined. Although some kind of pre-processing (e.g., smoothing) can significantly reduce if not remove noise, degradation of the performance of our method due to noisy data and comparison with other methods needs to be studied next. In principle, the described method is already geared to work for noisy data: instead of data points (t_i, x_i, s_i), we can use intervals $(\mathbb{T}_i, \mathbb{X}_i, \mathbb{S}_i)$ with almost no change to the code at all. The main challenge is to obtain reasonably good slope ranges $\mathbb{S}_i$ from the noisy data. Therefore, we think that our method is likely to work well for noisy data for large data sets, but not for sparse, noisy data. Moreover, in light of our new method's potential for parallelization, we will need to start investigating its performance for much larger systems.

Ultimately, it is unrealistic to expect a reconstruction method to provide tight parameter estimates for most biological systems. Small data sets may not always suffice to uniquely determine the structure of the underlying system, and, perhaps more importantly, it may be the case the several solutions are consistent with the data [25]. In these situations, however, it should be possible to adapt our method to produce disjoint sets of solutions, thus providing biologists with alterna-

tive models from which to select, and directions for new experiments to distinguish between the various alternatives.

References

[1] G. Alefeld, J. Herzberger, Introduction to Interval Computations, Academic Press, New York, 1983.
[3] CXSC – C++ eXtension for Scientific Computation, version 2.0. Available from <http://www.math.uni-wuppertal.de/org/WRST/xsc/cxsc.html>.
[5] Forte Developer 7: C++ Interval Arithmetic Programming Reference. Available from <http://docs.sun.com/app/docs/doc/816-2465>.
[7] INTLAB – INTerval LABoratory, version 4.1.2. Available from <http://www.ti3.tu-harburg.de/~rump/intlab/>.
[8] L. Jaulin, M. Kieffer, O. Didrit, Applied Interval Analysis, Springer, London, 2001.
[9] L. Jaulin, Nonlinear bounded-error state estimation of continuous-time systems, Automatica 38 (6) (2002) 1079.
[10] D. Kell, Metabolomics and systems biology: making sense of the soup, Curr. Opin. Microbiol. 7 (2004) 296.
[11] S. Kikuchi, D. Tominaga, M. Arita, K. Takahashi, M. Tomita, Dynamic modeling of genetic networks using genetic algorithm and S-system, Bioinformatics 19 (5) (2003) 643.
[12] P. Mendes, D. Kell, Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation, Bioinformatics 14 (10) (1998) 869.
[13] R.E. Moore, Interval Analysis, Prentice-Hall, Englewood Cliffs, New Jersey, 1996.
[14] R.E. Moore, Methods and Applications of Interval Analysis, SIAM Studies in Applied Mathematics, Philadelphia, 1979.
[15] P.K. Polisetty, E.O. Voit, E.P. Gatzke, Identification of metabolic system parameters using global optimization methods, Theor. Biol. Med. Model. (2006), 3:4. Available from <http://www.tbiomed.com/content/3/1/4>.
[16] PROFIL/BIAS – Programmer's Runtime Optimized Fast Interval Library/Basic Interval Arithmetic Subroutines. Available from <http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html>.
[17] T. Raïssi, N. Ramdani, Y. Candau, Set membership state and parameter estimation for systems described by nonlinear differential equations, Automatica 40 (10) (2004) 1771.
[18] M.A. Savageau, Biochemical systems analysis. I. Some mathematical properties of the rate law for the component enzymatic reactions, J. Theor. Biol. 25 (1969) 365.
[19] M.A. Savageau, Biochemical systems analysis. II. The steady-state solutions for an n-pool system using a power-law approximation, J. Theor. Biol. 25 (1969) 370.
[20] M.A. Savageau, Biochemical systems analysis. 3. Dynamic solutions using a power-law approximation, J. Theor. Biol. 26 (1970) 215.
[21] N.V. Torres, E.O. Voit, Pathway Analysis and Optimization in Metabolic Engineering, Cambridge University Press, Cambridge, 2002.
[22] K. Tsai, F.-S. Wang, Evolutionary optimization with data collection for reverse engineering of biological networks, Bioinformatics 21 (2005) 1180.
[23] W. Tucker, V. Moulton, Reconstructing metabolic networks using interval analysis, Lect. Notes Comput. Sci. 3692 (2005) 192.
[24] E.O. Voit, Computational Analysis of Biochemical Systems, Cambridge University Press, 2000.
[25] E.O. Voit, J. Almeida, Decoupling dynamical systems for pathway identification from metabolic profiles, Bioinformatics 20 (11) (2004) 1670.
[26] W.G. Walster, E. Hansen, Global Optimization Using Interval Analysis, Series in Pure and Applied Mathematics, vol. 264, CRC Press, 2003.