Reconstructing Metabolic Networks Using Interval Analysis

Warwick Tucker[1] and Vincent Moulton[2]

[1] Department of Mathematics, Uppsala University, Box 480, Uppsala, Sweden
warwick@math.uu.se
http://www.math.uu.se/~warwick
[2] School of Computing Sciences, University of East Anglia,
Norwich, NR4 7TJ, UK
Vincent.Moulton@cmp.uea.ac.uk
http://www.cmp.uea.ac.uk

Abstract. Recently, there has been growing interest in the modelling and simulation of biological systems. Such systems are often modelled in terms of coupled ordinary differential equations that involve parameters whose (often unknown) values correspond to certain fundamental properties of the system. For example, in metabolic modelling, concentrations of metabolites can be described by such equations, where parameters correspond to the kinetic rates of the underlying chemical reactions. Within this framework, the increasing availability of time series data opens up the attractive possibility of reconstructing approximate parameter values, thus enabling the *in silico* exploration of the behaviour of complex dynamical systems. The parameter reconstruction problem, however, is very challenging – a fact that has resulted in a plethora of heuristics methods designed to fit parameters to the given data.

In this paper we propose a completely deterministic method for parameter reconstruction that is based on interval analysis. We illustrate its utility by applying it to reconstruct metabolic networks using S-systems. Our method not only estimates the parameters very precisely, it also determines the appropriate network topologies. A major strength of the proposed method is that it proves that large portions of parameter space can be disregarded, thereby avoiding spurious solutions.

1 Introduction

A well-known and difficult problem in metabolic modeling is that of *parameter reconstruction*. A metabolic model is often given in terms of 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. The problem is then to search for a particular $p^\star$ within a parameter space $\mathbb{P}$ such that the solutions of the system $\dot{x} = f(x; p^\star)$ match a given data set, in some pre-specified manner. Typically, the data set is a time series, that is, samples taken along one or several trajectories of the target system.

R. Casadio and G. Myers (Eds.): WABI 2005, LNBI 3692, pp. 192–203, 2005.

As in many other settings, parameter reconstruction in metabolic modelling is often recast as a global optimization problem. Due to the high dimensionality of the problem, however, straight-forward optimization strategies rarely produce accurate parameter values. Today, most methods used for parameter reconstruction are thus based on heuristic algorithms such as, for example, machine learning, genetic algorithms and PL-models – see [9] for a recent overview. In this paper, we describe a very general and completely deterministic approach to solving the parameter reconstruction problem, 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.

Although our new approach is very general, we will focus on a particular class of differential equations commonly used to model biochemical networks, known as *S-systems* [17]. These have been extensively studied (see e.g. [5,3,10,7,18]), and have the appealing feature that the underlying metabolic network topology can be estimated along with the other parameters. In addition, several methods have been recently described for parameter reconstruction in S-systems (e.g. [10,18,16]).

2 Methods

2.1 Component-Wise Reconstruction via Slopes

Suppose that we are given a d-dimensional system of ordinary differential equations $\dot{x} = f(x; p)$, sampled at N distinct times (excluding the initial point, which is assumed to be known at time t_0), producing the data set $\{x(t_j)\}_{j=0}^{N}$, where each sample $x(t_j) = (x_1(t_j), \ldots, x_d(t_j))$ has d components. Rather than attempting to reconstruct parameters by solving the entire system $\dot{x} = f(x; p)$, it can be more helpful to obtain more detailed information localized at individual sample points. One way to do this is to use the samples to reconstruct the trajectories (e.g. via piece-wise splines) with some degree of smoothness. This enables the computation of an approximation of the vector field at each sample point:

$$s_{i,j} \approx f_i(x(t_j); p^\star), \qquad i = 1, \ldots, d; \quad j = 0, \ldots, N.$$

The number $s_{i,j}$ corresponds to the slope of the trajectory's i:th component at time t_j, see Figure 1.

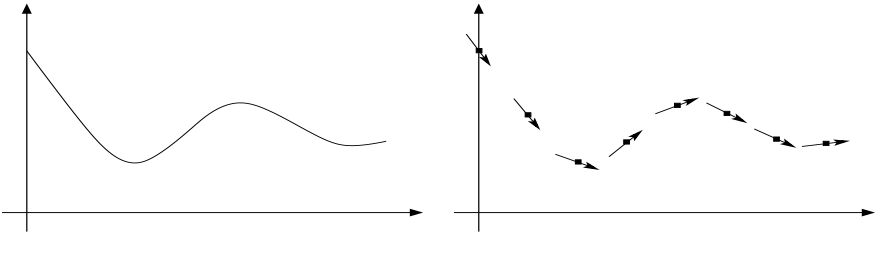

Fig. 1. (a) One component of a trajectory. (b) Sample data with slopes.

Equipped with this *enhanced* sample data, we can try to locate a point in $\mathbb{P}$ that minimizes the *defect*:

$$\Delta(p) = \sum_{i=1}^{d} \Delta_i(p) \stackrel{\text{def}}{=} \sum_{i=1}^{d} \sum_{j=0}^{N} \|f_i(x(t_j); p) - s_{i,j}\|,$$

for some convenient norm $\|\cdot\|$. Using the defect as a measurement of quality is not new, see e.g. [6] or, in the context of S-systems, [18]. The major advantage of this approach is that the system *decouples*, i.e., the computation of each $\Delta_i(p)$ depends only on a fraction of the total number of parameters: $\Delta_i(p) = \Delta_i(p_i)$, where $p_i \in \mathbb{P}_i$, and $\mathbb{P} = \mathbb{P}_1 \oplus \cdots \oplus \mathbb{P}_d$.

Assuming that each p_i has k (potential) components, the total dimension of the entire search space $\mathbb{P}$ is dk. Rather than searching through a dk-dimensional space, access to the enhanced sample data allows us to perform d independent searches in k-dimensions. The gain is immediate: introducing M grid-points in each parameter domain produces M^{dk} points in the first case, but only dM^k points in the latter. This gives a speed-up factor of M^d/d.

We point out that, at present, our proposed computation of the slopes is not very noise-tolerant. There are, however, several possible remedies that we aim to explore in the future. One possibility is to use piece-wise splines with set-valued coefficients. This approach fits well into the framework that we present below. Another option is to simply smooth the data (via e.g. least-squares) before fitting the splines.

2.2 Interval-Valued Slopes

Our approach is a modification of the enhanced data method, and therefore shares the same attractive decomposition property of the global parameter space $\mathbb{P}$. The major improvement is that we now compute ranges of slopes for entire domains of parameters. In essence, we extend the vector field f to a set-valued function F, accepting solid blocks in parameter space as input. The theoretical justification for this type of extension is given shortly. Let $[p_i]$ denote a box in $\mathbb{P}_i$, i.e., each component of $[p_i]$ is an interval. Then, for any point $p_i \in [p_i]$, we have

$$f_i(x(t_j); p_i) \in F_i(x(t_j); [p_i]),$$

i.e., the set $F_i(x(t_j); [p_i])$ contains *all possible* slopes corresponding to parameters taken from the box $[p_i]$. This fact gives us a simple criterion for discarding portions of the search space $\mathbb{P}_i$: if a box $[p_i]$, at a sample point $x(t_j)$, produces a range of slopes such that $s_{i,j} \notin F_i(x(t_j); [p_i])$, then *no* parameter in $[p_i]$ can have generated the sample data. If this situation occurs, we say that the parameter box $[p_i]$ violates the *cone condition* at time t_j, see Figure 2.

Our strategy in reconstructing the target parameter $p^\star$ is to adaptively partition each space $\mathbb{P}_i$ into successively smaller sub-boxes, retaining only those that satisfy the cone condition at all times. At some pre-selected level of coarseness,

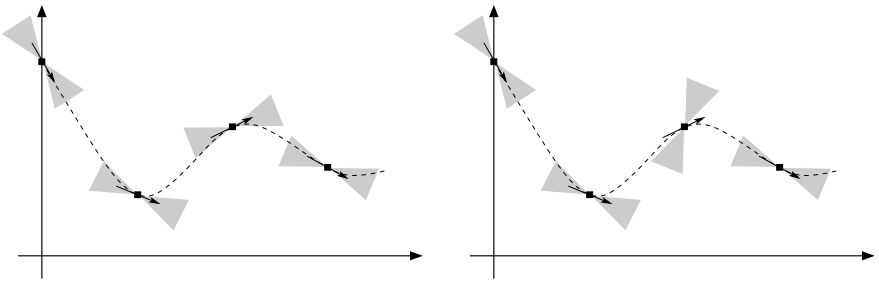

Fig. 2. (a) Cone condition satisfied at t_0, t_1, t_2, and t_3. (b) Violated at time t_2.

we terminate the process, and are left with a collection of boxes $[p_i^{(1)}], \ldots, [p_i^{(n)}]$, each of which satisfies $\mathcal{I}([p_i^{(j)}]) = \texttt{true}$, where

$$\mathcal{I}([p_i]) = \bigwedge_{j=0}^{N} \Big(s_{i,j} \in F_i(x(t_j); [p_i]) \Big) \tag{1}$$

is a boolean function that returns `true` if $[p]$ satisfies the cone condition at all sample times, and `false` otherwise.

2.3 Interval Analysis

Here, we will briefly describe the fundamentals of interval analysis. For a concise reference on this topic, see e.g. [12].

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

$$[a] \star [b] = \{a \star b \colon a \in [a], b \in [b]\},$$

except that $[a] \div [b]$ is undefined if $0 \in [b]$. Working exclusively with closed intervals, we can describe the resulting interval in terms of the endpoints of the operands:

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

When computing with finite precision, directed rounding must also be taken into account (see e.g. [11,13]).

A key feature of interval arithmetic is that it is *inclusion monotonic*, i.e., if $[a] \subseteq [A]$, and $[b] \subseteq [B]$, then

$$[a] \star [b] \subseteq [A] \star [B], \tag{3}$$

where we demand that $0 \notin [B]$ for division.

One of the main reasons for passing to the interval realm is that we want a simple way of enclosing the *range* $R(f;D) = \{f(x)\colon x \in D\}$ of a real-valued function $f\colon D \to \mathbb{R}$. 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 $[x]$ (and the real arithmetic operators with their interval counterparts) produces a rational interval function $F([x])$, called the *natural* interval extension of f. As long as no singularities are encountered, we have the inclusion $R(f;[x]) \subseteq F([x])$, by property (3). In fact, this type of range enclosure can be achieved for any reasonable function [12].

Higher-dimensional functions $f\colon \mathbb{R}^n \to \mathbb{R}$ can be extended to an interval function $F\colon \mathbb{IR}^n \to \mathbb{IR}$ in a similar manner. The function argument is then an *interval-vector* $[x] = ([x_1],\ldots,[x_n])$, which we also refer to as a *box*.

There exist several open source programming packages for interval analysis [4,8,14].

2.4 S-Systems

An S-system is a system of ordinary differential equations on the form:

$$\dot{x}_i = \alpha_i \prod_{j=1}^{d} x_j^{g_{ij}} - \beta_i \prod_{j=1}^{d} x_j^{h_{ij}} \qquad (i = 1,\ldots,d). \tag{4}$$

Each variable x_i represents the concentration of some reactant, and $\dot{x}_i$ denotes the time derivative of x_i. In a biochemical context, the non-negative parameters α_i and β_i are called *rate constants*. The real-valued parameters g_{ij} and h_{ij} are referred to as the *kinetic orders*. Each component of an S-system is made up of one positive and one negative term, corresponding to the production and consumption of the substance x_i, respectively. In essence, an S-system is a condensed version of a more general GMA – General Mass Action – model, obtained by aggregating individual reactions into the net processes of synthesis and degradation, see [15].

Using the following short-hand notation for the parameters

$$p_i = (\alpha_i, g_{i1},\ldots,g_{id},\beta_i,h_{i1},\ldots,h_{id}) \qquad (i = 1,\ldots,d),$$

we can express (4) more compactly as $\dot{x}_i = f_i(x;p_i)$. The entire S-system then becomes $\dot{x} = f(x;p)$. A d-dimensional S-system has $2d(d+1)$ parameters, so already for small systems the number of parameters becomes unwieldy. We reduce the number of parameters by assuming that no reactant x_j influences both the rate of production *and* the rate of degradation of another reactant x_i (see [1,18]). This assumption can be reformulated more succinctly as:

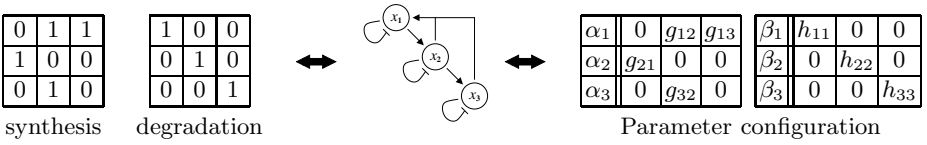

Fig. 3. A boolean topology encodes a metabolic network, and determines the parameter configuration of the S-system in (6)

$$g_{ij} \neq 0 \Rightarrow h_{ij} = 0, \tag{5}$$

and reduces the total number of non-zero parameters to $d(d+2)$, although we now must consider 2^d different parameter configurations for each component of the vector field f_i. Nevertheless, this is a good trade: filling each of the $2(d+1)$ parameter domains of f_i with M grid-points produces $M^{2(d+1)}$ points, compared to $2^d M^{d+2}$ points when using (5). This gives a speed-up factor of $(M/2)^d$.

A simple example of an S-system (appearing in [17] pp. 179-184) is given by:

$$\begin{aligned} \dot{x}_1 &= 7.5x_2^{-0.1}x_3^{-0.05} - 5x_1^{0.5} \\ \dot{x}_2 &= 2x_1^{0.5} - 1.44x_2^{0.5} \\ \dot{x}_3 &= 3x_2^{0.5} - 7.2x_3^{0.5}. \end{aligned} \tag{6}$$

The corresponding metabolic network is illustrated in Figure 3. This is an example of a *cascade* mechanism, which commonly appear in the context of gene regulation and immunology.

2.5 Set-Valued S-Systems

Extending the right-hand side of (4) to accept parameter boxes as input is a simple matter, and produces a vector field $F\colon \mathbb{R}^d \to \mathbb{R}^d$ whose components are interval-valued:

$$\dot{x}_i \in F_i(x;[p_i]) = [\alpha_i]\prod_{j=1}^{d} x_j^{[g_{ij}]} - [\beta_i]\prod_{j=1}^{d} x_j^{[h_{ij}]}. \tag{7}$$

It is easy to show ([2], p. 23) that this extension is *sharp*, i.e.,

$$R(f_i(x;\cdot);[p_i]) = F_i(x;[p_i]).$$

This sharpness property is not necessary for our method to work, but it does make it more efficient.

We briefly comment that it is possible to allow for uncertain data in the sense that the exact measurements x_j appearing the right-hand side of (7) be replaced by intervals $[x_j]$. This option will be explored in conjunction with the interval-based slope construction, mentioned in section 2.1.

2.6 The Main Algorithm

Given a collection of enhanced sample data $\{x_{i,j}; s_{i,j}\}_{i,j}$ generated from some target S-system with parameter $p^\star = (p_1^\star, \ldots, p_d^\star)$, the search is divided into d independent component-wise searches for $p_1^\star, \ldots, p_d^\star$. These $(d+2)$-dimensional searches can be performed as d parallel processes, seeing that they are completely independent. In what follows we fill focus on a single such search. For clarity, we will suppress the component index i.

Each search takes place within a global parameter region $\mathbb{P}$, which is initialized as a box $\mathbb{P} = ([P_1], \ldots, [P_{2(d+1)}])$. The bounds for this box are determined by biochemical knowledge (e.g. [17]). Utilizing the constraints (5), we initialize all 2^d possible different parameter configurations $\tilde{\mathbb{P}}_1, \ldots, \tilde{\mathbb{P}}_{2^d}$, each having $d+2$ non-zero parameters, and corresponding to different network topologies. Having done this, we examine each $\tilde{\mathbb{P}}_i$ separately (or all $\tilde{\mathbb{P}}_i$ in parallel). As a first step, we initialize a list `parameterList` with the unique element $\tilde{\mathbb{P}}_i$. This list is then passed on to the main loop of our search algorithm.

```
while( isEmpty(parameterList) == false ) {
  parameter = getCurrent(parameterList);
  if ( coneCondition(parameter) == true ) {
    if ( diameter(parameter) > Tol )
      splitAndStore(parameter, parameterList);
    else
      store(parameter, resultList);
  }
}
```

Within this loop, each member of `parameterList` is tested via the cone condition (1). If the condition is satisfied, 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`; otherwise it is bisected along its widest component, and the two resulting sub-boxes are returned to `parameterList` for further investigation. If, however, the cone condition is not satisfied, the current parameter box is excluded from the remaining search. When the search terminates, `resultList` contains all sub-boxes of size $\approx$ `Tol` satisfying the cone condition. If this list is empty, we have established that this particular network topology does not match our data.

Often, we have have access to sample data from several trajectories, that is, trajectories emanating from different initial points $x^{(1)}(t_0), \ldots, x^{(M)}(t_0)$. We can then modify the cone condition (1) to take this additional information into account:

$$\mathcal{I}([p_i]) = \bigwedge_{j=0}^{N} \bigwedge_{k=1}^{M} \left(s_{i,j}^{(k)} \in F_i(x^{(k)}(t_j); [p_i]) \right)$$

This additional data improves our method, seeing that it becomes easier to discard parameter regions.

3 Computational Results

Starting with sample data $\{t_j; x_{i,j}\}_{i,j}$ generated from some target S-system with parameter $p^\star$, we first generate the slope data $\{s_{i,j}\}_{i,j}$, as described earlier. These computations are performed by a collection of `Matlab` scripts, using its built-in spline functionality. This allows us to differentiate the reconstructed trajectories, and recover the slopes. It should be pointed out that the data itself is generated within `Matlab`, and that the sample times $t_0, \ldots, t_N$ are non-uniformly distributed. We choose a logarithmic distribution of the sample times, in order to capture the more vivid motion occurring for small times. In the examples presented below we use noise-free sample data $\{x_{i,j}\}_{i,j}$, as discussed earlier.

The actual parameter reconstruction is carried out by a prototype `C++` program, utilizing a modified version of the PROFIL/BIAS interval package [14]. The computations were performed on a single 1200MHz Intel Pentium M processor using 384MB of RAM.

3.1 A Fixed-Topology Cascade

Our first example is the S-system (6) corresponding to the network presented in Figure 3. Note that, since we are given the network topology a priori, the computational complexity of parameter reconstruction is significantly reduced.

For the computations, we used five sets of initial conditions, and each trajectory was sampled at 20 points in time. Following [17], the search region for each of the kinetic orders g_{ij}, and h_{ij} was set to contain $[-1, +1]$, whereas the rate orders α_i and β_i were sought for within $[0, 15]$. The stopping tolerance (i.e., the diameters of the final parameter intervals) was set to 1×10^{-3}. In Table 1, we present the target parameters together with the final result of our reconstruction. We use the notation "—" to indicate the, a priori, non-present parameters. The agreement is seen to be almost perfect. The entire search took 1 minute and 6 seconds.

The reconstructed parameters appearing in Table 1 were obtained as follows: when the global search has terminated, we are left with a collection of parameter

Table 1. The original parameter values (A) and their reconstructions (B) for the S-system (6)

i	α_i	g_{i1}	g_{i2}	g_{i3}	β_i	h_{i1}	h_{i2}	h_{i3}
A								
1	7.5	—	−0.1	−0.05	5.0	0.5	—	—
2	2.0	0.5	—	—	1.44	—	0.5	—
3	3.0	—	0.5	—	7.2	—	—	0.5
B								
1	7.49	—	−0.100	−0.0503	4.99	0.501	—	—
2	2.00	0.501	—	—	1.44	—	0.502	—
3	3.00	—	0.500	—	7.20	—	—	0.500

boxes $[p_1], \ldots, [p_K]$, all satisfying the cone condition. We reduce these boxes to one single box $[\mathbb{P}^\star]$ by forming their *hull* – the smallest box containing all parameter boxes $[p_1], \ldots, [p_K]$. We then have an enclosure of the target parameter $p^\star \in [\mathbb{P}^\star]$. Of course, taking the hull of all parameter boxes is a rather crude measure. We get a better feeling for where the center of mass of the boxes is located by computing the average of the collection of parameter boxes. In order to get a single point in parameter space as our "best guess", we simply take the midpoint of the average:

$$\bar{\mathbb{P}}^\star = \mathtt{Mid}\left(\frac{1}{K}\sum_{i=1}^{K}[p_i]\right).$$

It is the components of the resulting $\bar{\mathbb{P}}^\star$ that are presented in Table 1. Note, however, that any choice of parameters from one of the resulting boxes $[p_1], \ldots, [p_K]$ is consistent with our sample data.

Also note that our computations *prove* that parameters outside the produced ranges do not match the sample data. Considering e.g. the h_{33}-parameter of (6), we found that $h_{33} \in [0.496, 0.503]$. The remaining parameters were enclosed as follows:

$$\begin{aligned}
(\alpha_1, \alpha_2, \alpha_3) &\in ([7.34, 7.62], [1.96, 2.03], [2.98, 3.03]) \\
(g_{12}, g_{13}) &\in ([-0.103, -0.0982], [-0.0527, -0.0486]) \\
(g_{21}, g_{32}) &\in ([0.492, 0.509], [0.493, 0.506]) \\
(\beta_1, \beta_2, \beta_3) &\in ([4.84, 5.13], [1.40, 1.46], [7.18, 7.23]) \\
(h_{11}, h_{22}) &\in ([0.485, 0.519], [0.489, 0.517]).
\end{aligned}$$

Interestingly, some of the parameters reconstructed in [17] did not fall in the parameter intervals that we computed. For example, even when starting the search with initial guesses close to the true values, the "quasi-Newton" algorithm used in [17] produced e.g. $\alpha_1 = 9.237$, $\beta_3 = 3.236$, $h_{22} = 0.0397$, all of which our algorithm has *proved* to be unsuitable. This example was also studied in [16] using four different methods. The outcomes for e.g. α_3 were 1.25, 7.70, 7.3, and 1.45, respectively. Note that none of these values belong to the parameter enclosure $\alpha_3 \in [2.98, 3.03]$ produced by our method.

3.2 A 4-Dimensional S-System

Our second example appears in [18]:

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

In this example we are not given the network topology, which makes the parameter reconstruction significantly more demanding. For the computations, we used

five sets of initial conditions, and each trajectory was sampled at 20 points in time. Following [18], the search region for each of the kinetic orders g_{ij}, and h_{ij} was set to contain $[-1, +1]$, whereas the rate orders α_i and β_i were sought for within $[0, 20]$. The stopping tolerance was set to 2×10^{-3}. Note that, although each component of the vector field has 10 parameters to be determined, we use the constraints (5) to bring the number of non-zero parameter values down to six, which can be arranged in 16 different network topologies.

Table 2. The parameter values (and their reconstructions) of the S-system (8)

i	α_i	g_{i1}	g_{i2}	g_{i3}	g_{i4}	β_i	h_{i1}	h_{i2}	h_{i3}	h_{i4}
Original										
1	12	0.0	0.0	−0.8	0.0	10	0.5	0.0	0.0	0.0
2	8	0.5	0.0	0.0	0.0	3	0.0	0.75	0.0	0.0
3	3	0.0	0.75	0.0	0.0	5	0.0	0.0	0.5	0.2
4	2	0.5	0.0	0.0	0.0	6	0.0	0.0	0.0	0.8
Reconstructed										
1	12.00	0.0	0.0	−0.802	0.0	9.98	0.501	0.0	0.0	0.0
2	7.96	0.502	0.0	0.0	0.0	2.96	0.0	0.757	0.0	0.0
3	2.95	0.0	0.759	0.0	0.0	4.95	0.0	0.0	0.504	0.202
4	2.00	0.501	0.0	0.0	0.0	6.00	0.0	0.0	0.0	0.800

In Table 2, we present the target parameters together with the final result of our reconstruction. Once again, the agreement is seen to be almost perfect. The entire search took 3 hours, 29 minutes, and 27 seconds. This great increase in time, compared to the three-dimensional example, appears to indicate that the method scales very badly. Note, however, that this increase is mostly due to the fact that we were not given the topology of the four-dimensional system.

The reconstructed parameters appearing in Table 2 were obtained as in the previous example, but with one additional twist: after having computed the midpoint of the average, we set any parameter with value less than 5×10^{-4} to zero:

$$\bar{\mathbb{P}}^{\star} = \texttt{cutOff}\left(\texttt{Mid}\left(\frac{1}{K}\sum_{i=1}^{K}[p_i]\right); 5 \times 10^{-4}\right).$$

This *skeletalizing* procedure promotes sparse network topologies; in [18], the cut-off level is set to 1×10^{-1}.

4 Discussion

We have presented a novel method for reconstructing parameters using interval analysis. In particular, we have applied it to reconstruct metabolic networks using S-systems, and obtained very encouraging results. We stress that the proposed method is very general, and can be applied to *any* system of finitely parameterized differential equations.

Our method differs in a fundamental way from the main-stream reconstruction methods in that we solve the problem by a pruning scheme based on a boolean function (the cone condition), rather than recasting the parameter reconstruction as a global minimization problem. This has several advantages: First, it is well-known that global minimization is an intractable problem, in the sense that numerical solutions often converge to a local, rather than a global, minimum, and there is no way of telling the two cases apart. Second, the quantity to be minimized is often chosen to be a (weighted) least-square error. This implicitly pre-assumes rather strong statistical properties of the underlying data, assumptions that can not easily be verified. Our method simply retains the parameters that are consistent with the underlying data, avoiding both above-mentioned problems.

The transition to set-valued vector fields also allows us to dismiss, with a mathematical certainty, 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. At a sufficiently low tolerance, our method would then discard *all* parameter values.

In future work, we will refine the process of parameter exclusion, and exploit the problem's great potential for parallelization. This is an essential step towards exploring the scalability of our proposed method. We will also allow for noisy sample data, using interval-valued cubic splines in the generation of the slopes. We also plan to put our method to test on a larger class of problems (including generalized mass action models).

Acknowledgement

The authors would like to thank Korbinian Strimmer for introducing them to S-systems, and for helpful discussions.

References

1. Akutsu, T., Miyano, S. & Kuhara, S. Inferring qualitative relations in genetic networks and metabolic pathways, *Pacific Symposium on Biocomputing* **5** (2000) 120–301.
2. Alefeld, G. & Herzberger, J. *Introduction to Interval Computations*. Academic Press, New York (1983).
3. Alves, R. & Savageau, M. A. Comparing systemic properties of ensembles of biological networks by graphical and statistical methods, *Bioinformatics* **16:6** (2000) 527–533.
4. CXSC – C++ eXtension for Scientific Computation, version 2.0. Available from `www.math.uni-wuppertal.de/org/WRST/xsc/cxsc.html`
5. de Jong, H. Modeling and Simulation of Genetic Regulatory Systems: A Literature Review, *J. Comp. Biol.* **9:1** (2002) 67–103.
6. Enright, W. H. A New Error-Control for Initial Value Solvers, *Applied Mathematics and Computation* **31** (1998) 288–301.

7. Hlavacek, W. S. & Savageau, M. A. Rules for Coupled Expressions of Regulator and Effector Genes in Inducible Circuits, *J. Mol. Biol.* **255** (1996) 121–139.
8. INTLAB – INTerval LABoratory, version 4.1.2. Available from `www.ti3.tu-harburg.de/~rump/intlab/`
9. Kell, D. *Current Opinion in Microbiology* **7** (2004) 296–307.
10. Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K. & Tomita, M. Dynamic modeling of genetic networks using genetic algorithm and S-system, *Bioinformatics* **19:5** (2003) 643–650.
11. Kulisch, U. W. & Miranker, W. L. *Computer Arithmetic in Theory and Practice*. Academic Press, New York (1981).
12. Moore, R. E. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey (1966).
13. Moore, R. E. *Methods and Applications of Interval Analysis*. SIAM Studies in Applied Mathematics, Philadelphia (1979).
14. PROFIL/BIAS – Programmer's Runtime Optimized Fast Interval Library/Basic Interval Arithmetic Subroutines. Available from `www.ti3.tu-harburg.de/Software/PROFILEnglisch.html`
15. Torres, N. V & Voit, E. O. *Pathway Analysis and Optimization in Metabolic Engeneering*. Cambridge University Press, Cambridge (2002).
16. Tsai, K. & Wang, F. Evolutionary optimization with data collocation for reverse engineering of biological networks *Bioinformatics* Advance Access published online on October 28 (2004).
17. Voit, E. O. *Computational Analysis of Biochemical Systems*. Cambridge University Press, Cambridge (2000).
18. Voit, E. O. & Almeida, J. Decoupling dynamical systems for pathway identification from metabolic profiles, *Bioinformatics* **20:11** (2004) 1670–168