Reliable Computing (2006) **12:** 389–402
DOI: 10.1007/s11155-006-9009-2

Parameter Reconstruction for Biochemical Networks Using Interval Analysis

WARWICK TUCKER
Department of Mathematics, Uppsala University, Box 480, Uppsala, Sweden,
e-mail: warwick@math.uu.se

and

VINCENT MOULTON
School of Computing Sciences, University of East Anglia, Norwich, NR4 7TJ, UK,
e-mail: Vincent.Moulton@cmp.uea.ac.uk

(Received: 9 September 2005; accepted: 19 February 2006)

Abstract. In recent years, the modeling and simulation of biochemical networks has attracted increasing attention. Such networks are commonly modeled by systems of ordinary differential equations, a special class of which are known as S-systems. These systems are specifically designed to mimic kinetic reactions, and are sufficiently general to model genetic networks, metabolic networks, and signal transduction cascades. The parameters of an S-system correspond to various kinetic rates of the underlying reactions, and one of the main challenges is to determine approximate values of these parameters, given measured (or simulated) time traces of the involved reactants.

Due to the high dimensionality of the problem, a straight-forward optimization strategy will rarely produce correct parameter values. Instead, almost all methods available utilize genetic/evolutionary algorithms to perform the non-linear parameter fitting. We propose 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. The proposed method can in principle be applied to any system of finitely parameterized differential equations, and, as we demonstrate, yields encouraging results for low dimensional S-systems.

1. Introduction

A typical, and very general, problem in applied mathematics is that of *parameter reconstruction*. The goal of parameter reconstruction is to choose values for the parameters in a model to best describe some given set of data.

In the context of differential equations, the model can be 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 task 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 the given data set, in some pre-specified manner.

Typically, the data set consists of samples along one or several trajectories of the target system $\dot{x} = f(x; p^\star)$, see Figure 1. A trajectory of a d-dimensional system, sampled at N distinct times (excluding the initial point, which is assumed to be known at time t_0), produces the data set $\{x(t_j)\}_{j=0}^{N}$, where $x(t_j) = (x_1(t_j), \ldots, x_d(t_j))$.

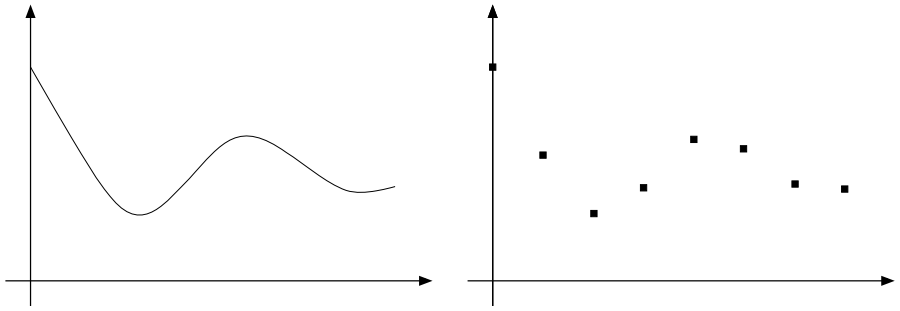

Figure 1. (a) One component of a trajectory. (b) Its sample data.

In what follows, we will use the short-hand notation $x_{ij} = x_i(t_j)$. We will also assume that the problem of reconstructing the parameters is *overdetermined*, that is, the cardinality of the data set is greater than the number of parameters to be determined.

1.1. Simultaneous Reconstruction via ODE Solving

Given a set of samples, one approach is to try to locate a point $p \in \mathbb{P}$ that minimizes the *data* error:

$$\mathcal{E}^{(\mathrm{data})}(p) = \sum_{i=1}^{d} \mathcal{E}_i^{(\mathrm{data})}(p) \stackrel{\mathrm{def}}{=} \sum_{i=1}^{d} \sum_{j=0}^{N} \varrho\Big(\varphi_i\big(t_j, x(t_0); p\big), x_{ij}\Big), \tag{1.1}$$

for some convenient metric $\varrho(\cdot,\cdot)$. Here, φ solves the differential equation:

$$\frac{d}{dt}\varphi(t,x;p) = f\big(\varphi(t,x;p);p\big), \qquad \varphi(0,x;p) = x, \tag{1.2}$$

and can thus be approximated by numerical means, given the parameter p. This approach is expensive seeing that, for each p, the entire system of differential equations must be solved to compute the component errors $\mathcal{E}_i^{(\mathrm{data})}(p)$.

1.2. Component-Wise Reconstruction via Slopes

Rather than attempting to reconstruct the parameter by solving the entire system of differential equations, it may prove wiser to obtain more detailed information localized at the individual sample points. One possibility is to use the samples to reconstruct the trajectories (e.g. via piece-wise splines) with some degree of smoothness. This enables us to compute an approximation of the vector field at each sample point:

$$s_{ij} \approx f_i\big(x(t_j);p^\star\big), \qquad i = 1,\ldots,d;\ \ j = 0,\ldots,N. \tag{1.3}$$

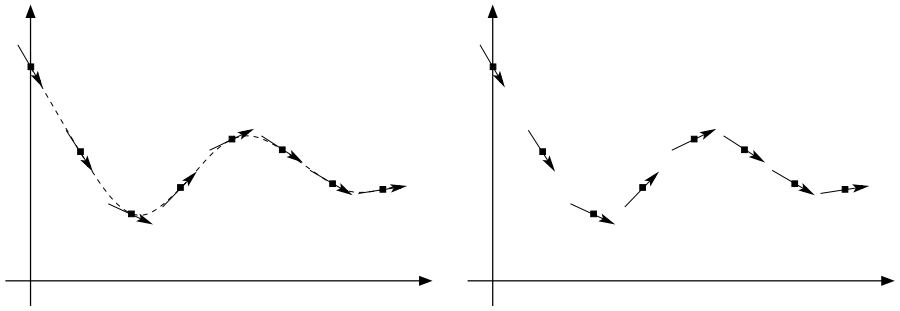

Figure 2. (a) One component of a trajectory. (b) Its *enhanced* sample data.

The number s_{ij} corresponds to the slope of the trajectory's ith component at time t_j, see Figure 2.

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

$$\mathcal{E}^{(\text{slope})}(p) = \sum_{i=1}^{d} \mathcal{E}_i^{(\text{slope})}(p) \stackrel{\text{def}}{=} \sum_{i=1}^{d} \sum_{j=0}^{N} \varrho\Big(f_i\big(x(t_j);p\big), s_{ij}\Big), \tag{1.4}$$

for some convenient metric $\varrho(\cdot,\cdot)$. The major advantage with this approach is that the system *decouples*, that is, the computation of each $\mathcal{E}_i^{(\text{slope})}(p)$ depends only on a fraction of the total number of parameters: $\mathcal{E}_i^{(\text{slope})}(p) = \mathcal{E}_i^{(\text{slope})}(p_i)$, 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 at most k non-zero 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 component produces M^{dk} points in the first case, but only dM^k points in the latter. This gives a speed-up factor of $M^d \,/\, d$.

Remark 1.1. As we only use approximations $s_{ij} \approx f_i(x(t_j);p^\star)$, we can not claim to be able to reconstruct the target parameter $p^\star$ with mathematical rigor. This limitation, however, is mild compared to the underlying assumption that the true trajectories are well-described by the data set $\{x(t_j)\}_{j=0}^N$. In practice, given a reasonable amount of data, a good approximation of the target parameter is obtained. Poorly estimated slopes s_{ij} will result in our algorithm dismissing *all* parameters at a very early stage, and are thus easily spotted.

Remark 1.2. When decoupling the system, information whether several components of the vector field share a common parameter is lost. Although it is possible to re-inject this information in the parameter reconstruction (e.g. communication between

parallel processes), it is not a trivial task. We use no such information in our algorithm.

1.3. Interval-Valued Slopes

Our approach is a modification of the enhanced data method, and therefore shares the same attractive decoupling property of the system, as described above. 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 boxes in parameter space as input. The mathematical justification for this type of extension is based on the theory of interval analysis. For a concise reference on this topic, see e.g. [1], [7]–[10]. For early papers, see [12], [15], and [16].

Let $[p_i]$ denote a box in $\mathbb{P}_i$, that is, each component of $[p_i]$ is an interval. Then, for any point $p_i \in [p_i]$, we have

$$f_i\big(x(t_j); p_i\big) \in F_i\big(x(t_j); [p_i]\big), \tag{1.5}$$

that is, 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 $s_{ij} \notin F_i\big(x(t_j); [p_i]\big)$, 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 3.

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

$$\mathcal{I}([p_i]) \stackrel{\text{def}}{=} \bigwedge_{j=0}^{N} \Big(s_{ij} \in F_i\big(x(t_j); [p_i]\big) \Big) \tag{1.6}$$

is a boolean function that returns `true` if $[p_i] \in \mathbb{P}_i$ satisfies the cone condition at all sample times, and `false` otherwise.

The strategy of dismissing subsets that are inconsistent with some constraint lies at the heart of many interval methods. In the context of ODEs, see e.g. [3] and references within.

In Section 2, we will introduce a particular family of differential equations whose parameters we will reconstruct using the method described above.

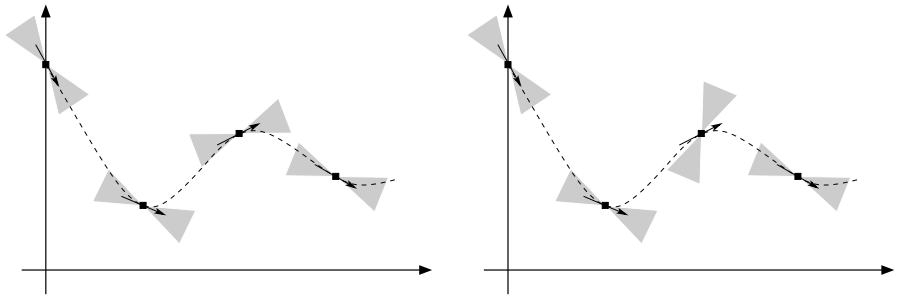

Figure 3. (a) Cone condition satisfied at t_0, t_1, t_2, and t_3. (b) Violated at time t_2.

2. S-systems

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

$$\dot{x}_i = \alpha_i \prod_{j=1}^{d} x_j^{g_{ij}} - \beta_i \prod_{j=1}^{d} x_j^{h_{ij}} \qquad (i = 1, \ldots, d). \tag{2.1}$$

These systems are designed to model quite general biochemical networks, see [13], and have been extensively studied, see e.g. [2], [4]–[6], and [14]. 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*. Thus 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.

Using the following short-hand notation for the parameters

$$p_i = (\alpha_i, g_{i1}, \ldots, g_{id}, \beta_i, h_{i1}, \ldots, h_{id}) \qquad (i = 1, \ldots, d), \tag{2.2}$$

we can express (2.1) more compactly as $\dot{x}_i = f_i(x; p_i)$. The entire S-system then becomes $\dot{x} = f(x; p)$. Two typical trajectories of a 5-dimensional S-system are illustrated in Figure 4.

The general problem of parameter reconstruction can be divided into two smaller tasks: first, one may determine the network *topology*. This is a qualitative property of the system that describes whether a reactant suppresses/induces the synthesis/degradation of another reactant. A key step in this stage is to determine which parameters are non-zero. Second, one is interested in the *rate* at which the synthesis/degradation occurs. This corresponds to finding approximate values of the non-zero parameters.

A d-dimensional S-system has $2d(d+1)$ parameters, so already for small systems the number of parameters becomes unwieldy, see Table 1.

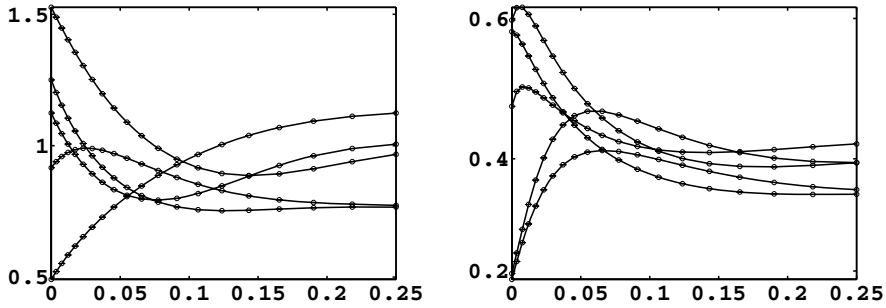

Figure 4. Typical trajectories of a 5-dimensional S-system.

Table 1. The total number of parameters in a d-dimensional S-system.

d	1	2	3	4	5	10	15
$2d(d+1)$	4	12	24	40	60	220	480

We will reduce the number of parameters by assuming that no reactant x_j both produces *and* consumes another reactant x_i. This assumption can be reformulated more succinctly as:

$$g_{ij} \neq 0 \Rightarrow h_{ij} = 0, \tag{2.3}$$

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. 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 (2.3). This gives a speed-up factor of $(M/2)^d$.

2.1. Set-Valued S-Systems

Extending the right-hand side of (2.1) to accept parameter boxes as input is a simple matter, and produces a set-valued vector field whose components are interval-valued:

$$\dot{x}_i \in F_i(x;[p_i]) = [\alpha_i]\prod_{j=1}^{d} x_j^{[g_{ij}]} - [\beta_i]\prod_{j=1}^{d} x_j^{[h_{ij}]} \qquad (i = 1,\ldots,d). \tag{2.4}$$

Since each parameter occurs only once in (2.4), it follows that this extension is *sharp*, that is,

$$R\big(f_i(x;\cdot);[p_i]\big) \stackrel{\text{def}}{=} \{f_i(x;p_i)\colon p_i \in [p_i]\} = F_i(x;[p_i]). \tag{2.5}$$

This sharpness property is not necessary for our method to work, but makes it more effective.*

3. The Main Algorithm

Given a collection of sample data $\{x_{ij}; s_{ij}\}_{i,j}$ 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, see e.g. [13]. Utilizing the constraints (2.3), we initialize 2^d different parameter configurations $\widetilde{\mathbb{P}}_1, \ldots, \widetilde{\mathbb{P}}_{2^d}$, each having $d+2$ non-zero parameters, and corresponding to different network topologies. Having done this, we examine each $\widetilde{\mathbb{P}}_i$ separately (or all $\widetilde{\mathbb{P}}_i$ in parallel). As a first step, we initialize a list `parameterList` with the unique element $\widetilde{\mathbb{P}}_i$. 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 (1.6). 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 the corresponding network topology does not match our data at this level of resolution.

* We have recently obtained encouraging preliminary results for a more general class of ODEs in which a parameter can occur several times.

** Using (2.3), there are d non-zero kinetic rates, and two rate constants per component of f.

It is not unusual to 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 augment the cone condition (1.6) to take this additional information into account:

$$\mathcal{I}([p_i]) \stackrel{\text{def}}{=} \bigwedge_{j=0}^{N} \bigwedge_{k=1}^{M} \left(s_{ij}^{(k)} \in F_i\big(x^{(k)}(t_j); [p_i]\big) \right). \tag{3.1}$$

This additional data improves our method, seeing that it becomes easier to discard parameter regions. It is often wiser to extend the sample data by adding samples from new trajectories, rather than increasing the number of samples points on already existing trajectories.

4. Computational Results

We have tested our method on three S-systems, and obtained encouraging results on all occasions.

Starting with the sample data $\{t_j; x_{ij}\}_{i,j}$ generated from some target S-system with parameter $p^\star$, we first generate the additional slope data $\{s_{ij}\}_{i,j}$, as described in Section 1.2. These computations are performed by a collection of `MATLAB` scripts, utilizing 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, see Figure 4. In the examples presented below we use noise-free sample data $\{x_{ij}\}_{i,j}$.

The actual parameter reconstruction was carried out by a prototype `C++` program, utilizing a modified version of the PROFIL/BIAS interval package [11]. The computations[⋆] were performed on a single 1200 MHz Intel Pentium M processor using 384 MB of RAM.

4.1. A 4-Dimensional S-System

The following S-system is taken from [14]:

$$\begin{aligned} \dot{x}_1 &= 12x_3^{-0.8} - 10x_1^{0.5}, \\ \dot{x}_2 &= 8x_1^{0.5} - 3x_2^{0.75}, \\ \dot{x}_3 &= 3x_2^{0.75} - 5x_3^{0.5}x_4^{0.2}, \\ \dot{x}_4 &= 2x_1^{0.5} - 6x_4^{0.8}. \end{aligned} \tag{4.1}$$

[⋆] We used directed rounding which, in light of Remark 1.1 of Section 1.2, is probably not called for. Swithing off the rounding reduces the reported CPU-times by roughly 25%.

Table 2. The parameter values (and their reconstructions) of the S-system (4.1).

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

We remind the reader that we do *not* know the topology of the S-system. This means that each component of the ODE is only known to have the following shape:

$$\dot{x}_i = \alpha_i x_1^{g_{i1}} x_2^{g_{i2}} x_3^{g_{i3}} x_4^{g_{i4}} - \beta_i x_1^{h_{i1}} x_2^{h_{i2}} x_3^{h_{i3}} x_4^{h_{i4}} \qquad i = 1, \ldots, 4.$$

For the computations, we used five sets of initial conditions, and each trajectory was sampled at 20 points in time. 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 (2.3) to bring the number of non-zero parameter values down to six, which can be arranged in 16 different network topologies.

In Table 2, we present the target parameters together with the final result of our reconstruction. The agreement is seen to be quite satisfactory.

The reconstructed parameters appearing in Table 2 were obtained as follows: when the global search has terminated, we are left with a collection of parameter boxes $[p_1], \ldots, [p_K]$, all satisfying the cone condition. We may reduce these boxes to one single box $[\mathbb{P}^\star]$ by forming the *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 forming the average of the collection of parameter boxes. In order to get a single point in parameter space as our "best guess," we take the midpoint of the average, and set any number less than 5×10^{-4} to zero:

$$\overline{\mathbb{P}}^\star = \operatorname{trunc}\left(\operatorname{mid}\left(\frac{1}{K}\sum_{i=1}^{K}[p_i]\right); 5 \times 10^{-4}\right). \tag{4.2}$$

Table 3. The computational effort for the S-system (4.1) with $\texttt{Tol} = 2 \times 10^{-3}$.

component	boxes	F-evaluations	CPU-time
1	83,987	5,620,510	1h 21m 33s
2	7,609	2,976,928	42m 23s
3	777	3,768,046	53m 36s
4	7,085	3,378,982	48m 36s

Here, we use the truncation function

$$\operatorname{trunc}(x;\varepsilon) = \begin{cases} 0 & \text{if } |x| \leq \varepsilon, \\ x & \text{if } |x| > \varepsilon. \end{cases} \tag{4.3}$$

It is the components of the resulting $\overline{\mathbb{P}}^{\star}$ that are presented in Table 2. Note, however, that any choice of parameters from one of the resulting boxes $[p_1], \ldots, [p_K]$ is consistent with our sample data.

The computational effort is presented in Table 3, where we list the number of final parameter boxes together with the number of required vector field evaluations.

This system was also studied in [14], in which the reconstruction was performed by a data smoothing followed by a non-linear regression combined with some simplifying assumptions not present in our method. The reported timings ($\approx$ 15 minutes) were based on a much more powerful computer (dual 2.4 GHz CPUs with 4 GB RAM). Also, some of the reported parameter values were off by more than 9%, whereas our values are correct to 1%.

4.2. A 5-Dimensional S-System

The following S-system is a slight modification of that studied in [4] and [6]:

$$\begin{aligned} \dot{x}_1 &= 5x_3x_5^{-1} - 10x_1^2, \\ \dot{x}_2 &= 10x_1^2 - 10x_2^2, \\ \dot{x}_3 &= 10x_2^{-1} - 10x_3^2, \\ \dot{x}_4 &= 8x_3^2x_5^{-1} - 10x_4^2, \\ \dot{x}_5 &= 10x_4^2 - 10x_5^2. \end{aligned} \tag{4.4}$$

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 of the kinetic orders g_{ij}, and h_{ij} was set to contain $[-3, +3]$, whereas the rate orders α_i and β_i were sought for within $[0, 15]$. The stopping tolerance was set to 2×10^{-2}. Note that, although each component of the vector field has 12 parameters to be determined, we use the constraints (2.3) to bring the number of non-zero parameter values down to seven, which can be arranged in 32 different network topologies.

Table 4. The parameter values of the S-system (4.4), and their reconstructions.

i	α_i	g_{i1}	g_{i2}	g_{i3}	g_{i4}	g_{i5}	β_i	h_{i1}	h_{i2}	h_{i3}	h_{i4}	h_{i5}
Original												
1	5	0	0	1	0	−1	10	2	0	0	0	0
2	10	2	0	0	0	0	10	0	2	0	0	0
3	10	0	−1	0	0	0	10	0	0	2	0	0
4	8	0	0	2	0	−1	10	0	0	0	2	0
5	10	0	0	0	2	0	10	0	0	0	0	2
Reconstructed												
1	5.00	0.0	0.0	0.999	0.0	−1.00	10.00	2.00	0.0	0.0	0.0	0.0
2	9.99	2.00	0.0	0.0	0.0	0.0	9.99	0.0	2.00	0.0	0.0	0.0
3	10.00	0.0	−1.00	0.0	0.0	0.0	10.00	0.0	0.0	2.00	0.0	0.0
4	7.99	0.0	0.0	2.00	0.0	−1.00	9.98	0.0	0.0	0.0	2.00	0.0
5	10.00	0.0	0.0	0.0	2.00	0.0	10.00	0.0	0.0	0.0	0.0	1.99

Table 5. The computational effort for the S-system (4.4) with `Tol` = 2×10^{-2}.

component	boxes	F-evaluations	CPU-time
1	5,789	1,635,320	49m 30s
2	4,528	1,906,320	56m 13s
3	13,132	2,091,374	1h 9m 11s
4	1,627	690,578	20m 26s
5	906	889,318	27m 5s

In Table 4, we present the target parameters together with the final result of our reconstruction. We used the average/truncation procedure described in the previous section, with truncation level 5×10^{-3}. Again, the agreement is seen to be very good. The computational effort is presented in Table 5.

A very similar system was studied in [6], in which the reconstruction was performed via a modified least-squares fit. The resulting optimization problem was solved by a genetic algorithm. The reported timings (several days) were based on a AIST CBRC Magi Cluster with 1040 CPUs (pentium III 933 MHz). Also, some of the reported parameter values were off by more than 18%, whereas our values are correct to 0.2%.

4.3. A FIXED-TOPOLOGY CASCADE

Our final example deals with a *cascade* mechanism. These appear in e.g. gene regulation and immunology.

Table 6. The parameter values (and their reconstructions) of the S-system (4.5).

i	α_i	g_{i1}	g_{i2}	g_{i3}	β_i	h_{i1}	h_{i2}	h_{i3}
Original								
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
Reconstructed								
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

Table 7. The computational effort for the fixed-topology S-system (4.5) with $\texttt{Tol} = 1 \times 10^{-3}$.

component	boxes	F-evaluations	CPU-time
1	905	83,595	54s
2	34	11,073	7s
3	83	8,733	6s

$$\begin{aligned} \dot{x}_1 &= 7.5x_2^{-0.1}x_3^{-0.05} - 5x_1^{0.5}, \\ \dot{x}_2 &= 2x_1^{0.5} - 1.44x_2^{0.5}, \\ \dot{x}_3 &= 3x_2^{0.5} - 7.2x_3^{0.5}. \end{aligned} \tag{4.5}$$

This particular model is treated in [13], and differs from the two previous examples in that we are given, a priori, the network topology. This reduces the computational complexity significantly.

In Table 6, we present the target parameters together with the final result of our reconstruction. The reconstructed parameter values are simply the midpoint of the average over all parameter boxes produced by our search. We use the notation "—" to indicate a non-present parameter.

For the computations, we used five sets of initial conditions, and each trajectory was sampled at 20 points in time. 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 was set to 1×10^{-3}. Once again, the agreement is seen to be a good match. The computational effort is presented in Table 7.

5. Conclusions

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 encouraging results. We stress that the proposed method

is quite general, and can (in principle) 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 discards the parameters that are inconsistent with the underlying data, avoiding both above-mentioned problems.

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 low stopping 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 statistical pre-processing 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).

References

1. Alefeld, G. and Herzberger, J.: *Introduction to Interval Computations*, Academic Press, New York, 1983.
2. Alves, R. and Savageau, M. A.: Comparing Systemic Properties of Ensembles of Biological Networks by Graphical and Statistical Methods, *Bioinformatics* **16** (6) (2000), pp. 527–533.
3. Deville, Y., Janssen, M., and van Hentenryck, P.: Consistency Techniques in Ordinary Differential Equations, *Constraints* **7** (2002), pp. 289–315.
4. Hlavacek, W. S. and Savageau, M. A.: Rules for Coupled Expressions of Regulator and Effector Genes in Inducible Circuits, *J. Mol. Biol.* **255** (1996), pp. 121–139.
5. de Jong, H.: Modeling and Simulation of Genetic Regulatory Systems: A Literature Review, *J. Comp. Biol.* **9** (1) (2002), pp. 67–103.
6. Kikuchi, S. et al.: Dynamic Modeling of Genetic Networks Using Genetic Algorithm and S-System, *Bioinformatics* **19** (5) (2003), pp. 643–650.
7. Kulisch, U. W. and Miranker, W. L.: *Computer Arithmetic in Theory and Practice*, Academic Press, 1981.
8. Moore, R. E.: *Interval Analysis*, Prentice Hall, Englewood Cliffs, 1966.
9. Moore, R. E.: *Methods and Applications of Interval Analysis*, SIAM Studies in Applied Mathematics, Philadelphia, 1979.
10. Neumaier, A.: Interval Methods for Systems of Equations, *Encyclopedia of Mathematics and Its Applications* **37**, Cambridge Univ. Press, Cambridge, 1990
11. PROFIL/BIAS—Programmer's Runtime Optimized Fast Interval Library/Basic Interval Arithmetic Subroutines,
`http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html`

12. Sunaga, T.: Theory of an Interval Algebra and Its Application to Numerical Analysis, *RAAG Memoirs* **2** (1958), pp. 29–46.
13. Voit, E. O.: *Computational Analysis of Biochemical Systems*, Cambridge University Press, 2000.
14. Voit, E. O. and Almeida, J.: Decoupling Dynamical Systems for Pathway Identification from Metabolic Profiles, *Bioinformatics* **20** (11) (2004), pp. 1670–1681.
15. Warmus, M.: Calculus of Approximations, *Bulletin de l'Academie Polonaise de Sciences* **4:5** (1956), pp. 253–257.
16. Young, R. C.: The Algebra of Multi-Valued Quantities, *Mathematische Annalen* **104** (1931), pp. 260–290.