Journal of Scientific Computing manuscript No.
(will be inserted by the editor)

Parameter range reduction in ordinary differential equation models

Andrew Skelton · Allan R. Willms

This is a post-print of the article by the same title published in:
The Journal of Scientific Computing, Vol. 62 (2) pp. 517–531, (2015).
doi = 10.1007/s10915-014-9865-6

Abstract This paper presents an algorithm for parameter range reduction in systems of ordinary differential equations. Parameter values are assumed only to be known to lie in potentially large regions of parameter space. Interval arithmetic and a family of monotonic discretizations are used to prune regions of parameter space that are inconsistent with given time series data. The algorithm is tested on two ordinary differential equation models and the reduced ranges are shown to significantly improve the performance of traditional parameter estimation methods.

Keywords parameter estimation · ordinary differential equations · numerical methods · interval arithmetic

PACS 34A55 · 65L06

1 Introduction

Given a system of ordinary differential equations (ODEs) that models some physical process, we are interested in the inverse problem of identifying an appropriate set of parameter values from time series data. We will, in particular, be concerned with models of the form

$$x' = f(t, x, \lambda), \qquad x \in \mathbb{R}^m, \lambda \in \mathbb{R}^q, \tag{1}$$

where the model parameters, λ, are to be determined from time series data of the form $S = \{(t^n_{\text{obs}}, x^n_{\text{obs}})\}_{n=1}^N$. We denote the observation time window

A. Skelton
University of Guelph, Guelph, Ontario, Canada, N1G 2W1
E-mail: skeltona@uoguelph.ca

A.R. Willms
E-mail: awillms@uoguelph.ca

$I_{\text{obs}} = [t^1_{\text{obs}}, t^N_{\text{obs}}]$ and for a fixed λ, the solution to (1) passing through the initial point (τ, ξ) as $x(t, \tau, \xi, \lambda)$.

Parameter identification in this context is typically accomplished by selecting a set of initial parameter values, numerically integrating the model equations and comparing the result to the time series data. A cost function, such as weighted least squares, is used to measure the suitability of these parameters. Using estimates of how the cost function varies with the parameters, new parameter values are chosen until the cost function is hopefully minimized.

It is often the case that little is known a priori about the parameter values, thus making an initial selection difficult. If the initial selection must be made from a very large region of parameter space, it is possible that this selection will result in a system that cannot be numerically integrated over the desired time window. If the cost function has multiple local minima, or large flat regions in parameter space, it may be difficult for the procedure to converge to a reasonable minimum. To combat this problem, one could employ a multistart method [12] in which a large number of initial parameter selections are made and the results are analyzed to identify a global minimizer. An alternative is a simulated annealing method [2] in which the algorithm attempts to efficiently explore as much of parameter space as possible. A variety of global methods are compared in [1,3,6]. In all cases, significant computational time is spent numerically integrating the model equations. It is useful to be able to reduce the size of the parameter space each method is required to search.

This paper presents an improvement on a parameter range reduction method first introduced in [13,14]. The algorithm discretizes the model equations and uses these discretizations to quickly prune regions of parameter space that are deemed to be inconsistent with the data. The final result is a collection of boxes from which a better initial parameter selection can be made. The method in [13,14] required that the discretization over a given time window be monotonic with respect to each parameter. We relax here this condition and show that it results in a smaller region of parameter space that fails to be inconsistent. For simplicity, we will refer to such regions as *consistent regions*.

The remainder of this paper is organized as follows. In Section 2, we provide a brief summary of interval arithmetic and present some theorems that will allow for better and more efficient parameter range reduction. In Section 3, we discuss the theory and structure of the algorithm itself. In Section 4, we present experimental results obtained from two ODE models and show the effect of reduced parameter ranges on traditional parameter estimation schemes.

2 Interval Arithmetic and Notation

We now briefly introduce some concepts in interval arithmetic. More detailed and complete reviews of interval analysis can be found in [4,7–9]. In this paper, interval-valued quantities will be denoted with a bold typeface, while under and over bars will denote respectively the lower and upper endpoints of an interval. If $\mathbf{a} = [\,\underline{\mathbf{a}}\,, \overline{\mathbf{a}}\,]$ and $\mathbf{b} = \left[\,\underline{\mathbf{b}}\,, \overline{\mathbf{b}}\,\right]$ and $\bullet$ denotes any of the operators

$+, -, \times, \div$, then we can define elementary interval arithmetic as

$$\mathbf{a} \bullet \mathbf{b} = \{a \bullet b \ : \ a \in \mathbf{a}, b \in \mathbf{b}\}.$$

This definition is equivalent to the following:

$$\begin{aligned}
\mathbf{a} + \mathbf{b} &= \left[\ \underline{\mathbf{a}} + \underline{\mathbf{b}} \, , \overline{\mathbf{a}} + \overline{\mathbf{b}} \ \right], \\
\mathbf{a} - \mathbf{b} &= \left[\ \underline{\mathbf{a}} - \overline{\mathbf{b}} \, , \overline{\mathbf{a}} - \underline{\mathbf{b}} \ \right], \\
\mathbf{a} \times \mathbf{b} &= \left[\ \min\{\underline{\mathbf{a}}\,\underline{\mathbf{b}}, \underline{\mathbf{a}}\,\overline{\mathbf{b}}, \overline{\mathbf{a}}\,\underline{\mathbf{b}}, \overline{\mathbf{a}}\,\overline{\mathbf{b}}\} \, , \max\{\underline{\mathbf{a}}\,\underline{\mathbf{b}}, \underline{\mathbf{a}}\,\overline{\mathbf{b}}, \overline{\mathbf{a}}\,\underline{\mathbf{b}}, \overline{\mathbf{a}}\,\overline{\mathbf{b}}\} \ \right], \\
\mathbf{a} \div \mathbf{b} &= \left[\ \underline{\mathbf{a}} \, , \overline{\mathbf{a}} \ \right] \times \left[\ \frac{1}{\overline{\mathbf{b}}} \, , \frac{1}{\underline{\mathbf{b}}} \ \right], \quad \text{if } 0 \notin \mathbf{b}.
\end{aligned}$$

Elementary functions can likewise be defined over intervals [4]. If n is a non-negative integer, for example, we can define

$$\mathbf{a}^n = \begin{cases} [1, 1] & \text{if } n = 0, \\ [\ \underline{\mathbf{a}}^n, \overline{\mathbf{a}}^{\,n} \ ] & \text{if } \underline{\mathbf{a}} \geq 0, \text{ or } n \text{ is odd}, \\ [\ \overline{\mathbf{a}}^{\,n}, \underline{\mathbf{a}}^n \ ] & \text{if } \overline{\mathbf{a}} \leq 0, \text{ and } n \text{ is even}, \\ [\ 0, \max(\ \underline{\mathbf{a}}^n, \overline{\mathbf{a}}^{\,n} \ ) \ ] & \text{if } \underline{\mathbf{a}} \leq 0 \leq \overline{\mathbf{a}}, \text{ and } n > 0 \text{ is even}. \end{cases}$$

Given a function $f : D \subset x \to \mathbb{R}$, we wish to determine an interval that encloses as tightly as possible the set $\text{Range}(f; D) = \{f(x) : x \in D\}$. There are many sophisticated interval range enclosure methods [9], but for our purposes we will require only the use of *natural interval extensions*, that are obtained by substituting all occurrences of each argument of f with its interval-valued equivalent. An interval-valued enclosure of the range of a function will be denoted $\mathbf{F} : \mathbf{x} \to \mathbf{I}$. We will denote the upper and lower endpoints of $\mathbf{I}$ respectively by $\underline{\mathbf{F}}(\mathbf{x})$ and $\overline{\mathbf{F}}(\mathbf{x})$. If f is a sufficiently nice function, it has been shown that interval extensions satisfy the property of *inclusion isotonicity* [7], that is,

$$\text{Range}(f; \mathbf{x}) \subseteq \mathbf{F}(\mathbf{x}).$$

If $\text{Range}(f; \mathbf{x}) = \mathbf{F}(\mathbf{x})$, then the interval enclosure $\mathbf{F}$ is said to be *sharp*. Consider, for example, the function $f : [0, 2] \to \mathbb{R}$ given by $f(x) = x - x^2$. It is easy to show that the range of this function is $\text{Range}(f; [0, 2]) = \left[-2, \frac{1}{4}\right]$. Using the natural interval extension $\mathbf{F}(\mathbf{x}) = \mathbf{x} - \mathbf{x}^2$, it can be see that

$$\text{Range}(f; [0, 2]) \subset \mathbf{F}([0, 2]) = [0, 2] - [0, 2]^2 = [0, 2] - [0, 4] = [-4, 2].$$

This interval enclosure is not sharp because of the *dependency problem*; the two instances of $\mathbf{x}$ are incorrectly treated as independent. We can, however, obtain sharpness by considering an alternate form of f. Using the natural interval extension of the alternate form $g(x) = -\left(x - \frac{1}{2}\right)^2 + \frac{1}{4}$, it can be seen that

$$\text{Range}(g; [0, 2]) = \mathbf{G}([0, 2]) = -\left([0, 2] - \frac{1}{2}\right)^2 + \frac{1}{4} = \left[-\frac{9}{4}, \frac{3}{4}\right] + \frac{1}{4} = \left[-2, \frac{1}{4}\right].$$

Since alternate forms of the same expression can give different results when extended to intervals, it is very important to consider the form of the original equation before applying an interval extension. The following theorems give useful conditions for determining if an interval enclosure will be sharp.

Theorem 1 *[7] An interval enclosure* $\mathbf{F}(\mathbf{x})$*, obtained from a natural interval extension, is sharp if and only if both* $\underline{\mathbf{F}}$ *and* $\overline{\mathbf{F}}$ *are computed in terms of a single endpoint of each of the variables on which* F *depends.*

Theorem 2 *[4] If* F *is monotonically nondecreasing or nonincreasing with respect to each argument, then a sharp enclosure* $\mathbf{F}$ *can be calculated by evaluating the function* F *at the appropriate end points of each interval argument.*

We will make frequent use of this monotonicity test. In addition to providing sharpness, establishing monotonicity can significantly decrease the computational time of the algorithm. In this case, the values of $\underline{\mathbf{F}}$ and $\overline{\mathbf{F}}$ must be calculated independently, but do not require the use of interval arithmetic. We note that any checks of monotonicity must be made over the entire range of values in each interval-valued argument. Any partial derivatives used in this analysis are therefore also interval extensions and may themselves fail to be sharp, which can lead to pessimistic results. It is, however, often the case that establishing monotonicity is not a necessary condition for sharpness.

Theorem 3 *[7] If each variable on which* F *depends appears only once in the expression, then the enclosure* $\mathbf{F}$*, obtained from a natural interval extension, is sharp.*

When we are constructing interval extensions, it is therefore useful to consider alternate rearrangements in which each interval-valued quantity, especially those for which monotonic properties cannot be established, appears only once in the given expression. In some cases, such a rearrangement will not be possible, but factoring is in general always beneficial.

Definition 1 [7] The interval arithmetic addition and multiplication operators are sub-distributive. That is, $\mathbf{x}(\mathbf{y}+\mathbf{z}) \subseteq \mathbf{xy}+\mathbf{xz}$.

The lesson to be learned is that the algebraic structure of any expression must be carefully considered before extending it to interval arithmetic if tight enclosures are desired.

3 Parameter Range Reduction

In this section, we outline the parameter range reduction scheme. We first discuss the family of discretizations used by the scheme and their interval extensions. We then discuss monotonicity tests that enable our interval extensions to be as sharp as possible. We then discuss the main test used by the scheme and finally, we outline the structure of the algorithm.

The algorithm requires as input an ordinary differential equation model, initial parameter ranges and a range for each model variable at every time $t \in I_{\text{obs}}$. It is assumed that the true value of each parameter lies within its initial range, that may be as wide as $(-\infty, +\infty)$ if no a priori information is known. Converting the discrete time series datas to a continuous representation may be accomplished using a variety of methods. In this paper, we use an algorithm presented in [10] that replaces time series data with a continuous piecewise linear band that encloses all data points. The results of the range reduction algorithm depend on the representation and thus on what the user thinks is a valid continuous representation that encloses the true solution.

3.1 Discretizations

The algorithm will make use of a specific family of linear multistep discretizations. In [14], it was shown that the best discretization formulae for parameter range reduction were called A1OUT discretizations, whose form is

$$F(t^0, h, s; x^0, x^1, \ldots, x^s, \lambda) := x^0 - x^s + h \sum_{i=0}^{s} \beta_i f^i, \tag{2}$$

where $h > 0$ is the constant step size, s is the number of steps in the discretization, each $x^i = [x_1^i, \ldots, x_m^i]^T$ is an independent variable in $\mathbb{R}^m$ and represents an approximation to the solution $x(t^i, \tau, \xi, \lambda)$ of the ODE at time $t^i = t^0 + ih$, and $f^i = f(t^i, x^i, \lambda)$. The time interval $[t^0, t^s]$ will be referred to as the discretization window. Superscripts are used to denote time indices and subscripts to denote spatial indices. For readability, we will often drop the first three arguments of F, when it is not necessary to emphasize this dependence. The β_i coefficients in equation (2) are chosen using the following criteria:

1. $\beta_i \geq 0, \quad 0 \leq i \leq s$, (monotonicity),
2. $\beta_i = \beta_{s-i}, \quad 0 \leq i \leq \lfloor \frac{s}{2} \rfloor$, (symmetry),
3. β_i are chosen to maximize the order of the discretization and, if this does not uniquely identify them, they are chosen to minimize the error constant as defined in [14].

A table of β_i values for $1 \leq s \leq 17$ can be found in [14] with details of their derivation.

The natural interval extension of (2) is given by

$$\mathbf{F}(t^0, h, s; \mathbf{x^0}, \mathbf{x^1}, \ldots, \mathbf{x^s}, \boldsymbol{\lambda}) := \mathbf{x^0} - \mathbf{x^s} + h \sum_{i=0}^{s} \beta_i \mathbf{f^i},$$

where $\boldsymbol{\lambda} := \boldsymbol{\lambda}_1 \times \boldsymbol{\lambda}_2 \times \cdots \times \boldsymbol{\lambda}_q$ is the Cartesian product of the parameter intervals (the *parameter box*), and $\mathbf{f^i} = \mathbf{f}(t^i, \mathbf{x^i}, \boldsymbol{\lambda})$ is the natural interval extension of f^i. Each $\mathbf{x^i}$ is determined from the continuous representations of the time series data and for simplicity, we will often write $\mathbf{x} := \mathbf{x^0}, \ldots, \mathbf{x^s}$. To obtain a sharper result, we may consider alternate rearrangements of F before deriving the natural interval extension.

3.2 Monotonicity

To ensure that our interval extension is as sharp as possible, we attempt to make use of as many monotonicity properties as possible. The monotonicity property of the A1OUT discretization family allows us to make the following statement [14]. If some component of the vector field, $f_j, 1 \leq j \leq m$, is monotonically nondecreasing with respect to some parameter λ_k over some set Ω in (t, x, λ)-space, then this monotonicity is inherited by the discretization F_j. That is,

$$\frac{\partial f_j}{\partial \lambda_k}(t, x, \lambda) \geq 0,\ \forall (t, x, \lambda) \in \Omega \implies \frac{\partial F_j}{\partial \lambda_k}(t^0, h, s; x^0, \ldots, x^s, \lambda) \geq 0,\ \forall (t^0, h, s),$$

provided that $(t^i, x^i, \lambda) \in \Omega, \forall i \in \{0, \ldots, s\}$. A similar statement is true if f is monotonically non-increasing. The required partial derivatives can be quickly computed over all parameter and variable values in their respective intervals, so we use this as a first test of monotonicity.

It is possible, however, that the vector field fails to be monotonic, yet the discretization applied to a specific discretization time window is monotonic. Consider for example an ODE of the form $x_1' = f_1(x_1, x_2, \lambda_1) = \lambda_1 x_2$. Since $\frac{\partial f_1}{\partial \lambda_1} = x_2$, monotonicity with respect to λ_1 cannot be established if our continuous representation of x_2 spans zero at some time in the discretization window. If we consider the discretization of f_1, however, we can see that

$$\frac{\partial F_1}{\partial \lambda_1}(t^0, h, s; x_1, x_2, \lambda) = h \sum_{i=0}^{s} \beta_i x_2^i.$$

The interval extension of this partial derivative may not span zero, even if some individual $\mathbf{x_2^i}$ does span zero.

The discretization equation (2) may also depend on some of the $m(s+1)$ quantities x_j^i, where $1 \leq j \leq m$ and $0 \leq i \leq s$. Since time-indexed variable quantities do not appear in (1), we may only establish monotonicity properties with respect to variables in the discretization equation (2). Consider as an example an ODE of the form $x_1' = f_1(x_1, x_2, \lambda) = \lambda x_1 - x_2$ and its corresponding discretization F_1. To establish monotonicity with respect to x_1^i, we consider the partial derivative

$$\frac{\partial F_1}{\partial x_1^i} = -\alpha_i + h\beta_i\lambda,$$

where

$$\alpha_i = \begin{cases} -1 & \text{if } i = 0 \\ 1 & \text{if } i = s \\ 0 & \text{otherwise} \end{cases}.$$

If $\beta_i = 0$, F_1 is trivially monotonic with respect to x_1^i. If $\beta_i \neq 0$, F_1 is non-monotonic with respect to x_1^i if $\frac{\alpha_i}{h\beta_i} \in \left[\, \underline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\lambda}} \,\right]$. Even if this is the case, we may still be able to obtain a useful monotonic property. If we have previously

established, for example, that F_1 is monotonically non-decreasing with respect to λ over our discretization widow, then $\underline{\mathbf{F_1}}$ is explicitly dependent on $\underline{\boldsymbol{\lambda}}$ and

$$\frac{\partial \underline{\mathbf{F_1}}}{\partial x_1^i} = -\alpha_i + h\beta_i \underline{\boldsymbol{\lambda}} < 0,$$

thus establishing monotonicity with respect to x_1^i for the determination of $\underline{\mathbf{F_1}}$. A similar analysis would hold when calculating $\overline{\mathbf{F_1}}$. Establishing monotonicity with respect to as many interval-valued quantities as possible results in a sharper bound on the interval extension. It also reduces computational expense when calculating the maximum and minimum values of the vector field over a given discretization time window.

3.3 Inconsistency Test

The core of the parameter range reduction algorithm is the following test. For a given vector field equation $f_j, 1 \leq j \leq m$ and corresponding discretization F_j over some discretization window $[t^0, t^s]$, we can calculate bounds for the enclosure $[\underline{\mathbf{F_j}}(\mathbf{x}, \boldsymbol{\lambda}), \overline{\mathbf{F_j}}(\mathbf{x}, \boldsymbol{\lambda})]$. If this interval does not intersect $[-E, E]$, where

$$E = |F(x(t^0, \tau, \xi, \lambda), \ldots, x(t^s, \tau, \xi, \lambda), \lambda)|$$

is the magnitude of the local discretization error of (2), then all parameter values $\lambda \in \boldsymbol{\lambda}$ are inconsistent with the data and can be discarded. In practice, it is impossible to determine E, since it depends on the true solution of the ODE, that in turn depends on the true parameter values. In [14], the authors concluded that the error in the data, represented in the discretization equation by the interval-valued variables drawn from the continuous representation, tends to dominate the discretization error. In our implementation, we have allowed the user to specify an approximate value for E if the user has any available information. If the variable ranges are relatively wide, then this value may be set to zero, as we have done in each of our examples below.

3.4 Algorithm Outline

The general outline of the parameter range reduction scheme to reduce a single q-dimensional box in parameter space is described below.

1: **while** progress on reducing box is being made **do**
2: **for** each set of discretization parameters t^0, h, s **do**
3: **for** each model equation $x_j' = f_j(t, x, \lambda),\ j = 1, \ldots, m$ **do**
4: compute $\underline{\mathbf{F_j}}(\mathbf{X}, \boldsymbol{\lambda})$
5: **if** $\underline{\mathbf{F_j}}(\mathbf{X}, \overline{\boldsymbol{\lambda}}) > E$ **then**
6: parameter box is inconsistent **return**
7: **else**
8: **for** each parameter $\lambda_k,\ k = 1, \ldots, q$ **do**

9: **if** $\underline{\mathbf{F_j}}(\mathbf{X}, \boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}_k=[\underline{\boldsymbol{\lambda}_\mathbf{k}}, \underline{\boldsymbol{\lambda}_\mathbf{k}}]} > E$ **then**
10: determine max. λ'_k such that $\underline{\mathbf{F_j}}(\mathbf{X}, \boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}_k=[\underline{\boldsymbol{\lambda}_\mathbf{k}}, \lambda'_k]} > E$
11: update $\boldsymbol{\lambda}_k = [\lambda'_k, \overline{\boldsymbol{\lambda}_\mathbf{k}}]$
12: **end if**
13: **if** $\underline{\mathbf{F_j}}(\mathbf{X}, \boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}_k=[\overline{\boldsymbol{\lambda}_\mathbf{k}}, \overline{\boldsymbol{\lambda}_\mathbf{k}}]} > E$ **then**
14: determine min. λ'_k such that $\underline{\mathbf{F_j}}(\mathbf{X}, \boldsymbol{\lambda})\Big|_{\boldsymbol{\lambda}_k=[\lambda'_k, \overline{\boldsymbol{\lambda}_\mathbf{k}}]} > E$
15: update $\boldsymbol{\lambda}_k = [\underline{\boldsymbol{\lambda}_\mathbf{k}}, \lambda'_k]$
16: **end if**
17: **end for** (each parameter)
18: **end if** (inconsistency test)
19: repeat steps 4 to 18 replacing inequality with $\overline{\mathbf{F_j}} < -E$
20: **end for** (each equation)
21: **end for** (each discretization window)
22: **end while** (progress)

If F_j is monotonically non-decreasing (or non-increasing) with respect to λ_k, then we need only complete lines 9-12 (or 13-16), since the value of $\underline{\mathbf{F}}$ will have been attained at $\underline{\boldsymbol{\lambda}_\mathbf{k}}$ (or $\overline{\boldsymbol{\lambda}_\mathbf{k}}$). The search performed in steps 10 and 14 uses an iterative secant method combined with a bisection method. We do not find the actual maximum or minimum, but rather an approximation within some tolerance. It is at this step of the algorithm that obtaining sharpness of the interval extension is most beneficial.

While looping through discretization parameters (line 2), the algorithm typically fixes s and h and changes only the time window $[t^0, t^s]$. The implementation of the program does, however, allow the user to specify a range of s and h values from which the algorithm can choose.

The procedure above describes how to reduce a single parameter box. Once the algorithm has determined that no further progress can be made, it splits the current parameter box along its widest edge and repeats the above procedure on each of the smaller boxes. A cap on the total number of boxes is provided by the user. The algorithm outputs all consistent boxes as well as the convex hull and centre of mass of these boxes. These last two values can be used to inform the initial guesses for traditional parameter estimation methods.

4 Results

To demonstrate the effectiveness of the algorithm, we first consider a two-dimensional nonlinear pendulum system with three parameters. The data were simulated by numerically integrating with a true parameter set, obtaining a dense set of sample data points and adding a low level of noise. We then consider a four-dimensional pharmacokinetic model with seven parameters. The

data for this model are sparse and were obtained from a real-world experiment. The true parameter values for this second model are unknown.

4.1 Nonlinear pendulum

If a nonlinear pendulum of length L and mass m, with damping coefficient a is exposed to a sinusoidal force of magnitude b at frequency ω, its motion can be modelled by

$$x' = y, \tag{3}$$

$$y' = -\frac{g}{L}\sin(x) - \frac{a}{m}y + \frac{b}{mL}\sin(\omega t). \tag{4}$$

It is assumed that $b = 1.4$, $\omega = 2$ and the gravitational constant $g = 9.8$ are known exactly. The problem is to determine the parameters L, m, a from time series data. We chose true parameters $L = 0.5, m = 0.6, a = 0.05$ and the system was integrated for 100 seconds with an initial condition vector $[3, 0]^T$. Data were sampled at intervals of 0.1 seconds. We added to the sample data normally distributed noise with a standard deviation equal to one-percent of the maximum amplitude of each variable. Continuous bands for the data were generated by the algorithm described in [10] and are plotted in Figure 1.

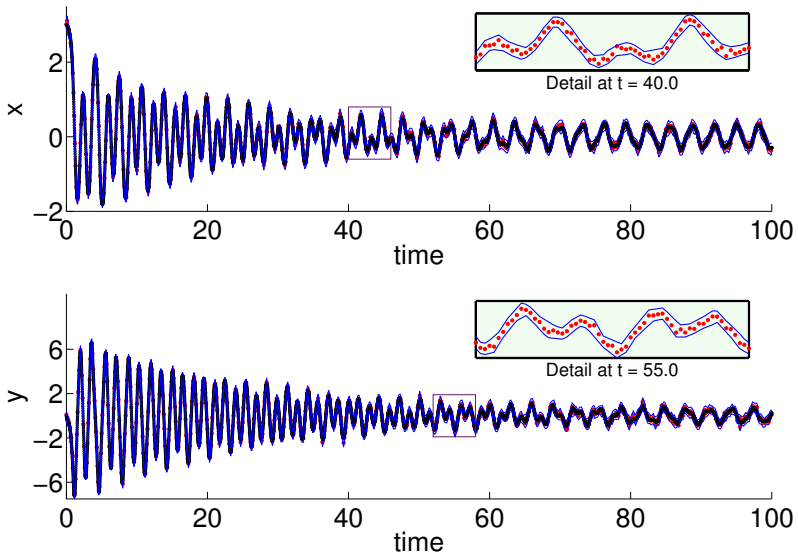

Fig. 1: Simulated data for the nonlinear pendulum and continuous bands generated by the algorithm described in [10] using $t_{\min} = 0.18$ and min height $= 0.15$ for x and using using $t_{\min} = 0.18$ and min height $= 0.40$ for y.

Table 1: Results for the nonlinear pendulum. The algorithm was tested for all values of s between 2 and 17 and for values of h between 0.05 and 1.00 in increments of 0.05. For each (s, h)-pair, results were averaged over 10 runs. The result with the lowest ratio of output hull volume to original volume in parameter space is presented below.

	L	m	a	Max boxes
Original	$[0.005, 50]$	$[0.006, 60]$	$[0.0005, 5]$	-
Best Hull ([14])	$[0.426, 0.623]$	$[0.274, 1.84]$	$[0.0005, 1.26]$	10000
Best Hull (transformed)	$[0.466, 0.529]$	$[0.452, 0.843]$	$[0.0005, 0.309]$	50
Best Hull (original)	$[0.467, 0.529]$	$[0.445, 0.806]$	$[0.0005, 0.292]$	50
Centre of Mass (original)	0.498	0.602	0.118	-

The natural interval extension of the discretization of equation (4) is

$$\mathbf{F_2} = \mathbf{y^0} - \mathbf{y^s} + h \sum_{i=0}^{s} \beta_i \left(-\frac{g}{\mathbf{L}} \sin(\mathbf{x^i}) - \frac{\mathbf{a}}{\mathbf{m}} \mathbf{y^i} + \frac{b}{\mathbf{mL}} \sin(\omega t^i) \right).$$

This expression contains multiple instances of the interval-valued quantities $\mathbf{L}, \mathbf{m}, \mathbf{a}, \mathbf{y^0}$, and $\mathbf{y^s}$, so the output of this function is unlikely to be tight. If instead we rewrite the interval-valued discretization equation as

$$\mathbf{F_2} = \mathbf{y^0} - \mathbf{y^s} - \frac{h}{\mathbf{L}} \left(g \sum_{i=0}^{s} \beta_i \sin(\mathbf{x^i}) - \frac{1}{\mathbf{m}} \sum_{i=0}^{s} \beta_i \sin(\omega t^i) \right) - \frac{h\mathbf{a}}{\mathbf{m}} \sum_{i=1}^{s} \beta_i \mathbf{y^i}.$$

then the only interval-valued quantities that are repeated are $\mathbf{y^0}$, $\mathbf{y^s}$ and $\mathbf{m}$. It can be seen, however, that since a and m are always positive, F_2 is monotonically decreasing with respect to y^s for all parameter and variable values, so the only impediments to sharpness are $\mathbf{y^0}$ and $\mathbf{m}$.

In [14,13], the authors defined new parameters $A = \frac{1}{L}, B = \frac{a}{m}, C = \frac{1}{mL}$, that transformed equation (4) to

$$y' = -gA \sin(x) - By + Cb \sin(\omega t). \tag{5}$$

This transformed model had significantly better monotonicity properties with respect to A, B, C than the original model had with respect to L, m, a. When the algorithm was allowed only to process discretization windows on which the discretization equation was monotonic with respect to all parameters, these improved monotonicity properties were advantageous. This improved monotonicity, however, comes at a cost, since the initial transformed parameter ranges must be inflated to enclose the original parameter values. Once a final set of valid parameter boxes is obtained, they are inflated again to transform back to the original parameters using the inverse transformation $L = \frac{1}{A}, m = \frac{A}{C}, a = \frac{AB}{C}$. Since the new algorithm can process discretization windows on which F is not strictly monotonic, this transformation does not provide as great an advantage.

Table 1 presents results obtained using the updated algorithm. It can be seen that the new algorithm presented in this paper resulted in a significantly better parameter range reduction than obtained using the algorithm in [14]. This improvement also uses far fewer boxes and thus less computational time. The original model performs as well or slightly outperforms the transformed model in all cases except the lower bound of **m**. This suggests that while the transformation was not necessary in this case, it is a potentially useful strategy that should not be discarded.

The best results in Table 1 were obtained after 1.57 seconds for the original model and 2.30 seconds for the transformed model. Simulations were run on a 2.4 GHz Intel Core i5 processor.

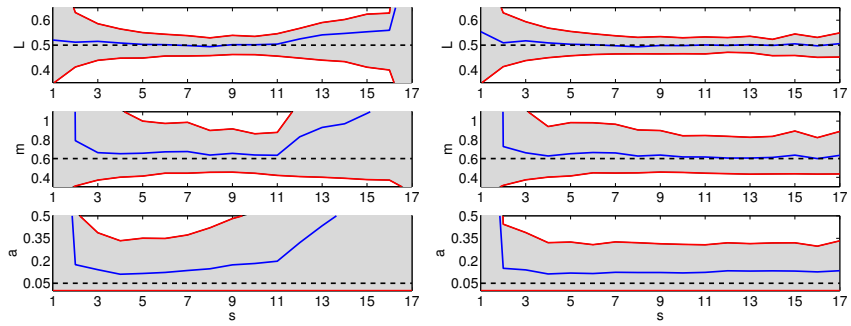

Fig. 2: Hull of consistent boxes and centre of mass versus s for step size $h = 0.05$. In each plot, the upper and lower solid curves represent the upper and lower values of the hull of consistent boxes. The middle solid curve represents the centre of mass of all consistent boxes. The dashed horizontal line represents the true parameter value. Plots on the left are results from when only monotonic windows are allowed, while plots on the right represent results from the improved scheme.

Another advantage to the updated algorithm is illustrated in Figure 2. The plots show the hull of all consistent boxes and the centre of mass of all consistent boxes for each model parameter for the original model for a fixed $h = 0.05$ and a range of s values. The plots on the left show the results when only monotonic windows are permitted. It can be seen that significant reductions in the ranges of m and a occur for narrow (and parameter-specific) ranges of s-values. The plots on the right show that for wide regions of s-values, consistently good results are obtained and that the centre of mass is a good initial guess for a traditional parameter estimation algorithm. This result also occurs when s is fixed and h is allowed to vary over a range of reasonable values. We conclude that the reduction scheme in this paper is less sensitive to the choices of s and h. This is a nice property since the user must select

Table 2: Comparison of the effect of parameter range reduction on input to traditional optimization methods. The original parameter ranges and centres of mass, and the reduced parameter ranges and centres of mass, were each input into five traditional optimization methods. The five methods were the MATLAB functions (A) `fminsearch`, (B) `GlobalSearch`, (C) `MultiStart`, (D) `patternsearch`, (E) `sumulannealbnd`. The final value of the least squares cost function and the total run time are also presented. Time measurements are in clock seconds on a 2.4 GHz Intel Core i5 processor.

		Initial		Optimization Method				
		Range	Start	A	B	C	D	E
Original	L	$[0.005, 50]$	25.0025	1.85	0.57	0.51	5.86	0.54
	m	$[0.006, 60]$	30.003	48.84	3.31	1.63	4.95	6.79
	a	$[0.0005, 5]$	2.50025	25.65	0.82	3.82	5.00	1.33
	Cost	-	-	3318	1370	2773	3489	1258
	time	-	-	117	1927	1080	116	2807
Reduced	L	$[0.467, 0.529]$	0.498	0.50	0.50	0.50	0.49	0.50
	m	$[0.445, 0.806]$	0.60	0.65	0.62	0.61	0.56	0.65
	a	$[0.0005, 0.292]$	0.118	0.058	0.053	0.054	0.046	0.057
	Cost	-	-	30	27	38	32	28
	time	-	-	45	1229	880	113	1121

an s and h value (or range of values) with limited information to guide their choice.

In addition to the improvements made in this paper to the parameter range reduction algorithm, we also switched from using the banding algorithm in [11] to the banding algorithm in [10]. To differentiate between the improved results obtained by the modifications to the parameter range reduction algorithm and those obtained by changing the banding algorithm, we also tested the algorithm in this paper with the prior banding algorithm. We obtained the best reductions $\mathbf{L} = [0.432, 0.559], \mathbf{m} = [0.381, 1.23]$ and $\mathbf{a} = [0.0005, 0.7383]$. Comparing this to the results in Table 1, we can see that while significant improvements can be obtained by using better data bands, this accounts only for a portion of the range reduction improvements.

As a measure of the usefulness of the reduced parameter ranges and centre of mass approximations, we conducted traditional parameter estimation searches on this problem and data. We gave as input the a priori ranges and midpoints to each of 5 MATLAB optimization routines: `fminsearch`, `GlobalSearch`, `MultiStart`, `patternsearch` and `sumulannealbnd`. We then gave as input the reduced parameter hull as bounds and centre of mass as an initial guess to the same optimization routines. In all cases, we used a least squares cost function and all default function settings. The results are presented in Table 2. In each case, the optimization routine was unable to locate a minimum close to the true parameter values, when given as input the initial a priori ranges. When given the reduced ranges, all five algorithms were able to locate a reasonable minimum and in some cases, was able to do so in less time.

We note that this was not intended as a comparison between optimization methods since we used only default settings.

4.2 Pharmacokinetic model

The following model and corresponding data were taken from [5, pg. 152] and originally attributed to Nicholas Holford of the Department of Pharmacology and Clinical Pharmacology at the University of Auckland as a challenge to mathematical modellers. Assume that a patient is orally given $D_{\text{total}} = 48.15$ milligrams of a drug and the drug is absorbed into the blood stream at a constant rate for the first E_T hours. Therefore, the amount of drug that has entered the system after t hours is given by

$$I(t) = \begin{cases} \frac{D_{\text{total}} t}{E_T} & \text{if } 0 \le t \le E_T, \\ D_{\text{total}} & \text{if } E_T \le t. \end{cases}$$

Blood concentrations of the drug, D_b, and of its only metabolite, M_b, are measured (mg/l) at various times. The drug is transported between the blood and the tissue and both the drug and metabolite leave the blood to the urine, where cumulative amounts, D_u and M_u respectively, are measured (mg), again at various times. The mathematical model presented in [5] depends also on the concentration of drug in the tissue. This quantity is unmeasured, but in [14], the authors showed that this dependence can be removed. This reduces the system to the following four equations

$$\begin{aligned} D_b' &= \frac{I'(t) - (k_1 + \mu_D + k_M)D_b + \kappa(I(t) - D_u - M_u)}{V_b} - \kappa(D_b + M_b), \\ M_b' &= \frac{k_M D_b - \mu_M M_b}{V_b}, \\ D_u' &= \mu_D D_b, \\ M_u' &= \mu_M M_b, \end{aligned}$$

where the model parameters are the rate constants k_1 and k_2 for the transport of drug between the blood and tissue, the rate constant k_M for metabolism of drug in the blood stream, the constants μ_D and μ_M that determining the rate of excretion of drug and metabolism respectively from the blood to the urine and V_b, V_t, the effective volume of blood and tissue. The parameter $\kappa = \frac{k_2}{V_t}$ results from the model reduction. The seventh parameter is E_T, appearing in both the expressions for $I(t)$ and $I'(t)$.

Data for this model were obtained from [5, pg. 153] and consist of 15 recordings for each of the variables D_b and M_b taken in non-uniform time intervals from $t = 0.82$ to $t = 24.57$ hours and 11 recordings for each of the variables D_u and M_u taken at times distributed between $t = 1.00$ and $t = 72.00$ hours. The data and the continuous bands used in the algorithm are plotted in Figure 3. This model was presented as a modelling challenge due to

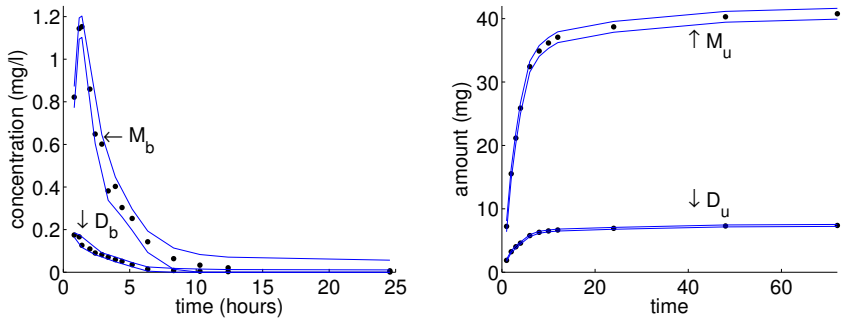

Fig. 3: Pharmacological data and continuous representations. The bands for all four data sets were obtained by the banding algorithm presented in [10]. The bands for D_b and M_b were obtained using $t_{\min} = 1.0$ and minimum heights of 0.02 and 0.1 respectively, while the bands for D_u and M_u were obtained using $t_{\min} = 4.0$ and minimum heights of 0.3 and 1.7 respectively.

Table 3: Results for the pharmacokinetic model. The algorithm was tested for all values of s between 2 and 17 and for values of h between 0.06 and 1.00 in increments of 0.02. For each (s, h)-pair, results were averaged over 10 runs. The result with the lowest ratio of output hull volume to original volume in parameter space is presented below. Runs terminated when an upper limit of 100 boxes was reached.

	V_b	k_1	κ	k_M	μ_D	μ_M	E_T
Original	$[3.0, 7.0]$	$[0.01, 500]$	$[0.01, 100]$	$[0.01, 150]$	$[0.01, 50]$	$[0.01, 50]$	$[0.1, 10]$
Best Hull	$[3.0, 7.0]$	$[143, 321]$	$[0.85, 1.54]$	$[33, 86]$	$[6.4, 13.5]$	$[6.2, 9.9]$	$[0.1, 2.84]$
C. of M.	5.3	238	1.17	62	9.8	8.1	1.35

the disparate scales, sparse non-uniform data and lack of a priori knowledge of reasonable parameter values.

Results from the parameter range reduction algorithm are presented in Table 3. The original ranges were drawn from bounds presented in [5]. It can be seen that we obtained excellent reductions for most of the parameter ranges. No progress, however, was made reducing the range of V_b. A cap on the maximum number of boxes was set at 100. Increasing this value tended not to result in an improved hull of consistent boxes, but rather resulted in a sharpened centre of mass approximation. The best result with 100 boxes was obtained in 12.3 seconds on a 2.4 GHz Intel Core i5 processor.

In [5], the authors analyzed the equations to provide biologically reasonable initial parameter selections for their estimation routine. They obtained weighted least squares cost values of 305.4944 and 44.91649 for two different initial selections of parameter values. In Table 4, we present results obtained from various traditional optimization packages applied to this pharmacokinetic

Table 4: Comparison of the effect of parameter range reduction on traditional optimization methods. The original parameter ranges and centres of mass, and the reduced parameter ranges and centres of mass, were each inputted into 5 traditional optimization methods. The five methods were the MATLAB functions (A) `fminsearch`, (B) `GlobalSearch`, (C) `MultiStart`, (D) `patternsearch`, (E) `sumulannealbnd`. The final value of the weighted least squares cost function and the total run time are also presented. Time measurements are in clock seconds on a 2.4 GHz Intel Core i5 processor.

		Initial		Optimization Method				
		Range	Start	A	B	C	D	E
Original	V_b	$[3.0, 7.0]$	5.0	-	3.02	3.00	3.00	3.01
	k_1	$[0.01, 500]$	250.005	-	214.14	78.85	0.068	201.73
	κ	$[0.01, 100]$	50.005	-	100.00	100.00	99.78	90.78
	k_M	$[0.01, 150]$	75.005	-	136.17	150.00	143.69	144.49
	μ_D	$[0.01, 50]$	25.005	-	25.79	28.50	27.28	26.52
	μ_M	$[0.01, 50]$	25.005	-	32.10	32.93	31.01	38.67
	E_T	$[0.1, 10]$	5.05	-	6.38	6.44	6.42	6.48
	Cost	-	-	failed	26.3	26.0	25.9	26.5
	time	-	-	n/a	1630	700	537	642
Reduced	V_b	$[3.0, 7.0]$	5.3	2.81	4.98	3.25	3.57	3.99
	k_1	$[143, 321]$	238	240.96	308.23	265.29	246.37	221.18
	κ	$[0.85, 1.54]$	1.17	1.23	1.08	1.27	1.05	1.12
	k_M	$[33, 86]$	62	50.88	77.23	54.59	61.91	52.56
	μ_D	$[6.4, 13.5]$	9.8	9.13	13.50	9.75	11.03	9.55
	μ_M	$[6.2, 9.9]$	8.1	9.75	9.90	9.57	9.86	9.69
	E_T	$[0.01, 2.84]$	1.35	1.12	0.96	1.03	1.14	1.05
	Cost	-	-	0.80	1.03	0.84	0.85	0.97
	time	-	-	82	2011	418	288	367

model. Using the midpoints of the initial parameter ranges, we found that the MATLAB local optimization function `fminsearch` was unable to converge due to excessive stiffness of the model given the choice of parameter values. The four global optimization functions fared slightly better, but still were unable to obtain reasonable parameter values. In contrast, the global optimization functions were able to converge to parameter sets with significantly smaller cost function values when given the reduced ranges as input. The functions `MultiStart` and `patternsearch` converged to a similar cost function value but their output parameter sets differed in a number of values. The local optimizer `fminsearch` (that does not accept as input variable bounds) converged to a point outside of the a priori initial ranges. It would appear that the system is over-parameterized. In Figure 4, we plot the projections of all consistent boxes onto the $k_M - \mu_M$ plane. It is clear that there is some dependency among the parameters that should be resolved before proceeding with a parameter estimation scheme.

5 Conclusion

This paper considered the problem of identifying parameters in an ODE model given time series data. The algorithm presented reduces a priori ranges of parameter values and outputs tighter bounds and a starting value for traditional parameter estimation schemes. The algorithm is shown to be an improvement to a previously reported version of the algorithm. The algorithm outputs reduced parameter ranges and centre of mass approximations that are shown to be effective inputs to traditional parameter estimation schemes.

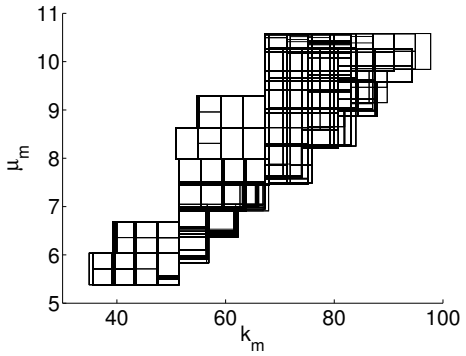

Fig. 4: Projection of consistent parameter boxes. The parameter reduction scheme was run with $s = 4, h = 0.15$ and with an upper limit of 2000 boxes.

The algorithm requires all model variables to be measured. In many cases, this is not a reasonable assumption. For example, it may not be reasonable to assume that time series data will be available for the velocity of the nonlinear pendulum. In the case of the pharmacokinetic model, it was possible to rewrite the differential equations to remove an unmeasured quantity, but this is not always possible. By considering the instances of unmeasured variables in the discretization equation to be additional parameters that can be reduced, it is possible to obtain good parameter ranges in the presence of partial data sets. Strong results in this line of research have been found and their details will be reported elsewhere.

6 Acknowledgements

This work was supported by an Ontario Graduate Scholarship, a Natural Sciences and Engineering Council of Canada (NSERC) Postgraduate Scholarship and a NSERC Discovery Grant.

References

1. Bard, Y.: Comparison of gradient methods for the solution of nonlinear parameter estimation problems. SIAM J. Numer. Anal. **7**, 157–186 (1970)
2. Gonzalez, O., Kuper, C., Jung, K., Naval, P., Mendoza, E.: Parameter estimation using simulated annealing for s-system models of biochemical networks. Bioinformatics. **23**(4), 480–486 (2007)
3. Guus, C., Boender, E., Romeijn, H.: Stochastic methods. In: R. Horst, P. Pardalos (eds.) Handbook of global optimization, pp. 829–869. Kluwer Academic Publishers, Dordrecht, The Netherlands (1995)
4. Hansen, E., Walster, G.: Global optimization using interval analysis, 2nd edition. Pure and Applied Mathematics. M. Dekker, New York (2004)
5. Knott, G., Kerner, D.: MLAB Applications Manual. Civilized Software Inc, Bethesda, MD (2004). URL http://www.civilized.com/files/appman.pdf
6. Moles, C., Mendes, P., Banga, J.: Parameter estimation in biochemical pathways: A comparison of global optimization methods. Genome Research. **13**, 2467–2474 (2003)
7. Moore, R.: Interval Analysis. Prentice-Hall, Englewood Cliffs, New Jersey (1966)
8. Moore, R.: Methods and Applications of Interval Analysis. Studies in Applied Mathematics. SIAM, Philadelphia (1979)
9. Ratschek, H., Rokne, J.: Computer Methods for the Range of Functions. Mathematics and Its Applications. Ellis Horwood Limited, New York (1984)
10. Skelton, A., Willms, A.: An algorithm for continuous piecewise linear bounding of discrete time series data. submitted (2013)
11. Szusz, E., Willms, A.: A linear time algorithm for near minimax continuous piecewise linear representations of discrete data. SIAM J. Sci. Comput. **32**, 2584–2602 (2010). DOI http://dx.doi.org/10.1137/090769077. URL http://dx.doi.org/10.1137/090769077
12. Ugray, Z.: Scatter search and local nlp solvers: A multistart framework for global optimization. INFORMS Journal on Computing **19**(3), 328–340 (2007)
13. Willms, A.: Parameter range reduction for ode models using cumulative backward differentiation formulas. Journal of Computational and Applied Mathematics **203**, 87–102 (2007)
14. Willms, A., Szusz, E.: Parameter range reduction for ODE models using monotonic discretizations. Computational and Applied Mathematics **247**, 124151 (2013)