Robust Parameter Estimation for Rational Ordinary Differential Equations

Oren Bassik[a], Yosef Berman[a], Soo Go[b], Hoon Hong[c], Ilia Ilmer[b], Alexey Ovchinnikov[a,b,d], Chris Rackauckas[e], Pedro Soto[f], Chee Yap[g]

[a]*Ph.D. Program in Mathematics, CUNY Graduate Center, New York, NY, USA*
[b]*Ph.D. Program in Computer Science, CUNY Graduate Center, New York, NY, USA*
[c]*Department of Mathematics, North Carolina State University, Raleigh, NC, USA*
[d]*Department of Mathematics, CUNY Queens College, Queens, NY, USA*
[e]*Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA, USA*
[f]*Mathematical Institute, University of Oxford, Oxford, UK*
[g]*Department of Computer Science, Courant Institute of Mathematical Science, New York University, New York, NY, USA*

Abstract

We present a new approach for estimating parameters in rational ODE models from given (measured) time series data. In typical existing approaches, an initial guess for the parameter values is made from a given search interval. Then, in a loop, the corresponding outputs are computed by solving the ODE numerically, followed by computing the error from the given time series data. If the error is small, the loop terminates and the parameter values are returned. Otherwise, heuristics/theories are used to possibly improve the guess and continue the loop. These approaches tend to be non-robust in the sense that their accuracy often depends on the search interval and the true parameter values; furthermore, they cannot handle cases where the parameters are only locally identifiable.

In this paper, we propose a new approach, which does not suffer from the above non-robustness. In particular, it does not require making good initial guesses for the parameter values or specifying search intervals. Instead, it uses differential algebra, rational function interpolation of the data, and multivariate polynomial system solving. We also compare the performance of the resulting software with several other estimation software packages.

Keywords: Parametric ODE Models, Parameter Estimation, Differential Algebra, Symbolic-Numeric Differentiation, Mathematical Software

1. Introduction

1.1. Overall problem

Parametric ODEs are ubiquitous in science and engineering. In order to use them, one first needs to estimate the parameters, which is usually done using measured time series data. Hence, the overall problem is: given a parametric ODE and some measured data, to estimate the parameters.

1.2. State of the art

Due to its importance, this problem has been the subject of intensive research efforts that have yielded various theories and approaches. They can be roughly categorized into three approaches: *shooting*, *two-stage*, and *algebraic*.

Shooting approaches (also known as simulation-based approaches) directly use the ODE solver to compute the loss function of the current estimate. This is used to define a nonlinear optimization to improve the parameter estimates. This popular approach has seen many software implementations; to list a few: AMIGO2, COPASI, Data2Dynamics, SBtoolbox2/IQM; see [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. Many studies have further developed the shooting approaches to improve robustness, adding techniques such as multiple shooting [16, 17, 18], automatic differentiation

Email addresses: orebas@yahoo.com (Oren Bassik), yberman1@gradcenter.cuny.edu (Yosef Berman), sgo@gradcenter.cuny.edu (Soo Go), hong@ncsu.edu (Hoon Hong), iilmer@gradcenter.cuny.edu (Ilia Ilmer), aovchinnikov@qc.cuny.edu (Alexey Ovchinnikov), crackauc@mit.edu (Chris Rackauckas), pedro.soto@maths.ox.ac.uk (Pedro Soto), yap@cs.nyu.edu (Chee Yap)

and adjoint techniques for more accurate derivatives [19, 20, 21], and landscape smoothing techniques like the prediction-error method [22], all aimed at helping optimization techniques avoid local minima. In many cases, shooting methods include an intermediate step of training a surrogate model (or metamodel or emulator) of the simulation or loss function, typically using methods like radial basis functions [23, 24] or neural network machine learning models [25, 26, 27], in order to replace the requirement of simulating the model at each new set of parameters. Whether using a surrogate or not, this class of techniques define the core loss function via a simulation process and require some amount of repeated solving of the equations for which the parameters are being estimated. These challenges, particularly the dependence on initial guesses and the risk of converging to local minima, are most pronounced in commonly used local optimization algorithms.

Global optimization strategies represent another important class of methods designed specifically to overcome the limitations of local optimizers, particularly the risk of converging to suboptimal local minima. This diverse family of techniques [28] includes multi-start methods (often combined with local searches from various starting points), stochastic methods like simulated annealing [29], and evolutionary approaches such as genetic algorithms [30, 31]. Their primary advantage is a more exhaustive exploration of the parameter space, significantly increasing the chances of locating the basin of attraction for the true global minimum. However, this broader search typically incurs a substantial computational burden, often demanding a significantly greater number of simulations. Furthermore, effective use still requires careful user configuration, including the specification of search bounds (which can still impact performance) and the tuning of various algorithm-specific hyperparameters (e.g., population size, mutation rates, termination criteria).

Critically, standard global optimization techniques do not attempt to address the issue of completeness of the set of solutions, for systems where parameters are only locally and not globally identifiable. They typically lack mechanisms to ascertain the total number of such distinct solutions beforehand, focusing instead on converging to one potentially non-unique minimum. Indeed, they are typically not aware that there may be multiple equally good parameter sets that yield optimal or near-optimal fits. This is in stark contrast with approaches which attempt from the outset to fully characterize the identifiable parameter space.

Two-stage approaches (also known as collocation, smoothing, or principle differential analysis) in contrast avoid numerically solving the ODE by fitting collocation polynomials to the data to obtain predictions of the data which are then directly fit against the data [32, 33]. As the result, these approaches are generally much more efficient. Previous studies have used various objects such as B-splines [34, 35, 36] and local least-squares [37, 38, 39, 40]. One downside to two-stage approaches is requiring observation of all states. Furthermore, it has been noted that the error introduced through the derivative estimation tends to lead to less accurate final results in real-world scenarios with less dense data [35, 33]. These studies suggest finishing a parameter estimation by using the output of a two-stage approach as the initial condition to a simulation-based approach.

Algebraic approaches tackle rational differential models by exploiting theoretical results from differential algebra. The paper [9] expresses parameters algebraically in terms of derivatives of input and output functions and then estimates these derivatives using the numerical data. Two difficulties with this approach are the size of these algebraic expressions and the computational complexity of deriving them. The papers [41, 42, 43, 44] study how to use input-output equations in parameter estimation and how to lower the orders of derivatives used in the estimation (because higher order derivatives typically introduce larger errors with numerical data) by different approaches, including using integrals, cf. [45, Section 3.2]. Computing input-output equations could be computationally very costly, although recently there have been improvements in algorithms computing them [46]. The paper [47] considers an algebraic approach to parameter estimation in linear systems. Finally, the paper [48] investigates the parameter manifold in the case where the differential model is structurally non-identifiable.

1.3. The proposed approach and novelty

The proposed approach could roughly be classified as a two stage approach. However, it is substantially different from the previous two stage approaches, in that it estimates parameters *robustly*, in the following sense:

1. The proposed approach produces good estimates no matter what the specified search intervals are. In fact, it does not even require specifying any search interval.
2. The proposed approach produces good estimates no matter where the true parameter values are.
3. The proposed approach produces good estimates even when the parameters are not globally identifiable, as long as they are locally identifiable.

It achieves the robustness by judiciously combining:

- differential algebra approach from the parameter identifiability analysis developed in [49, 50],
- estimation of the derivatives of the ODE solutions from rational interpolation of the data,
- overdetermined polynomial system solving by squaring the system and filtering the solution set.

As such, this technique can be used in isolation for dense data, or together with shooting approaches by providing more robust initial conditions for sparse data.

1.4. Software Implementation

We implemented the proposed approach into a Julia package called "ParameterEstimation.jl". All code and benchmarking data are available at https://github.com/iliailmer/ParameterEstimation.jl.

2. Problem Statement

In this section, we state the problem solved by the proposed approach and illustrate it using a toy example.

Input:

1. An ODE model Σ
$$\begin{cases} \boldsymbol{x}'(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\mu}) \\ \boldsymbol{y}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\mu}) \\ \boldsymbol{x}(0) = \boldsymbol{x_0} \end{cases} \tag{1}$$
where we use bold fonts for vector:
   - $\boldsymbol{f}$ and $\boldsymbol{g}$ are vectors of rational functions describing the model states and outputs, respectively,
   - $\boldsymbol{x}$ are state variables,
   - $\boldsymbol{u}$ are input (control) variables,
   - $\boldsymbol{y}$ are output variables,
   - $\boldsymbol{\mu}$ are unknown model parameters and $\boldsymbol{x_0}$ are unknown initial conditions, treated as parameters to be estimated.
2. A data $D = ((t_1, \boldsymbol{y}_1), \ldots, (t_n, \boldsymbol{y}_n))$ where $\boldsymbol{y}_i$ is the measured value of $\boldsymbol{y}$ at time t_i

Output: Estimated values for the parameters $\boldsymbol{\mu}$ and $\boldsymbol{x_0}$.

Input: (*Toy example*)

$$\begin{array}{rcl} \Sigma & : & \left\{ \begin{array}{lcl} x' & = & -\mu x \\ y & = & x^2 + x \\ x(0) & = & x_0 \end{array} \right. \\ D & : & (0.000, 2.000), (0.333, 1.563), (0.666, 1.229), (1.000, 0.974) \end{array}$$

Output: $(\mu, x_0) \approx (0.499, 1.000)$

The estimated values $(\mu, x_0) \approx (0.499, 1.000)$ are accurate, as the data D was generated by numerically solving the model Σ with the true values $\mu = 0.500$ and $x_0 = 1.000$.

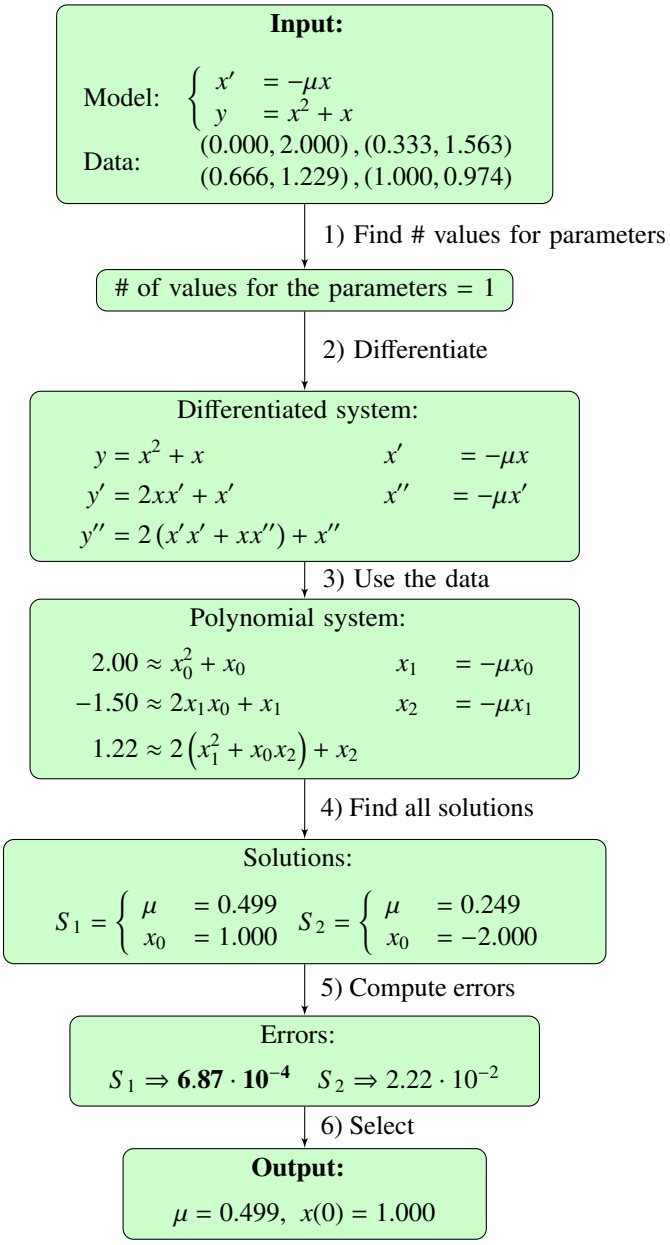

Figure 1: Flowchart of the proposed parameter estimation approach.

3. The Proposed Approach

We will briefly describe the proposed approach by first giving a flow chart of the steps. (See Figure 1)

1. **Determine Parameter Solution Count:** Use the model Σ and techniques from [49, 50] (cf. [51]) to find the number k of distinct parameter value sets that are consistent with the model structure and outputs. For the toy example, analysis shows $k = 1$, meaning we expect a unique solution set for (μ, x_0).
2. **Differentiate the System:** Differentiate the model equations (both dynamics and outputs) sufficiently many times with respect to time t, based on the analysis in Step 1 (see [49, 50] for details). For the toy example, differentiating up to the second order yields:

$$\begin{cases} y &= x^2 + x \\ y' &= 2xx' + x' \\ y'' &= 2\left(x'x' + xx''\right) + x'' \\ x' &= -\mu x \\ x'' &= -\mu x' \end{cases}$$

3. **Derive Numerical Polynomial System:** Use the data D to obtain a numerical polynomial system from the differentiated equations.
   (a) Interpolate Data: Approximate the output functions (here, just $y(t)$) by interpolating the data points in D with a suitable function, typically a rational function $g(t)$. For the toy example data

$$D = \{(0.000, 2.000), (0.333, 1.563), (0.666, 1.229), (1.000, 0.974)\},$$

   rational interpolation might yield

$$y(t) \approx g(t) = \frac{0.58t^2 - 3.11t + 6.82}{t + 3.41}.$$

   (b) Substitute Interpolant: Replace the output variables and their derivatives in the differentiated system (from Step 2) with the interpolating function $g(t)$ and its derivatives. For the toy example, this gives:

$$\begin{cases} g(t) &\approx x^2 + x \\ g'(t) &\approx 2xx' + x' \\ g''(t) &\approx 2\left(x'x' + xx''\right) + x'' \\ x' &= -\mu x \\ x'' &= -\mu x' \end{cases}$$

   where

$$g'(t) = \frac{0.58t^2 + 3.96t - 17.41}{(t + 3.41)^2} \quad \text{and} \quad g''(t) = \frac{48.32}{(t + 3.41)^3}.$$

   (c) Evaluate at $t = 0$: Evaluate the resulting approximate differential system at $t = 0$, using the notation $x_i = x^{(i)}(0)$. For the toy example, evaluating $g(0)$, $g'(0)$, and $g''(0)$ yields the numerical polynomial system:

$$\begin{cases} 2.00 &\approx x_0^2 + x_0, \\ -1.50 &\approx 2x_1x_0 + x_1, \\ 1.22 &\approx 2\left(x_1x_1 + x_0x_2\right) + x_2 \\ x_1 &= -\mu x_0 \\ x_2 &= -\mu x_1 \end{cases}$$

   This system has 5 equations in the 4 unknowns μ, x_0, x_1, x_2.
4. **Solve Polynomial System:** Find all real solutions to the (typically overdetermined) numerical polynomial system derived in Step 3. This often involves squaring the system and using numerical algebraic geometry techniques like homotopy continuation [52]. For the toy example, solving the system yields two real solution sets for the parameters and initial derivatives:

$$\begin{aligned} S_1 : (\mu, x_0, x_1, x_2) &\approx (0.499, \quad 1.000, -0.499, \quad 0.249) \\ S_2 : (\mu, x_0, x_1, x_2) &\approx (0.249, -2.000, \quad 0.498, -0.124) \end{aligned}$$

   Focusing on the parameters (μ, x_0), the solutions are approximately $(0.499, 1.000)$ and $(0.249, -2.000)$.

5. **Compute Solution Errors:** For each distinct parameter solution $(\widehat{\mu}, \widehat{x_0})$ found in Step 4, evaluate its quality by:
   (a) Simulating the original ODE model Σ using these parameter values $(\widehat{\mu}, \widehat{x_0})$.
   (b) Computing the error (e.g., root mean square error) between the simulated output $\widehat{y}(t_i)$ and the measured data y_i at the time points t_i.

   For the toy example's two solutions:

$$\begin{aligned} S_1 : (\mu, x_0) &\approx (0.499, \quad 1.000) \quad \Rightarrow \text{Error } e_1 \approx 6.87 \times 10^{-4} \\ S_2 : (\mu, x_0) &\approx (0.249, -2.000) \quad \Rightarrow \text{Error } e_2 \approx 2.22 \times 10^{-2} \end{aligned}$$

6. **Select Best Solutions:** Select the k parameter solution(s) (where k was determined in Step 1) that yield the smallest errors calculated in Step 5. If available, user-provided parameter range constraints can be applied before this final selection. For the toy example, $k = 1$, and solution S_1 has the significantly smaller error. Thus, the final estimated parameters are $(\mu, x_0) \approx (0.499, 1.000)$.

4. Implementation details

Steps 1) and 2)

The two steps are are done by calling SIAN [49, 50].

Step 3)

We perform interpolation of the dataset D. In this interpolation, we have considerable freedom in choosing the interpolation method, with the following desired properties:

1. easily calculated analytic derivatives, which are as well interpolated as the function itself
2. the ability to fit a wide variety of shapes, including asymptotic behavior, decay to 0, and periodicity,
3. a preference for parsimonious models, and
4. numerical stability.

Our software supports a variety of modern interpolation methods, via `BaryRational.jl` and other interpolation packages. Of note, we support Fourier extension, Floater-Hormann rational interpolation [53], as well as the "AAA" algorithm introduced in [54], which uses a barycentric rational representation adapted to input data. Experimental data show that AAA rational interpolation interpolates smooth functions in such a way that, for a wide variety of data outputs from differential equations, even higher order derivatives are well interpolated. In our software, we use AAA as a default interpolation scheme, but allow iterating over multiple schemes in a parallel fashion to select the best one.

Once interpolation is done, we apply high-order automatic differentiation provided by the `TaylorSeries.jl` [55] and `ForwardDiff.jl` packages to the interpolated function to estimate derivative values of higher order.

We substitute the estimated values of the derivatives of the output y into the system obtained in Step 2). This typically results in an over-determined system of polynomial equations.

Step 4)

Since the obtained polynomial system is typically overdetermined, it must be reduced to a square system before solving. This is achieved by collecting equations one by one, requiring that a newly added equation increases the rank of the Jacobian of the system. If, for a given equation, the rank is unchanged, the equation is not added. Then we find all solutions of the squared system. By default, we are using homotopy-based solving from `HomotopyContinuation.jl`. We also support `MSolve` [56] via the Oscar algebra system in Julia [57, 58]. This method finds all solutions with theoretical guarantees. However, it has been less stable for some of the systems we encountered. All results presented here use homotopy-based solving, the default option.

Steps 5) and 6)

For each obtained solution (parameter estimates), we compute the data by numerically solving the model. Then we determine errors between the computed data and the given (measured) data. We keep the estimates with minimal errors. Such estimates may exist for different interpolation schemes; we keep the ones with minimal error across all schemes.

The selection of the final parameters to display needs a small caveat. In the absence of interpolation or numerical error, steps 1-3 will describe the expected k solutions precisely. Squaring the polynomial system may introduce spurious solutions. These typically have large errors and rank poorly, but this is not guaranteed. Moreover, interpolation errors, noise, and numerical errors may result in the best solution ranking lower than the first k solutions. Therefore, as a point of precision, note that our method always produces the complete set of solutions expected from identifiability theory, but the best solution may not be among the set with minimal error.

Extra features

We would like to emphasize the following extra features:

- The iteration over interpolation schemes and underlying polynomial solving via `HomotopyContinuation.jl` [52] are parallelizable and can take advantage of a multicore (or multithreaded) computing environment.
- `ParameterEstimation.jl` provides identifiability interface to SIAN [49, 50]. One can assess local and global identifiability of the parameters and initial conditions before estimation.

Example Usage

The following Julia code solves the toy example problem from Section 2 using our software.

```
using ParameterEstimation
using ModelingToolkit

# --- Model
@parameters mu
@variables  t x(t) y(t)
D = Differential(t)
@named Sigma = ODESystem([D(x) ~ -mu * x],t,[x],[mu])
outs = [y ~ x^2 + x]

# --- Data
data = Dict(
  "t"      => [0.000, 0.333, 0.666, 1.000],
  x^2 + x => [2.000, 1.563, 1.229, 0.974])

# --- Run
res = estimate(Sigma, outs, data);
```

You should see the following output:

```
Parameter(s)         : mu    = 0.499
Initial Condition(s): x(t) = 1.000
```

Current Limitations

We list a few limitations of the current preliminary implementation:

- As a two-stage approach, the issues common to this class of approaches apply. The data must be time series data which is sufficiently dense for the rational polynomial fitting to give accurate derivative approximations.
- Our software supports rational ODEs models (the right-hand sides are rational functions), while software such as AMIGO2 and IQM also support ODE models that are not necessarily rational.

- We use SIAN that is based on a Monte-Carlo algorithm that returns the bound on number of solutions with high probability, though detailed analysis is still ongoing.
- The high order derivative approximations via rational polynomials may suffer from errors due to noise or rounding. We leave the extension of the technique to usage with robust derivative approximations [36, 59] as a topic of further study.
- **Scalability:** While the current implementation successfully handles the benchmark models presented, including the Crauste model with 18 parameters and initial conditions, within reasonable time on standard hardware (one run of each benchmark system completed in approximately 15 minutes), the computational cost is primarily driven by the polynomial system solving step. The complexity of this step scales significantly with the number of variables and equations. Therefore, applying this method to substantially larger systems may require careful selection and configuration of the underlying polynomial solver (e.g., choosing between homotopy continuation and symbolic methods based on system characteristics) or dedicated computational resources. For a detailed comparison of polynomial solver performance on the systems generated by our approach, see [60].

5. Performance

Median of Relative Errors in %

Software			IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range			[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Models – Size	Harmonic (2)	4	64.2	63.0	65.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Van der Pol (4)	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	FitzHugh-Nagumo (6)	5	68.8	139.2	159.7	0.6	0.2	0.3	0.0	0.0	0.0	0.0
	HIV (8)	15	75.5	89.7	133.8	2.0	7.9	57.8	0.0	0.0	0.0	0.0
	Mammillary 3 (10)	8	74.3	88.1	85.6	5.6	12.8	11.8	0.0	0.0	0.0	0.0
	Lotka-Volterra (12)	5	71.7	76.5	76.5	22.1	55.0	75.6	0.0	0.0	1.0	0.0
	Crauste (14)	18	85.3	109.3	139.8	26.5	67.2	144.8	0.0	77.6	24.5	0.0
	Biohydrogenation (16)	10	81.2	113.4	78.9	53.9	110.6	284.6	0.0	0.0	22.2	0.0
	Mammillary 4 (18)	11	76.0	88.4	105.1	46.9	61.1	97.1	0.0	53.8	51.9	0.0
	SEIR (20)	7	101.7	173.0	258.1	17.1	49.6	105.4	17.9	19.8	30.8	0.0

Mean of Relative Errors in %

Software			IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range			[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Models – size	Harmonic (2)	4	66.3	68.7	101.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Van der Pol (4)	4	7.5	0.0	9.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	FitzHugh-Nagumo (6)	5	82.4	131.1	210.3	8.3	15.9	20.4	0.0	0.0	0.2	0.0
	HIV (8)	15	85.6	104.2	144.7	14.8	36.9	102.9	18.5	43.2	28.4	0.0
	Mammillary 3 (10)	8	76.1	104.3	128.2	11.9	21.5	24.7	0.0	0.0	0.0	0.0
	Lotka-Volterra (12)	5	72.2	75.4	75.4	54.7	76.9	111.7	13.6	112.6	29.7	0.0
	Crauste (14)	18	99.6	125.2	179.7	47.3	72.9	226.4	2.6	105.7	51.7	0.0
	Biohydrogenation (16)	10	93.7	184.0	130.3	77.7	151.1	306.6	17.8	13.5	28.3	0.0
	Mammillary 4 (18)	11	94.1	109.6	105.3	66.2	67.7	118.8	29.8	56.7	59.0	0.1
	SEIR (20)	7	132.3	230.8	338.9	18.4	60.6	118.1	25.9	40.7	50.8	0.0

Table 1: The above two tables show the median and the mean of the Root Mean Square Relative Errors (RMSRE) over 10 randomly generated data sets. For the details on how the data sets are generated and how the errors are computed, see Subsections 5.2, 5.3 and 5.4. The last three models are grouped separately from the rest to indicate that the model has parameters that are only locally identifiable. "Size" means the number of state variables plus the number of parameters.

In this section, we study the performance (in particular robustness) of several software packages on a collection of benchmarks models. (see Appendix A for model equations and details).

5.1. Software Packages

We ran the following software packages:

- IQM [6]

  It uses a single shooting approach with a local derivative-free optimizer (Nelder-Mead, as specified by `symplexIQM`).

- SciML [61, 62]

  It is intentionally a naive single shooting implementation which serves as a baseline for one of the most common parameter estimation approaches. While such a package can easily be improved using methods like multiple shooting and global optimizers, this setup demonstrates the expected results from the standard simple approach. A polyalgorithm mixing ADAM with BFGS is used to demonstrate the results of a common gradient-based optimization scheme.

- AMIGO2 [11]

  It uses a multi-start global optimization approach. This approach can always be tuned to achieve correct results by choosing sufficient bounds/tolerances and population points, though not all of these values are typically known in advance.

- `ParameterEstimation.jl` (this work)

We could not run the following software packages: Data2Dynamics [12], COPASI [8], and IQR[1]/IQDesktop[2] because of technical difficulties in installation.

5.2. Settings used for Software Packages

IQM, SciML and AMIGO2 require search ranges or initial guesses for the parameters and initial conditions or certain additional settings to be specified by the user that pertain specifically to the optimization task performed by each software package, for instance, maximal # of iterations for gradient-based optimization, specific loss/error functions to be optimized.

ParameterEstimation.jl is built on top of existing algorithms for solving polynomial systems and interpolating data points. Their implementations involve some settings.

For studying performance of the software systems, we used the following settings:

- For IQM, we set the maximal number of function evaluations at 200,000 and used a `simplexIQM` solver.

- For SciML, we set the number of function evaluation at 200,000 and set the learning rate at 0.01.

- For AMIGO2, we set the maximal number of function evaluations at 200,000, set the IVP solver integration tolerances at 10^{-13} and set the solver tolerances to 10^{-13} and used a nonlinear least squares solver, which meets or exceeds the settings suggested by Julio Banga[3]. Due to the non-deterministic nature of choosing starting points in the global optimization used by AMIGO2, multiple runs with the same example and settings can produce different estimates (possibly with larger or smaller error). We only ran it once and reported the results.

- For ParameterEstimation.jl, default options were used for most settings. The default polynomial solver is HomotopyContinuation.jl, with a tolerance of 10^{-12}. The default setting is to use a fixed set of interpolation schemes including AAA, Floater-Hormann of orders (3,6,8), and Fourier interpolation (though, for all of the models in this paper, AAA interpolation alone is sufficient.)

[1] https://iqrtools.intiquan.com/

[2] https://iqdesktop.intiquan.com/book/

[3] Private communication, May-June 2023

5.3. Test Data Generation

For all models and all software packages, the test data were generated uniformly as follows. We chose 21 time points equally spaced over $[-0.5, 0.5]$ and also chose the reference values of the parameters and initial conditions randomly from the interval $[0.1, 0.9]$ using the random.uniform function in Python's numpy package. For reproducibility, we used random.seed(0) at the start of choosing the 10 data sets for each system. In case of errors (e.g., due to singularity), the set was discarded and a new set of random values was generated instead. These uniform random choices were made for the sake of studying/comparing the robustness of the approaches in isolation, independent of the physical systems underlying the models. The random values were at least 0.1 chosen away from 0.0 and 1.0 to avoid potential degeneracy.

For each model and for each software package, the input script (containing the model description, the generated data and settings) was generated automatically.

5.4. Estimation Error

To evaluate the quality of the estimation, for each data set, we computed the *Root Mean Square Relative Error* defined as

$$RMSRE = \sqrt{\frac{1}{\#P}\sum_{p\in P}\left(\frac{p_{true} - p_{estimated}}{p_{true}}\right)^2} \times 100,$$

where P denotes the identifiable[4] set of the system parameters and the initial values of the state variables in the input system. The appendix provides RMSRE for each model, search interval, and random data set. Table 1 summarizes them using median and mean.

5.5. Discussion

The inspection of the test results in Table 1 and Appendix shows that the proposed approach outperforms the others with regard to estimation error in all but one instance. Furthermore, the proposed approach is robust in the following aspects:

Search Interval: See Table 1. Note that IQM, SciML and AMIGO2 (based on shooting approaches) require the user to specify a search interval. Note also that their relative errors are *dependent* on search interval, sometimes even *non-monotonically*. For an example, consider the relative error of AMIGO2 on the Crauste Model. If the upper bound of the search interval is increased from 1.0 to 2.0, the median (mean) of the relative error is increased from 0.0% (2.6%) to 77.6% (105.7%). However, if the upper bound is increased to 3.0, the median (mean) of the relative error is decreased to 24.5% (51.7%). See Random Data 3, 5, 9 and 10 in the Crauste table of Appendix for specific instances of this behavior. This non-monotonic behavior could add difficulty to the users in specifying a suitable search interval.

The proposed approach's relative error is, by design, *independent* of search interval, because it does not require/use search interval.

Parameter values: See the tables in Appendix. Note that the relative errors of IQM, SciML and AMIGO2 are *dependent* on the values of the parameters (randomly sampled from [0.1, 0.9]), sometime drastically. For an example, see the Crauste table. AMIGO2 has essentially no error for some random values of parameters but huge error for the others.

The proposed approach's relative error is practically *independent* of the values of the parameters.

Multiple solutions: See Table 1. The first seven models are globally identifiable (hence with unique solution for the parameters). The last three models contain parameters that are only locally identifiable (hence with multiple solutions for the parameter values). Note that the relative errors of IQM, SciML and AMIGO2 are significantly larger for locally identifiable models. It is because the optimization-based approaches may miss an intended value of a parameter.

The proposed approach's relative error is practically *independent* of identifiability, as long as it is at least locally identifiable. In fact, it finds *all* the solutions, making it easier for the users to quickly understand the full landscape of solutions.

[4]For the Biohydrogenation model, we excluded the errors from the estimates for the initial value of x_7, which is not identifiable.

As noted, the primary computational bottleneck lies in solving the derived polynomial system. A crucial direction for future research is to investigate the scalability of this approach by integrating it with high-performance polynomial system solvers, potentially leveraging distributed or GPU-accelerated algorithms, to tackle significantly larger and more complex biological or engineering models.

6. Conclusion

Parameter estimation for ordinary differential equation (ODE) models is a fundamental task in systems biology and engineering. This paper introduced a novel approach specifically designed for rational ODEs, which aims to overcome the robustness issues often encountered with traditional methods. Our technique uniquely combines differential algebra for structural analysis, rational function interpolation for approximating output derivatives from time series data, and numerical polynomial system solving to compute parameter candidates.

The primary contribution of this work is a robust parameter estimation method demonstrated through benchmarks against established software. Our approach exhibits independence from user-specified search intervals and initial parameter guesses, unlike typical local optimization-based methods whose performance can be sensitive to these choices. Furthermore, it successfully identifies parameter values even when they are only locally identifiable, finding all possible solutions consistent with the data and model structure, whereas shooting-based methods may converge to a single, possibly incorrect, local minimum. These features make it particularly useful for exploring the parameter landscape and providing reliable estimates or initial guesses for other methods.

Despite its advantages, the proposed approach has limitations that suggest avenues for future research. Currently, it requires relatively dense time series data for accurate rational interpolation and derivative estimation, and it is restricted to models with rational function dynamics. The sensitivity to noise in the data, particularly how it affects higher-order derivative approximations, warrants further investigation, potentially incorporating robust differentiation techniques. As discussed, the main computational cost lies in solving the polynomial system; thus, future work should focus on rigorous scalability analysis, exploring integrations with high-performance solvers to handle larger systems, developing extensions for non-rational models, and applying the method to a wider range of complex real-world biological and engineering problems.

Acknowledgements

We are grateful to Julio Banga and Eva Balsa-Canto for extensive feedback on the use of AMIGO2, to Henning Schmidt, developer of IQRTools/IQDesktop for numerous discussions on installation of the software, to Gleb Pogudin for helpful feedback on the proposed algorithm and information on other parameter estimation approaches, to Xinglong Zhou and Jianing Qi for initial explorations of the role of polynomial interpolation, to the referees for useful suggestions, and finally to the CCiS at Queens College and CIMS NYU for the computational resources.

This work was supported by the NSF grants DMS-1760448, DMS-1853650, CCF-2212460, CCF-2212461, CCF-2212462, PSC-CUNY grants #65605-00 53, #66551-00 54, and DARPA HR00112290091. This research was, in part, funded by the U.S. Government. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. Approved for Public Release, Distribution Unlimited.

References

[1] Y. Bard, Nonlinear Parameter Estimation, Academic Press, New York, 1974.

[2] J. Beck, K. Arnold, Parameter Estimation in Engineering and Science, Wiley, New York, 1977.

[3] L. Ljung, System Identification: Theory for the User, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[4] E. Walter, L. Pronzato, Identification of parametric models from experimental data, Communications and control engineering, Springer, 1997.

[5] E. Sontag, For differential equations with r parameters, $2r + 1$ experiments are enough for identification, Journal of Nonlinear Science 12 (6) (2003) 553–583.
URL https://doi.org/https://doi.org/10.1007/s00332-002-0506-0

[6] H. Schmidt, M. Jirstrand, Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology, Bioinformatics 22 (4) (2005) 514–515.
URL https://doi.org/10.1093/bioinformatics/bti799

[7] Z. Li, M. R. Osborne, T. Prvan, Parameter estimation of ordinary differential equations, IMA Journal of Numerical Analysis 25 (2) (2005) 264–285.
URL https://doi.org/10.1093/imanum/drh016
[8] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, U. Kummer, COPASI—a COmplex PAthway SImulator, Bioinformatics 22 (24) (2006) 3067–3074.
URL https://doi.org/10.1093/bioinformatics/btl485
[9] M. Fliess, C. Join, H. Sira-Ramirez, Non-linear estimation is easy, International Journal of Modelling, Identification and Control 4 (1) (2008) 12–27.
URL https://doi.org/10.1504/IJMIC.2008.020996
[10] J. A. Egea, D. Henriques, T. Cokelaer, A. F. Villaverde, A. MacNamara, D.-P. Danciu, J. R. Banga, J. Saez-Rodriguez, Meigo: an open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics, BMC Bioinformatics 15 (1) (2014) 136.
URL https://doi.org/10.1186/1471-2105-15-136
[11] E. Balsa-Canto, D. Henriques, A. Gábor, J. R. Banga, AMIGO2, a toolbox for dynamic modeling, optimization and control in systems biology, Bioinformatics 32 (21) (2016) 3357–3359.
URL https://doi.org/10.1093/bioinformatics/btw411
[12] A. Raue, B. Steiert, M. Schelker, C. Kreutz, T. Maiwald, H. Hass, J. Vanlier, C. Tönsing, L. Adlung, R. Engesser, W. Mader, T. Heinemann, J. Hasenauer, M. Schilling, T. Höfer, E. Klipp, F. Theis, U. Klingmüller, B. Schöberl, J. Timmer, Data2Dynamics: a modeling environment tailored to parameter estimation in dynamical systems, Bioinformatics 31 (21) (2015) 3558–3560.
URL https://doi.org/10.1093/bioinformatics/btv405
[13] A. Gábor, A. F. Villaverde, J. R. Banga, Parameter identifiability analysis and visualization in large-scale kinetic models of biosystems, BMC Systems Biology 11 (1) (2017) 54.
URL https://doi.org/10.1186/s12918-017-0428-y
[14] A. Villaverde, F. Fröhlich, D. Weindl, J. Hasenauer, J. Banga, Benchmarking optimization methods for parameter estimation in large kinetic models, Bioinformatics 35 (5) (2019) 830–838.
URL https://doi.org/10.1093/bioinformatics/bty736
[15] J. DiStefano III, Dynamic systems biology modeling and simulation, Academic Press, 2015.
[16] H. G. Bock, E. Kostina, J. P. Schlöder, Direct multiple shooting and generalized Gauss-Newton method for parameter estimation problems in ODE models, in: Multiple Shooting and Time Domain Decomposition Methods: MuS-TDD, Heidelberg, May 6-8, 2013, Springer, 2015, pp. 1–34.
URL https://doi.org/10.1007/978-3-319-23321-5_1
[17] O. Aydogmus, A. H. Tor, A modified multiple shooting algorithm for parameter estimation in odes using adjoint sensitivity analysis, Applied Mathematics and Computation 390 (2021) 125644.
URL https://doi.org/10.1016/j.amc.2020.125644
[18] E. M. Turan, J. Jäschke, Multiple shooting for training neural differential equations on time series, IEEE Control Systems Letters 6 (2021) 1897–1902.
URL https://doi.org/10.1109/LCSYS.2021.3135835
[19] Y. Ma, V. Dixit, M. J. Innes, X. Guo, C. Rackauckas, A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions, in: 2021 IEEE High Performance Extreme Computing Conference (HPEC), 2021, pp. 1–9.
URL https://doi.org/10.1109/HPEC49654.2021.9622796
[20] I. M. Navon, Practical and theoretical aspects of adjoint parameter estimation and identifiability in meteorology and oceanography, Dynamics of Atmospheres and Oceans 27 (1-4) (1998) 55–79.
URL https://doi.org/10.1016/S0377-0265(97)00032-8
[21] K. Łakomiec, K. Fujarewicz, Parameter estimation of non-linear models using adjoint sensitivity analysis, Advanced Approaches to Intelligent Information and Database Systems (2014) 59–68.
URL https://doi.org/10.1007/978-3-319-05503-9_6
[22] R. Larsson, Z. Sjanic, M. Enqvist, L. Ljung, Direct prediction-error identification of unstable nonlinear systems applied to flight test data, IFAC Proceedings Volumes 42 (10) (2009) 144–149.
URL https://doi.org/10.3182/20090706-3-FR-2004.00024
[23] K. Khowaja, M. Shcherbatyy, W. Karl Härdle, Surrogate models for optimization of dynamical systems, in: Foundations of Modern Statistics, Springer, 2019, pp. 563–593.
URL https://doi.org/10.1007/978-3-031-30114-8_16
[24] J. Sun, L. Wang, D. Gong, A joint optimization algorithm based on the optimal shape parameter–gaussian radial basis function surrogate model and its application, Mathematics 11 (14) (2023) 3169.
URL https://doi.org/10.3390/math11143169
[25] E. Newman, L. Ruthotto, J. L. Hart, B. G. van Bloemen Waanders, Efficient training of neural network surrogate methods with variable projection., Tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States) (2020).
URL https://www.osti.gov/servlets/purl/1767098
[26] Y. Sun, E. Cucuzzella, S. Brus, S. H. K. Narayanan, B. Nadiga, L. Van Roekel, J. Hückelheim, S. Madireddy, Surrogate neural networks to estimate parametric sensitivity of ocean models (2023).
URL https://arxiv.org/abs/2311.08421
[27] L. Cai, L. Ren, Y. Wang, W. Xie, G. Zhu, H. Gao, Surrogate models based on machine learning methods for parameter estimation of left ventricular myocardium, Royal Society Open Science 8 (1) (2021) 201121.
URL https://doi.org/10.1098/rsos.201121
[28] R. Horst, P. M. Pardalos, N. V. Thoai, Introduction to Global Optimization, Kluwer Academic Publishers, 2000.
[29] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.

[30] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., 1989.
[31] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, 1996.
[32] J. O. Ramsay, C. Dalzell, Some tools for functional data analysis, Journal of the Royal Statistical Society: Series B (Methodological) 53 (3) (1991) 539–561.
URL https://doi.org/10.1111/j.2517-6161.1991.tb01844.x
[33] D. Campbell, Probabilistically solving differential equation models (Nov 2013).
URL https://www.birs.ca/events/2013/5-day-workshops/13w5151/videos/watch/201311140905-Campbell.html
[34] N. J.-B. Brunel, Parameter estimation of ODE's via nonparametric estimators, Electronic Journal of Statistics 2 (none) (2008) 1242 – 1267. doi:10.1214/07-EJS132.
URL https://doi.org/10.1214/07-EJS132
[35] A. Poyton, M. S. Varziri, K. B. McAuley, P. J. McLellan, J. O. Ramsay, Parameter estimation in continuous-time dynamic models using principal differential analysis, Computers & Chemical Engineering 30 (4) (2006) 698–708.
URL https://doi.org/10.1016/j.compchemeng.2005.11.008
[36] L. Li, M. B. Brown, K.-H. Lee, S. Gupta, Estimation and inference for a spline-enhanced population pharmacokinetic model, Biometrics 58 (3) (2002) 601–611.
URL https://doi.org/10.1111/j.0006-341X.2002.00601.x
[37] H. Liang, H. Wu, Parameter estimation for differential equation models using a framework of measurement error in regression models, Journal of the American Statistical Association 103 (484) (2008) 1570–1583.
URL https://doi.org/10.1198/016214508000000797
[38] E. Roesch, C. Rackauckas, M. P. Stumpf, Collocation based training of neural ordinary differential equations, Statistical Applications in Genetics and Molecular Biology 20 (2) (2021) 37–49.
URL https://doi.org/10.1515/sagmb-2020-0025
[39] J. O. Ramsay, G. Hooker, D. Campbell, J. Cao, Parameter estimation for differential equations: a generalized smoothing approach, Journal of the Royal Statistical Society Series B: Statistical Methodology 69 (5) (2007) 741–796.
URL https://doi.org/10.1111/j.1467-9868.2007.00610.x
[40] D. Campbell, O. Chkrebtii, Maximum profile likelihood estimation of differential equation parameters through model based smoothing state estimates, Mathematical Biosciences 246 (2) (2013) 283–292.
URL https://doi.org/10.1016/j.mbs.2013.03.011
[41] F. Boulier, A. Korporal, F. Lemaire, W. Perruquetti, A. Poteaux, R. Ushirobira, An algorithm for converting nonlinear differential equations to integral equations with an application to parameter estimation from noisy data, in: V. Gerdt, W. Koepf, W. Seiler, E. Vorozhtsov (Eds.), Computer Algebra in Scientific Computing. CASC 2014, Lecture Notes in Computer Science, Springer, Cham, 2014, pp. 28–43.
URL https://doi.org/10.1007/978-3-319-10515-4_3
[42] C. Jauberthie, N. Verdière, Bounded-error parameter estimation using integro-differential equations for Hindmarsh-Rose model, Algorithms 15 (2022) 179.
URL https://doi.org/10.3390/a15060179
[43] N. Verdière, D. Manceau, S. Zhu, L. Denis-Vidal, Inverse problem for a coupling model of reaction-diffusion and ordinary differential equations systems. Application to an epidemiological model, Applied Mathematics and Computation 375 (2020) 125067.
URL https://doi.org/10.1016/j.amc.2020.125067
[44] S. Zhu, N. Verdière, L. Denis-Vidal, D. Kateb, Identifiability analysis and parameter estimation of a chikungunya model in a spatially continuous domain, Ecological Complexity 34 (2018) 80–88.
URL https://doi.org/10.1016/j.ecocom.2017.12.004
[45] H. Sira-Ramírez, C. García-Rodríguez, J. Cortés-Romero, A. Luviano-Juárez, Algebraic Identification and Estimation Methods in Feedback Control Systems, Wiley, 2014.
URL https://doi.org/10.1002/9781118730591
[46] R. Dong, C. Goodbrake, H. Harrington, G. Pogudin, Differential elimination for dynamical models via projections with applications to structural identifiability, SIAM Journal on Applied Algebra and Geometry 7 (1) (2023).
URL https://doi.org/10.1137/22M1469067
[47] A. Quadrat, Towards an effective study of the algebraic parameter estimation problem, IFAC-PapersOnLine 50 (1) (2017) 6220–6225.
URL https://doi.org/10.1016/j.ifacol.2017.08.1021
[48] M. Komatsu, T. Yaguchi, Method for estimating hidden structures determined by unidentifiable state-space models and time-series data based on the Gröbner basis (2020).
URL https://arxiv.org/abs/2012.11906
[49] H. Hong, A. Ovchinnikov, G. Pogudin, C. Yap, SIAN: software for structural identifiability analysis of ODE models, Bioinformatics 35 (16) (2019) 2873–2874.
URL https://doi.org/10.1093/bioinformatics/bty1069
[50] H. Hong, A. Ovchinnikov, G. Pogudin, C. Yap, Global identifiability of differential models, Communications on Pure and Applied Mathematics 73 (9) (2020) 1831–1879.
URL https://doi.org/10.1002/cpa.21921
[51] D. J. Bates, J. D. Hauenstein, N. Meshkat, Identifiability and numerical algebraic geometry, PLoS one 14 (12) (2019) e0226299.
URL https://doi.org/10.1371/journal.pone.0226299
[52] P. Breiding, S. Timme, HomotopyContinuation.jl: A Package for Homotopy Continuation in Julia, in: International Congress on Mathematical Software, Springer, 2018, pp. 458–465.
URL https://doi.org/10.1007/978-3-319-96418-8_54
[53] M. S. Floater, K. Hormann, Barycentric rational interpolation with no poles and high rates of approximation 107 (2) 315–331. doi:

10.1007/s00211-007-0093-y.
URL https://doi.org/10.1007/s00211-007-0093-y
[54] Y. Nakatsukasa, O. Sète, L. N. Trefethen, The AAA Algorithm for Rational Approximation 40 (3) A1494–A1522. doi:10.1137/16M1106122.
URL https://epubs.siam.org/doi/10.1137/16M1106122
[55] L. Benet, D. P. Sanders, Taylorseries. jl: Taylor expansions in one and several variables in julia, Journal of Open Source Software 4 (36) (2019) 1043.
URL https://doi.org/10.21105/joss.01043
[56] J. Berthomieu, C. Eder, M. Safey El Din, msolve: A Library for Solving Polynomial Systems, in: 2021 International Symposium on Symbolic and Algebraic Computation, 46th International Symposium on Symbolic and Algebraic Computation, Saint Petersburg, Russia, 2021. doi:10.1145/3452143.3465545.
URL https://hal.sorbonne-universite.fr/hal-03191666
[57] Oscar – open source computer algebra research system, version 0.11.3-dev (2023).
URL https://oscar.computeralgebra.de
[58] W. Decker, C. Eder, C. Fieker, M. Horn, M. Joswig (Eds.), The OSCAR book, 2024.
[59] R. Chartrand, Numerical differentiation of noisy, nonsmooth data, International Scholarly Research Notices 2011 (2011).
URL https://doi.org/10.5402/2011/164564
[60] A. Demin, A. Ovchinnikov, F. Rouillier, Parameter estimation in ode models with certified polynomial system solving (2025).
URL https://arxiv.org/
[61] C. Rackauckas, Q. Nie, Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia, The Journal of Open Research Software 5 (1), exported from https://app.dimensions.ai on 2019/05/05 (2017). doi:10.5334/jors.151.
URL https://app.dimensions.ai/details/publication/pub.1085583166
[62] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, Universal differential equations for scientific machine learning (2020).
URL https://arxiv.org/abs/2001.04385
[63] B. van der Pol, A theory of the amplitude of free and forced triode vibrations, Radio Rev. 1 (1920) 701–710.
[64] J. Lucero, J. Schoentgen, Modeling vocal fold asymmetries with coupled van der Pol oscillators, Acoust Soc. Am. 133 (060165) (2013).
URL https://doi.org/10.1121/1.4798467
[65] R. FitzHugh, Impulses and physiological states in theoretical models of nerve membrane, Biophysical Journal 1 (6) (1961) 445–466.
URL https://doi.org/10.1016/S0006-3495(61)86902-6
[66] J. Nagumo, S. Arimoto, S. Yoshizawa, An active pulse transmission line simulating nerve axon, Proceedings of the IRE 50 (10) (1962) 2061–2070.
URL https://doi.org/10.1109/JRPROC.1962.288235
[67] D. Wodarz, M. Nowak, Mathematical models of HIV pathogenesis and treatment, BioEssays 24 (12) (2002) 1178–1187.
URL https://doi.org/10.1002/bies.10196
[68] L. Z. Benet, General treatment of linear mammillary models with elimination from any compartment as used in pharmacokinetics, Journal of Pharmaceutical Sciences 61 (4) (1972) 536–541.
URL https://doi.org/10.1002/jps.2600610408
[69] Q. Hui, W. M. Haddad, V. Chellaboina, T. Hayakawa, Adaptive control of mammillary drug delivery systems with actuator amplitude constraints and system time delays, European Journal of Control 11 (6) (2005) 586–600.
URL https://doi.org/10.3166/ejc.11.586-600
[70] A. J. Lotka, Analytical note on certain rhythmic relations in organic systems, Proceedings of the National Academy of Sciences 6 (7) (1920) 410–415.
URL https://doi.org/10.1073/pnas.6.7.410
[71] V. Volterra, Variations and fluctuations of the number of individuals in animal species living together, ICES Journal of Marine Science 3 (1) (1928) 3–51.
URL https://doi.org/10.1093/icesjms/3.1.3
[72] H. Hass, C. Loos, E. Raimúndez-Álvarez, J. Timmer, J. Hasenauer, C. Kreutz, Benchmark problems for dynamic modeling of intracellular processes, Bioinformatics 35 (17) (2019) 3073–3082.
URL https://doi.org/10.1093/bioinformatics/btz020
[73] F. Crauste, J. Mafille, L. Boucinha, S. Djebali, O. Gandrillon, J. Marvel, C. Arpin, Identification of nascent memory CD8 T cells and modeling of their ontogeny, Cell Systems 4 (3) (2017) 306–317.
URL https://doi.org/10.1016/j.cels.2017.01.014
[74] R. Muñoz-Tamayo, L. Puillet, J.-B. Daniel, D. Sauvant, O. Martin, M. Taghipoor, P. Blavy, Review: To be or not to be an identifiable model. is this a relevant question in animal science modelling?, animal 12 (2017) 1–12. doi:10.1017/S1751731117002774.
[75] P. Moate, R. Boston, T. Jenkins, I. Lean, Kinetics of ruminal lipolysis of triacylglycerol and biohydrogenation of long-chain fatty acids: New insights from old data, Journal of Dairy Science 91 (2) (2008) 731–742. doi:https://doi.org/10.3168/jds.2007-0398.
URL https://www.sciencedirect.com/science/article/pii/S0022030208714176
[76] G. Bellu, M. P. Saccomani, S. Audoly, L. D'Angió, DAISY: A new software tool to test global identifiability of biological and physiological systems, Computer Methods and Programs in Biomedicine 88 (1) (2007) 52 – 61.
URL http://dx.doi.org/10.1016/j.cmpb.2007.07.002
[77] N. Tuncer, T. T. Le, Structural and practical identifiability analysis of outbreak models., Mathematical Biosciences 299 (2018) 1–18.
URL https://doi.org/10.1016/j.mbs.2018.02.004

Appendix A: Benchmark Model Details

In this section, we present the ODE systems used in the benchmarks.

Harmonic Oscillator Model

It is a toy model for harmonic oscillators without damping.

$$\begin{cases} \dot{x_1} = -a\, x_2 \\ \dot{x_2} = \frac{1}{b}\, x_1 \end{cases} \tag{2}$$

$$\begin{cases} y_1 = x_1, \\ y_2 = x_2 \end{cases} \tag{3}$$

Van der Pol Oscillator Model

It models harmonic oscillators with nonlinear damping [63], which was introduced to study oscillations in electronic circuits, but was found to be useful for various other phenomena such as vocal fold [64].

$$\begin{cases} \dot{x_1} = a x_2 \\ \dot{x_2} = -x_1 - b(x_1^2 - 1)x_2 \end{cases} \tag{4}$$

$$\begin{cases} y_1 = x_1, \\ y_2 = x_2 \end{cases} \tag{5}$$

FitzHugh-Nagumo Model

It models (two-dimensional simplification of) spike generation in squid giant axons [65, 66].

$$\begin{cases} \dot{V} = g\left(V - \frac{V^3}{3} + R\right) \\ \dot{R} = \frac{1}{g}\,(V - a + b\,R) \end{cases} \tag{6}$$

$$\begin{cases} y_1 = V \end{cases} \tag{7}$$

HIV Dynamics Model

It models HIV infection dynamics during interaction with the immune system over the course of various treatments. It is based on [67, equations (6)] and the paragraph that follows.

$$\begin{cases} \dot{x} = \lambda - d\,x - \beta\,x\,v \\ \dot{y} = \beta\,x\,v - a\,y \\ \dot{v} = k\,y - u\,v \\ \dot{w} = c\,x\,y\,w - c\,q\,y\,w - b\,w \\ \dot{z} = c\,q\,y\,w - h\,z \end{cases} \tag{8}$$

$$\begin{cases} y_1 = w, \\ y_2 = z, \\ y_3 = x, \\ y_4 = y + v \end{cases} \tag{9}$$

Mammillary 3-compartment Model

It models pharmacokinetic processes, such as protein metabolism in organism [68, 69].

$$\begin{cases} \dot{x_1} = -(a_{21} + a_{31} + a_{01})\, x_1 + a_{12}\, x_2 + a_{13}\, x_3 \\ \dot{x_2} = a_{21}\, x_1 - a_{12}\, x_2 \\ \dot{x_3} = a_{31}\, x_1 - a_{13}\, x_3 \end{cases} \tag{10}$$

$$\begin{cases} y_1 = x_1, \\ y_2 = x_2 \end{cases} \tag{11}$$

Lotka-Volterra Model

It models interactions between two species (a predator and a prey) in an ecosystem [70, 71].

$$\begin{cases} \dot{r} = k_1\, r - k_2\, r\, w, \\ \dot{w} = k_2\, r\, w - k_3\, w \end{cases} \tag{12}$$

$$\begin{cases} y_1 = r \end{cases} \tag{13}$$

Crauste Model

It models the behavior of CD8 T-cells. It is based on the Crauste system [72, Section 4] originally described in [73].

$$\begin{cases} \dot{N} = -\mu_N\, N - \delta_{NE}\, N\, P \\ \dot{E} = \delta_{NE}\, N\, P - \mu_{EE}\, E^2 - \delta_{EL}\, E + \rho_E\, E\, P \\ \dot{S} = \delta_{EL}\, S - S\, \delta_{LM} - \mu_{LL}\, S^2 - \mu_{LE}\, E\, S \\ \dot{M} = \delta_{LM}\, S - \mu_M\, M \\ \dot{P} = \rho_P\, P^2 - \mu_P\, P - \mu_{PE}\, E\, P - \mu_{PL}\, S\, P \end{cases} \tag{14}$$

$$\begin{cases} y_1 = N, \\ y_2 = E, \\ y_3 = S + M, \\ y_4 = P \end{cases} \tag{15}$$

Biohydrogenation Model

It models the kinetic processes involved in biohydrogenation of non-esterified fatty acids [74, 75].

$$\begin{cases} \dot{x_4} = -\frac{k5\, x_4}{k_6 + x_4} \\ \dot{x_5} = \frac{k_5\, x_4}{k_6 + x_4} - \frac{k_7\, x_5}{k_8 + x_5 + x_6} \\ \dot{x_6} = \frac{k_7\, x_5}{k_8 + x_5 + x_6} - \frac{k_9\, x_6\, (k_{10} - x_6)}{k_{10}} \\ \dot{x_7} = \frac{k_9\, x_6\, (k_{10} - x_6)}{k_{10}} \end{cases} \tag{16}$$

$$\begin{cases} y_1 = x_4, \\ y_2 = x_5 \end{cases} \tag{17}$$

Mammillary 4-Compartment Model

It comes from examples used in the analysis of DAISY [76] identifiability software. It represents a 4-compartment system.

$$\begin{cases} \dot{x_1} = -k_{01}\, x_1 + k_{12}\, x_2 + k_{13}\, x_3 + \\ \qquad\qquad k_{14} x_4 - k_{21}\, x_1 - k_{31}\, x_1 - k_{41}\, x_1 \\ \dot{x_2} = -k_{12}\, x_2 + k_{21}\, x_1 \\ \dot{x_3} = -k_{13}\, x_3 + k_{31}\, x_1 \\ \dot{x_4} = -k_{14}\, x_4 + k_{41}\, x_1 \end{cases} \tag{18}$$

$$\begin{cases} y_1 = x_1, \\ y_2 = x_2, \\ y_3 = x_3 + x_4 \end{cases} \tag{19}$$

SEIR Model

It models an epidemic that involves compartments related to disease progression stages with prevalence observations [77].

$$\begin{cases} \dot{S} = -b\, S\, I/N \\ \dot{E} = b\, S\, I/N - \nu\, E \\ \dot{I} = \nu\, E - a\, I \\ \dot{N} = 0 \end{cases} \tag{20}$$

$$\begin{cases} y_1 = I, \\ y_2 = N \end{cases} \tag{21}$$

Appendix B: Detailed Benchmark Results

Harmonic

Software	IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Random Data 1	58.1	58.1	58.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 2	62.5	62.5	62.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 3	54.7	54.7	327.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 4	81.3	81.3	81.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 5	75.8	75.8	75.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 6	64.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 7	66.6	66.6	66.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 8	63.5	63.5	63.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 9	79.8	168.9	227.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 10	56.0	56.0	56.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Van der Pol

Software	IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Random Data 1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 6	0.0	0.0	96.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 7	75.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

FitzHugh-Nagumo

Software	IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Random Data 1	82.0	165.1	128.0	9.3	6.5	15.4	0.0	0.0	0.0	0.0
Random Data 2	71.0	164.9	293.3	0.2	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 3	58.0	58.0	58.0	0.3	1.1	0.1	0.0	0.0	0.0	0.0
Random Data 4	228.3	243.5	237.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 5	66.6	105.7	105.7	0.9	0.3	0.2	0.0	0.0	0.0	0.0
Random Data 6	55.9	174.2	274.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 7	81.2	81.2	88.0	70.2	143.7	178.1	0.0	0.0	2.2	0.0
Random Data 8	35.8	113.6	118.8	0.9	0.1	0.4	0.0	0.0	0.0	0.0
Random Data 9	101.3	191.5	191.5	1.3	7.3	9.0	0.3	0.3	0.3	0.0
Random Data 10	43.8	13.2	608.2	0.0	0.0	0.3	0.0	0.0	0.0	0.0

HIV

Software	IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Random Data 1	74.5	90.0	67.6	2.9	7.8	11.9	0.0	0.0	0.0	0.0
Random Data 2	125.8	113.5	119.2	0.8	0.1	310.2	0.0	0.0	0.0	0.0
Random Data 3	74.7	89.3	148.4	45.2	8.1	18.8	0.0	0.0	0.0	0.0
Random Data 4	66.4	185.7	264.6	0.5	0.0	0.0	0.0	0.0	0.0	0.0
Random Data 5	74.3	81.9	91.7	0.1	151.7	151.8	0.0	0.0	143.1	0.0
Random Data 6	80.1	68.2	91.5	44.2	79.8	96.8	184.3	4.6	0.0	0.0
Random Data 7	76.4	86.9	214.2	40.2	43.1	306.5	0.5	297.7	0.0	0.0
Random Data 8	89.1	118.7	204.4	1.0	0.5	0.4	0.0	0.0	0.0	0.0
Random Data 9	67.6	82.3	80.0	12.4	78.3	131.8	0.0	130.2	140.5	0.0
Random Data 10	127.4	125.8	165.3	0.5	0.1	0.7	0.0	0.0	0.0	0.0

Mammillary 3

Software	IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Random Data 1	67.0	67.2	80.3	0.5	12.5	9.4	0.0	0.0	0.0	0.0
Random Data 2	78.2	85.4	98.1	44.4	89.4	105.4	0.0	0.0	0.0	0.0
Random Data 3	89.1	104.1	69.5	6.0	4.3	12.6	0.0	0.0	0.0	0.0
Random Data 4	51.6	51.6	51.6	7.2	13.2	29.1	0.0	0.0	0.0	0.0
Random Data 5	57.9	60.1	102.9	1.1	20.0	39.2	0.0	0.0	0.0	0.0
Random Data 6	78.8	98.2	374.3	0.4	0.6	4.6	0.0	0.0	0.0	0.0
Random Data 7	111.6	300.4	71.8	4.4	6.3	5.3	0.0	0.0	0.0	0.0
Random Data 8	93.4	136.7	275.4	5.1	17.1	10.5	0.0	0.0	0.0	0.0
Random Data 9	70.4	90.8	90.8	16.9	3.7	11.0	0.0	0.0	0.0	0.0
Random Data 10	63.3	48.7	67.1	33.4	47.5	20.2	0.0	0.0	0.0	0.0

Lotka-Volterra

Software	IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Random Data 1	80.1	80.1	80.1	11.5	29.6	55.7	0.0	166.6	0.0	0.0
Random Data 2	64.2	77.4	77.4	19.7	28.0	42.9	0.7	0.0	0.0	0.0
Random Data 3	64.1	75.6	75.6	45.3	72.2	95.4	0.0	59.0	5.3	0.0
Random Data 4	76.1	76.1	76.1	262.8	242.1	229.5	0.0	0.0	3.8	0.0
Random Data 5	70.7	65.8	65.8	22.1	68.1	112.2	0.0	0.0	0.0	0.0
Random Data 6	76.8	76.8	76.8	43.1	41.9	44.6	0.0	0.0	0.0	0.0
Random Data 7	77.1	77.1	77.1	106.0	187.2	296.0	66.9	749.2	0.0	0.0
Random Data 8	69.4	80.9	80.9	4.5	8.7	23.7	64.4	150.8	249.4	0.0
Random Data 9	70.4	71.1	71.1	10.2	89.6	192.5	0.0	0.0	2.1	0.0
Random Data 10	72.7	72.7	72.7	22.0	1.6	24.3	4.2	0.0	36.9	0.0

Crauste

Software	IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Random Data 1	86.9	119.2	150.4	12.2	27.0	39.7	0.0	0.0	81.1	0.0
Random Data 2	171.1	239.7	238.1	22.1	82.3	135.8	0.0	53.9	132.3	0.0
Random Data 3	78.7	66.2	467.4	61.5	121.2	203.0	13.8	113.2	0.0	0.0
Random Data 4	134.6	109.6	129.2	30.8	69.5	130.2	12.0	0.0	59.1	0.0
Random Data 5	92.7	189.2	66.2	63.1	154.3	261.7	0.0	101.2	0.0	0.0
Random Data 6	80.6	167.0	84.7	16.4	35.2	484.7	0.0	0.0	49.1	0.0
Random Data 7	83.6	78.0	83.0	9.4	32.8	69.7	0.0	0.0	0.0	0.0
Random Data 8	82.5	108.9	86.6	11.9	65.0	117.0	0.0	127.8	195.0	0.0
Random Data 9	125.2	101.6	304.6	42.8	120.7	153.9	0.0	248.6	0.0	0.0
Random Data 10	59.8	72.6	187.2	203.0	21.1	668.3	0.0	412.3	0.0	0.0

Biohydrogenation

Software	IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Random Data 1	83.6	120.3	132.8	16.9	31.5	206.3	0.0	0.0	0.0	0.0
Random Data 2	76.7	372.9	69.7	69.1	161.5	646.2	0.0	34.2	44.4	0.0
Random Data 3	72.5	110.5	72.6	169.6	24.8	67.5	0.0	0.0	0.0	0.0
Random Data 4	66.8	90.7	169.5	38.7	38.1	341.5	0.0	0.0	0.0	0.1
Random Data 5	80.9	411.8	213.4	33.7	303.8	181.6	43.0	0.0	66.9	0.0
Random Data 6	107.6	85.1	73.4	22.7	59.7	359.8	0.0	53.4	0.0	0.0
Random Data 7	171.5	381.4	344.3	132.4	407.4	607.8	0.0	0.0	0.0	0.0
Random Data 8	81.5	63.1	69.0	147.5	214.3	227.7	62.3	0.0	48.8	0.0
Random Data 9	118.4	116.4	80.2	111.0	223.9	395.3	72.9	47.8	48.1	0.0
Random Data 10	78.0	88.4	77.6	35.7	45.7	31.8	0.0	0.0	75.1	0.0

Mammillary 4

Software	IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Random Data 1	54.0	73.2	111.7	15.7	26.0	116.4	32.8	41.2	29.7	0.4
Random Data 2	63.4	235.1	87.6	213.9	2.5	2.4	0.0	0.0	0.0	0.0
Random Data 3	72.0	58.1	65.8	25.0	5.3	6.2	0.0	74.1	74.1	0.0
Random Data 4	62.5	147.1	55.5	53.6	53.6	68.1	137.7	106.3	144.1	0.2
Random Data 5	87.8	89.9	156.5	12.9	129.0	191.7	0.0	26.1	26.1	0.0
Random Data 6	79.2	86.8	172.7	16.7	157.7	190.0	9.8	66.3	0.0	0.0
Random Data 7	108.3	98.4	130.2	40.2	40.2	10.9	118.1	0.0	109.4	0.0
Random Data 8	183.1	159.8	108.4	66.9	68.6	408.2	0.0	144.7	123.8	0.0
Random Data 9	157.8	84.5	63.3	124.7	95.3	95.3	0.0	89.0	0.0	0.0
Random Data 10	72.9	63.0	101.8	92.8	99.0	99.0	0.0	18.9	82.8	0.0

SEIR

Software	IQM			SciML			AMIGO2			Parameter Estimation.jl
Search Range	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	[0,1]	[0,2]	[0,3]	Any
Random Data 1	99.7	140.4	204.1	4.9	55.4	109.1	0.0	12.1	0.0	0.0
Random Data 2	103.8	192.9	294.3	24.7	26.0	62.1	0.0	53.3	55.6	0.0
Random Data 3	296.0	720.9	1096.5	24.2	20.0	59.0	28.3	10.9	38.7	0.0
Random Data 4	73.8	112.7	161.6	19.4	24.6	63.9	33.6	1.7	65.4	0.0
Random Data 5	134.1	255.1	390.5	10.1	43.7	101.8	46.3	46.3	0.0	0.0
Random Data 6	82.8	133.1	198.1	16.5	19.6	57.6	7.4	24.1	22.9	0.0
Random Data 7	84.0	125.0	185.6	45.3	122.7	178.6	29.0	15.5	16.9	0.0
Random Data 8	129.5	180.4	266.3	11.6	63.1	143.6	0.0	75.8	74.8	0.0
Random Data 9	90.8	165.5	249.9	17.7	100.1	165.2	113.9	167.2	226.8	0.0
Random Data 10	228.4	281.6	342.3	9.8	130.8	240.0	0.0	0.0	7.1	0.0