Making Algebraic Parameter Estimation Practical for Noisy Data via Gaussian Process Regression

Oren Bassik[*] Alexander Demin[†] Alexey Ovchinnikov[‡]

Abstract

Parameter estimation for ordinary differential equation (ODE) models is a critical task often hindered by noisy data and the limitations of traditional optimization methods. The differential-algebraic approach offers theoretical advantages, such as eliminating the need for initial parameter guesses, and finding all global solutions without getting stuck in local minima. The algebraic approach has been historically hampered by its sensitivity to measurement noise. In this work, we resolve this limitation by replacing the noise-sensitive derivative estimation step with a robust probabilistic framework based on Gaussian Process Regression (GPR). We demonstrate through a comprehensive benchmark across 550 datasets on 11 dynamical systems that our GPR-enhanced method is highly competitive with modern optimization techniques. When paired with a local refinement step, our method achieves the highest success rate for high-precision parameter estimation (55.6% at a 1% error tolerance) and the lowest median maximum error of all tested methods, making the powerful differential-algebraic method a practical and reliable tool for parameter estimation from real-world, noisy data.

1 Introduction

Parameter estimation for systems of ordinary differential equations (ODEs) is a fundamental challenge in nearly every field of science and engineering. While ODE models are ubiquitous, their practical use requires precise values for unknown parameters that must be inferred from experimental data. The dominant paradigm for this task is nonlinear optimization, where parameter values are sought that minimize the discrepancy between a model's simulated output and observed measurements. These "shooting" methods require repeated numerical integration of the ODE system, often once per iteration per candidate parameter set, which can be computationally intensive. Furthermore, these methods are susceptible to well-known drawbacks: they often require carefully chosen initial guesses to avoid converging to non-optimal local minima, and they typically identify only a single parameter set, which is insufficient for systems where parameters are only locally identifiable.

An alternative, the differential-algebraic approach, offers a compelling solution to these challenges. By transforming the ODE system into a set of algebraic equations, this method can, in principle, find all valid parameter sets without requiring any initial guesses from the user. This is achieved by converting the problem into one of solving a multivariate polynomial system, for which robust numerical solvers exist that can find all solutions globally. This makes it a powerful tool for global analysis and for systems with complex parameter landscapes. However, this approach has a critical vulnerability: it relies on computing high-order derivatives of the observed output signals,

[*]CUNY Graduate Center, New York, NY, USA.
[†]HSE University, Moscow, Russia.
[‡]CUNY Queens College and CUNY Graduate Center, New York, NY, USA.

a process that is notoriously sensitive to measurement noise. This sensitivity has largely prevented the widespread adoption of algebraic methods for practical applications, where data is invariably corrupted by noise.

To address this challenge, we introduce a methodology that replaces the noise-sensitive derivative estimation step with a robust statistical framework: Gaussian Process Regression (GPR). While the foundational algebraic framework was presented in [3] and benchmarked on noise-free data, the central contribution of this paper is demonstrating that the integration of GPR makes the method practical for realistic, noisy measurements, a claim we validate with an extensive benchmark.

The main contributions of this paper are:

1. The introduction of Gaussian Process Regression into the differential-algebraic parameter estimation workflow, extending this powerful theoretical method to be robust for noisy experimental data.

2. A comprehensive benchmark of this GPR-enhanced method against modern local and global optimization techniques across a diverse suite of 11 ODE systems and multiple noise levels.

3. A detailed analysis of the trade-offs between precision, robustness, and speed, providing practical guidance on which method to choose based on application requirements.

2 Background

Our methodology builds upon two distinct areas of scientific computing: the theory of differential algebra for parameter estimation and the statistical framework of Gaussian Process Regression. This section provides the necessary background in both areas.

2.1 The Differential-Algebraic Approach to Parameter Estimation

The differential-algebraic approach transforms a parameter estimation problem for a system of differential equations into a problem of solving a system of algebraic equations. The key insight is that the parameters of an ODE system are intrinsically linked to the higher-order derivatives of its outputs. A detailed exposition of this method can be found in [3, Section 3].

Given an ODE system with rational functions $\mathbf{f}$ and $\mathbf{g}$:

$$\begin{cases} \mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p}) \\ \mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{p}) \end{cases} \tag{1}$$

where $\mathbf{x}$ are the states, $\mathbf{p}$ are the parameters, and $\mathbf{y}$ are the observable outputs, one can repeatedly differentiate the output equation $\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{p})$ with respect to time. Each differentiation introduces higher-order derivatives of $\mathbf{y}$ on one side and replaces derivatives of $\mathbf{x}$ with expressions from the system dynamics on the other. This process systematically eliminates the state variables $\mathbf{x}$ and their derivatives, resulting in a system of equations that algebraically relate the parameters $\mathbf{p}$ to the outputs $\mathbf{y}$ and their time derivatives $(\mathbf{y}, \mathbf{y}', \mathbf{y}'', \dots)$.

At any specific time point t_0, this symbolic system can be made numerical. If one can provide numerical values for the derivatives of the outputs at t_0, i.e., $y(t_0), y'(t_0), y''(t_0), \dots$, the result is a multivariate polynomial system solely in the unknown parameters $\mathbf{p}$. This system can then be solved using numerical algebraic geometry techniques, such as homotopy continuation, to find all possible solutions for $\mathbf{p}$ that are consistent with the observed data at that instant.

The power of this method lies in its ability to convert a global, dynamic optimization problem into a local, algebraic one. However, its practical success hinges entirely on the ability to obtain accurate numerical estimates for the derivatives of $\mathbf{y}(t)$ from discrete, noisy measurements.

2.2 Gaussian Process Regression for Derivative Estimation

Traditional methods for obtaining derivatives from data, such as finite differences or interpolation via splines or rational functions, are highly sensitive to noise. Differentiation is a high-pass filter, meaning it amplifies high-frequency components of a signal—and measurement noise is predominantly high-frequency. This amplification can render derivative estimates unusable, causing the algebraic method to fail.

To overcome this, we turn to Gaussian Process Regression (GPR). GPR is a non-parametric, Bayesian approach to regression that is exceptionally well-suited for fitting noisy data and estimating derivatives. Instead of fitting a single function to the data, GPR defines a prior distribution over a space of functions and then updates this prior to a posterior distribution based on the observed data.

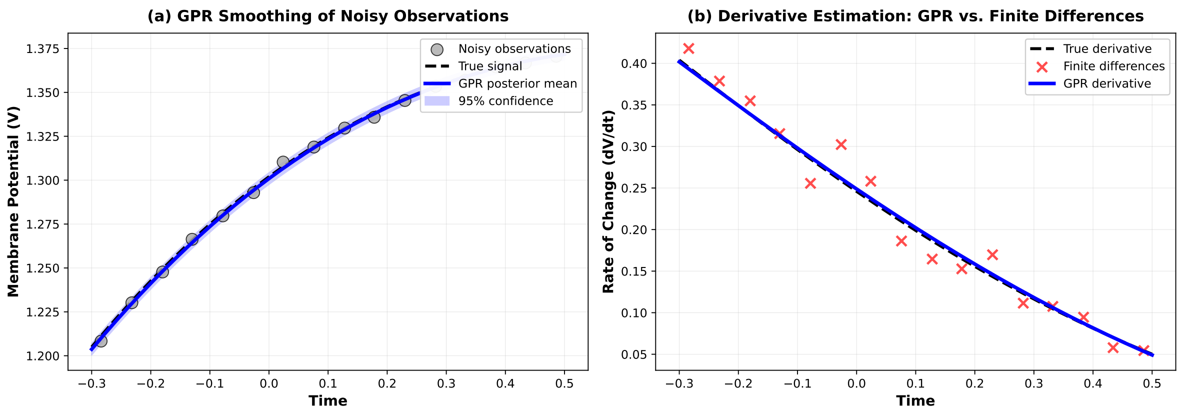

Figure 1: Demonstration of Gaussian Process Regression for derivative estimation using the FitzHugh-Nagumo system with 1% noise and 40 sparse observations. **(a)** GPR smoothing: Sparse noisy observations (gray points) are fit with a GPR model (blue line with 95% confidence interval). The GPR posterior mean accurately recovers the underlying smooth signal (black dashed line) despite significant measurement noise. **(b)** Derivative estimation comparison: Finite differences applied to the sparse noisy data (red X markers) versus the GPR-based derivative (blue line), compared against the true derivative (black dashed line). This demonstrates GPR's ability to produce stable, accurate derivative estimates even from noisy sparse data—the key enabler for the algebraic parameter estimation method.

2.2.1 A Brief Primer on Gaussian Process Regression

For readers unfamiliar with the technique, we provide a brief mathematical overview of Gaussian Process Regression [22]. A Gaussian Process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution. A GP is fully specified by its mean function $m(x)$ and covariance function (or kernel) $k(x, x')$. We can write this as:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{2}$$

In a regression context, we assume that our noisy observations y_i at inputs t_i are generated from a latent function $f(t)$ corrupted by Gaussian noise: $y_i = f(t_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$.

Given a set of training data (our noisy observations), the GP framework allows us to compute the posterior predictive distribution for the function's value at a new point, t_*. This posterior is also a Gaussian distribution, and its mean can be used as the optimal smoothed estimate of the underlying function. The posterior mean function, $\hat{f}(t)$, is analytic and can be differentiated to any order.

The choice of kernel is crucial as it encodes our prior assumptions about the function's properties, such as smoothness. For this work, we use the common squared exponential (or RBF) kernel:

$$k(t, t') = \sigma_f^2 \exp\left(-\frac{(t - t')^2}{2\ell^2}\right) \tag{3}$$

This kernel is infinitely differentiable and is defined by two hyperparameters: the length scale ℓ, which controls the smoothness or characteristic frequency of the function, and the signal variance σ_f^2. These, along with the noise variance σ_n^2, are not set manually but are learned from the data by maximizing the log marginal likelihood, a process that automatically adapts the model to the specific characteristics of the observed data.

The key features of GPR that make it ideal for this task are:

1. **Principled Smoothing:** A GPR model's smoothness assumptions are encoded in its kernel function (e.g., the Radial Basis Function kernel). This provides a principled way to smooth the data that is more robust than ad-hoc filtering.

2. **Explicit Noise Modeling:** GPR explicitly models the measurement noise as part of its statistical model. The variance of this noise can be learned directly from the data as a hyperparameter, allowing the model to adapt to the level of noise present.

3. **Analytic Differentiability:** The mean of the posterior distribution from a GPR model is a smooth, analytic function. This posterior mean function can be differentiated to any order to obtain reliable derivative estimates.

As demonstrated in a comprehensive benchmark study [2], the combination of GPR for smoothing followed by automatic differentiation of its posterior mean is the most robust and accurate method for estimating high-order derivatives from noisy time-series data. By using GPR, we are not just interpolating the data; we are performing a principled regression that separates the underlying smooth signal from the corrupting noise, which is the essential prerequisite for robust derivative estimation.

3 Methodology: A GPR-Enhanced Algebraic Framework

The core of our proposed method is a framework that systematically transforms the problem of parameter estimation from noisy time-series data into a solvable algebraic problem. The crucial innovation lies in the use of Gaussian Process Regression to robustly estimate the required derivatives from data. We will now describe the steps of the algorithm in detail, borrowing from the exposition in [3].

3.1 Step 1: Structural Identifiability Analysis

Before attempting to estimate parameters, we first perform a structural identifiability analysis on the ODE model using methods from differential algebra [17, 23], as implemented in `SIAN.jl` [14] and `StructuralIdentifiability.jl`. Structural identifiability analysis determines, from the model structure alone, whether it is theoretically possible to uniquely determine the parameters from noise-free data. This crucial first step, typically absent from optimization-based workflows, yields two key outputs: (1) the number of distinct parameter sets, k, that are consistent with the model outputs, and (2) the required order of differentiation. It also identifies any parameters that are structurally unidentifiable; by definition, the values of these parameters do not impact the observables and they are excluded from our analysis.

3.2 Step 2: Symbolic Differentiation

Using the differentiation order determined from the identifiability analysis, we differentiate the equations of the ODE system symbolically with respect to time. This process, as described in Section 2.1, generates a set of symbolic equations relating the system parameters to the time derivatives of the state and output variables.

3.3 Step 3: GPR-based Derivative Estimation

This step is the core of our contribution, replacing the original noise-sensitive interpolation step with a robust regression framework. Given a set of noisy measurements $\{(t_i, \mathbf{y}_i)\}_{i=1}^n$ for each output component, we perform the GPR fitting and differentiation procedure.

3.4 Step 4: Polynomial System Formulation and Solving

The numerical derivative values obtained from GPR are substituted into the symbolic equations generated in Step 2. This results in a concrete, numerical multivariate polynomial system. The variables in this system are the unknown parameters $\mathbf{p}$ and the values of the state variables $\mathbf{x}$ and their derivatives at the chosen time point t_0.

This system is typically overdetermined. We reduce it to a square system by selecting a subset of equations that maximizes the rank of the Jacobian, ensuring a well-posed problem. This square system is then solved using a numerical algebraic geometry solver, such as one based on homotopy continuation (a method that tracks solutions along a continuous path from a simple, known system to the target system) [6], to find all real solutions for a single time point. This solver is modular in our software, and for smaller systems one can switch to a Groebner-based solver which can be more efficient.

3.5 Step 5: Aggregation via Multiple Time Points

In practice, performing the estimation at a single time point t_0 can be sensitive to the local data quality. To improve robustness, we repeat the estimation process at multiple time points (e.g., 10 points for the benchmarks in this paper). This "multiple shooting" approach yields a collection of candidate parameter sets from each point. This process is embarrassingly parallelizable, as each time point can be processed independently. The resulting sets of solutions are aggregated before the final filtering step.

3.6 Step 6: Solution Filtering and Validation

The aggregated set of candidate solutions is first filtered to remove any solutions that are non-physical (e.g., negative parameter values if known to be positive) or, optionally, lie outside user-provided bounds. For the remaining candidates, we perform a full numerical simulation of the original ODE system and compute the root-mean-square error (RMSE) against the original data. The solutions with RMSE below a tolerance (with a reasonable default provided) are selected as final solutions.

3.7 Step 7: Optional Polishing Step

In addition to the raw algebraic solutions, we consider a polished variant in which the ODEPE-GPR solution is used as an initial guess for a local least-squares refinement using the SciML optimizer. For this step, we use hyperparameters appropriate for a local search, assuming the initial guess is already close to a minimum, and run a limited number of iterations. We denote this variant `ODEPE-GPR (polished)`.

4 Experimental Setup

To validate the performance and robustness of our GPR-enhanced algebraic method (which we will refer to as `ODEPE-GPR`), we conducted a comprehensive benchmark against established parameter estimation methods. This section details the benchmark systems, baseline methods, data generation protocol, and evaluation metrics used in our study.

4.1 Benchmark Systems

We selected a diverse suite of 11 ODE models from various scientific domains to ensure our evaluation is generalizable. The systems were chosen to cover a range of complexities in terms of the number of parameters, number of states, and dynamic behaviors (e.g., oscillations, stiff dynamics). The benchmark suite includes well-known models from ecology (Lotka-Volterra), epidemiology (SEIR), neuroscience (FitzHugh-Nagumo), and immunology (Crauste). A full description of the models, including their equations and observed outputs, can be found in the supplementary materials.

4.2 Baseline Methods for Comparison

We compare the performance of our method against three distinct classes of parameter estimation software:

1. **Original Algebraic Method (`ODEPE-AAA`):** This is our own implementation of the differential-algebraic method that uses the AAA rational interpolation algorithm for derivative estimation instead of GPR. This baseline serves to directly isolate and quantify the improvement gained by incorporating GPR.

2. **Local Optimization (`SciML`):** We use the SciML ecosystem in Julia [21] to represent a standard, modern local optimization approach. This single-shooting method uses a gradient-based optimizer (BFGS) to minimize a least-squares loss function with a search range of $[0, 10]$. This baseline is representative of common practice in the field.

3. **Global Optimization (`AMIGO2`):** We use AMIGO2 [1], a widely-used toolbox in systems biology that implements a multi-start global optimization approach. This method performs

multiple local optimizations from different starting points to better explore the parameter space and avoid local minima. We test two common search ranges, $[0, 10]$ and $[0, 100]$.

For the optimization-based methods, we provide search ranges reflecting common scenarios where a user might have different levels of prior knowledge. Our method, `ODEPE-GPR`, does not require such a range, initial parameter guesses, or multi-start strategies. While users may optionally provide physical bounds or adjust the fit-quality tolerance used during solution filtering, the method provides reasonable defaults and can operate fully autonomously.

4.2.1 Implementation Details

For reproducibility, we provide the key implementation details for each method:

ODEPE-GPR: Gaussian process regression is performed using GaussianProcesses.jl with an RBF (squared exponential) kernel. An independent GP is fitted to each observed output component. Hyperparameters (length scale, signal variance, noise variance) are optimized via L-BFGS maximization of the log marginal likelihood. Derivatives are computed via automatic differentiation of the GP posterior mean. The polynomial system solver uses homotopy continuation (HomotopyContinuation.jl). For the polished variant, solutions are refined using L-BFGS optimization for up to 50 iterations.

SciML: Parameter estimation uses the BFGS optimizer from Optim.jl with a maximum of 200,000 iterations. The ODE system is solved using the Tsit5 adaptive Runge-Kutta method with absolute and relative tolerances of 10^{-13}. Gradients are computed using ForwardDiff.jl automatic differentiation.

AMIGO2: Global optimization uses the enhanced Scatter Search (eSS) algorithm with a maximum of 200,000 function evaluations and 600-second time limit. Local refinement is performed using the nl2sol algorithm with up to 100,000 iterations and tolerances of 10^{-13}. ODE integration uses CVODES with absolute and relative tolerances of 10^{-13}.

4.3 Data Generation and Noise Protocol

For each benchmark system, we generated synthetic data to create a controlled experimental environment where the ground-truth parameters are known.

1. **Parameter Sampling:** For each of 10 experimental trials, true parameter values and initial conditions were sampled uniformly from the interval $[0.1, 0.9]$.

2. **Data Simulation:** The ODE system was solved using a high-precision numerical integrator (Vern9 with tolerances 10^{-13}) over the time interval $[-1, 1]$ to generate a baseline, noise-free trajectory. We recorded 1001 equally spaced time points for each system.

3. **Observables:** For each system, only a subset of state variables were treated as observable, reflecting realistic experimental scenarios where not all system states can be directly measured. The specific observed outputs for each system are detailed in the supplementary materials.

4. **Noise Injection:** To simulate realistic experimental data, we added Gaussian white noise to the noise-free trajectory. We considered one noise-free condition and four nonzero noise levels. Independent, zero-mean Gaussian noise with standard deviation $\sigma = \alpha \cdot \mathrm{SD}(y_{\mathrm{true}})$ was added to each observed component at each time point, where $\alpha \in \{0, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}\}$. The same noise level α was applied to all outputs of a given system, giving a range from noise-free to significant noise conditions.

This protocol results in 11 systems × 5 noise conditions × 10 trials = 550 total datasets per method, allowing us to assess the robustness and average performance of each estimation method.

4.4 Evaluation Metrics

To quantify the performance of each method, we use two primary metrics. Importantly, any parameters known to be structurally unidentifiable from the analysis in Step 1 are excluded a priori from all error calculations.

1. **Success Rate:** A parameter estimation run is considered a "success" based on multiple relative error thresholds. We report the percentage of trials where the relative error for every identifiable parameter is less than 1% (SR-1), 10% (SR-10), and 50% (SR-50). This multi-threshold approach captures both the precision and the overall reliability of a method.

2. **Run-wise worst-parameter error:** For each run, we compute the maximum relative error over all identifiable parameters. The relative error for a parameter p is defined as $|p_{\text{true}} - p_{\text{estimated}}|/|p_{\text{true}}|$. For runs that failed to produce any result, we assign a penalty error of 10^6 to the maximum error for that run. We then summarize these per-run worst errors by reporting their median and 90th percentile (P90) across all runs. We use the median because it is robust to occasional outliers. This run-level aggregation ensures that a method is only deemed successful when *all* parameters in a run meet the quality threshold. For locally identifiable systems where our method returns multiple solutions, we select the solution closest to the known ground truth for error calculation.

5 Results

Our benchmark provides a nuanced view of the performance of different parameter estimation strategies. The results reveal a clear trade-off between precision, robustness, and computational speed.

5.1 Overall Performance

The aggregate performance of all methods across all test systems and noise levels is summarized in Table 1. Performance is evaluated using run-level metrics, where a run is deemed successful if the relative error for *all* its parameters falls below a specified threshold. We report the fraction of successful runs at 1%, 10%, and 50% thresholds (Success@X%). To characterize accuracy, we first find the maximum parameter error within each run and then report the median and 90th percentile (P90) of these maximum errors across all runs.

The results highlight a distinct trade-off between computational efficiency and parameter precision. `ODEPE-GPR (polished)` stands out for its accuracy, achieving the highest success rates at strict 1% (55.6%) and 10% (62.7%) error tolerances. It also delivers the lowest median maximum error (0.30%), making it the most precise method on average. This high precision, however, requires the most computation time (1571.9s). In contrast, `SciML` is the fastest method by a significant margin (222.2s) and demonstrates the most robustness at a looser tolerance, with the highest Success@50% rate (73.1%). Its P90 error, while high, suggests a more graceful degradation on the most challenging runs. The standard `ODEPE-GPR` and `AMIGO2 [0,10]` offer balanced performance profiles, positioning them as competitive intermediates.

Ultimately, the findings suggest that the optimal choice of method is application-dependent. For problems demanding the highest possible parameter accuracy where computational cost is a

Table 1: Overall performance with run-level aggregation. Success@X%: fraction of runs where *all* parameters have relative error <X%. Median/P90 Max Error: For each run, the maximum parameter error is computed; the median and 90th percentile across all runs are reported. Failed runs are assigned 10^6 penalty.

Method	Success@1%	Success@10%	Success@50%	Median Max (%)	P90 Max (%)
ODEPE-GPR	52.4	61.6	70.2	0.56	403.5
ODEPE-GPR (polished)	55.6	62.7	71.6	0.30	430.1
SciML	48.4	56.2	73.1	1.64	197.3
AMIGO2 [0,10]	53.5	60.2	67.3	0.36	772.0
AMIGO2 [0,100]	49.3	55.8	62.4	1.44	2316.7

secondary concern, `ODEPE-GPR (polished)` is the leading option. For high-throughput applications where speed is critical and moderate accuracy is sufficient, `SciML` provides a fast and robust alternative.

5.2 Robustness to Measurement Noise

The performance degradation as noise increases is detailed in Table 2. This view confirms the trade-offs seen in the overall performance.

Table 2: Median worst-parameter error (%) by noise level using run-level aggregation. For each run, the maximum parameter error is computed; the median across runs at each noise level is reported. Failed runs are assigned 10^6 penalty.

Method	0	10^{-8}	10^{-6}	10^{-4}	10^{-2}
ODEPE-GPR	0.00	0.11	0.93	29.89	216.34
ODEPE-GPR (polished)	0.00	0.04	0.36	19.61	230.15
SciML	0.01	0.01	0.07	9.89	93.26
AMIGO2 [0,10]	0.00	0.00	0.15	9.92	163.69
AMIGO2 [0,100]	0.00	0.00	0.16	8.28	150.10

In the noise-free and very low-noise regimes ($\alpha = 0$ to 10^{-6}), `ODEPE-GPR (polished)` achieves the best overall accuracy, with the lowest median worst-parameter error across systems. However, as the noise level increases past the critical threshold of 10^{-4}, GPR-based methods degrade more rapidly than the `SciML` optimizer, which proves to be the most robust method in high-noise conditions. On simpler systems such as the harmonic oscillator and Van der Pol, all three methods perform comparably at low noise.

To complement the error magnitude analysis, we examine success rates at different precision thresholds across noise levels. Tables 3, 4, and 5 show the percentage of runs achieving complete success (all parameters below threshold) at 10%, 50%, and 1% error tolerances, respectively.

Table 3: Success@10% by noise level. A run is successful if the maximum relative error across all parameters is less than 10%. Values shown are the percentage of successful runs at each noise level.

Method	0	10^{-8}	10^{-6}	10^{-4}	10^{-2}
ODEPE-GPR (polished)	98.2	80.9	72.7	41.8	20.0
SciML	68.2	70.0	68.2	50.9	23.6
AMIGO2 [0,10]	81.8	74.5	69.1	50.0	25.5

Table 4: Success@50% by noise level. A run is successful if the maximum relative error across all parameters is less than 50%. Values shown are the percentage of successful runs at each noise level.

Method	0	10^{-8}	10^{-6}	10^{-4}	10^{-2}
ODEPE-GPR (polished)	98.2	90.0	80.0	60.9	29.1
SciML	84.5	85.5	80.9	73.6	40.9
AMIGO2 [0,10]	83.6	79.1	75.5	64.5	33.6

Table 5: Success@1% by noise level. A run is successful if the maximum relative error across all parameters is less than 1%. Values shown are the percentage of successful runs at each noise level.

Method	0	10^{-8}	10^{-6}	10^{-4}	10^{-2}
ODEPE-GPR (polished)	97.3	72.7	58.2	32.7	17.3
SciML	62.7	65.5	61.8	37.3	14.5
AMIGO2 [0,10]	80.9	70.9	63.6	36.4	15.5

These success rate tables reveal distinct patterns in method reliability as noise increases. At the strictest 1% threshold (Table 5), `ODEPE-GPR (polished)` excels in noise-free conditions with 97.3% success, but this advantage narrows considerably as noise increases, with all methods converging to 14–17% success at the highest noise level. The 10% threshold (Table 3) shows a more pronounced divergence: `ODEPE-GPR (polished)` maintains strong performance in low-noise regimes, but `SciML` demonstrates superior robustness at medium-to-high noise levels. At the 50% threshold (Table 4), `SciML` achieves the highest success rate at 10^{-2} noise (40.9%), compared to 29.1% for `ODEPE-GPR (polished)`.

This analysis shows that methods do not fail catastrophically but degrade gracefully, producing less accurate estimates as noise increases. The choice of method should therefore be guided by both the expected noise level and the required precision of the application: `ODEPE-GPR (polished)` for maximum accuracy in low-noise scenarios, and `SciML` for reliability when noise is substantial.

5.3 Performance Across System Complexity

To understand how method performance varies across different ODE systems, we examine median worst-parameter errors at both low and high noise levels. Tables 6 and 7 show the performance of the three leading methods across all 11 benchmark systems at 10^{-6} and 10^{-2} noise levels, respectively. These values represent the median of the maximum parameter error per run, consistent with the run-level aggregation used throughout this paper.

Table 6: Median worst-parameter error (%) at low noise (10^{-6}) by system. For each run, the maximum parameter error is computed; the median across runs is reported.

System	ODEPE-GPR (polished)	SciML	AMIGO2 [0,10]
Biohydrogenation	56.2	196.3	94.8
Crauste	58.5	14.4	1.15
DAISY MaMil3	0.25	0.058	0.060
DAISY MaMil4	120.4	45.4	109.3
FitzHugh-Nagumo	0.001	0.001	0.002
Harmonic Oscillator	0.000	0.000	0.000
HIV	0.59	0.032	0.043
Lotka-Volterra	0.15	0.20	0.16
SEIR	11.3	17.9	56.8
Slow-Fast	0.006	0.006	0.016
Van der Pol	0.000	0.000	0.000

Table 7: Median worst-parameter error (%) at high noise (10^{-2}) by system. For each run, the maximum parameter error is computed; the median across runs is reported.

System	ODEPE-GPR (polished)	SciML	AMIGO2 [0,10]
Biohydrogenation	220.6	215.6	> 1000
Crauste	468.7	218.7	906.6
DAISY MaMil3	457.0	128.8	250.1
DAISY MaMil4	776.8	70.3	499.4
FitzHugh-Nagumo	73.7	11.8	10.5
Harmonic Oscillator	0.15	0.19	0.19
HIV	349.9	100.0	577.2
Lotka-Volterra	255.7	113.3	486.2
SEIR	939.0	131.0	419.4
Slow-Fast	23.3	10.8	27.6
Van der Pol	0.37	0.21	0.21

At low noise (Table 6), all three methods perform well on most systems, with errors typically below 1%. However, certain systems like Biohydrogenation and DAISY MaMil4 prove challenging even at low noise. Notably, simple systems like the Harmonic Oscillator, FitzHugh-Nagumo, and Van der Pol achieve near-perfect parameter recovery (errors $< 0.001\%$) across all methods.

At high noise (Table 7), the performance landscape changes dramatically. SciML consistently achieves the lowest errors across most systems, demonstrating superior robustness. The GPR-based methods show significant performance degradation, with many systems exhibiting errors exceeding 100%. Interestingly, the Harmonic Oscillator and Van der Pol systems remain remarkably robust even at high noise, suggesting that system structure plays a crucial role in noise sensitivity.

5.4 Computational Performance

While this paper focuses primarily on accuracy and robustness rather than computational efficiency, we provide runtime statistics for completeness. Table 8 shows the median wall-clock time per run across all systems and noise levels.

Table 8: Median runtime per run across all systems and noise levels. Runtimes represent wall-clock time and are provided for general reference only, as the implementations vary in optimization level and parallelization strategy.

Method	Median Runtime (s)
ODEPE-GPR	448.7
ODEPE-GPR (polished)	625.7
SciML	207.8
AMIGO2 [0,10]	418.0
AMIGO2 [0,100]	377.0

These runtimes should be interpreted with caution, as the implementations vary significantly in their level of optimization and parallelization. The code was not optimized for speed, Julia compilation overhead is included, and threading strategies differ across methods. SciML is the fastest, followed by AMIGO2 and ODEPE-GPR. The polished variant adds moderate overhead for local refinement. For problems where accuracy is paramount and computational resources are available, the additional runtime of ODEPE-GPR (polished) may be justified by its superior precision in low-noise regimes.

6 Discussion

Our results present a compelling case for the viability of GPR-enhanced algebraic parameter estimation. The primary takeaway is that by addressing the critical weakness of the algebraic method—its sensitivity to noise—we have created a tool that is not only robust but also highly competitive with established optimization techniques, all while preserving the unique advantages of the algebraic approach.

6.1 Why GPR Succeeds Where Interpolation Fails

In preliminary experiments with the AAA-based algebraic method (`ODEPE-AAA`), we observed severe sensitivity to noise, with success rates dropping to near zero as soon as any noise was added to the data. This supports our conclusion that direct interpolation of noisy data is a poor paradigm for derivative estimation in this setting. Because interpolants are constrained to pass close to every noisy point, differentiation amplifies noise catastrophically. In contrast, regression-based approaches like GPR explicitly trade off data-fit and smoothness, making them far more suitable for this task.

GPR is a regression method built on a probabilistic foundation. It does not assume that the data points are exact. Instead, it assumes that a smooth, underlying function exists and that the observations are a combination of this function and some measurement noise. By optimizing its hyperparameters, GPR effectively learns to separate the signal from the noise. The resulting posterior mean function is a principled, smoothed representation of the underlying dynamics, whose derivatives are consequently stable and accurate. This fundamental difference in modeling assumptions is the key to our method's success.

6.2 Practical Advantages and Trade-offs

The primary advantage of our GPR-enhanced method over traditional optimization approaches is that it requires no initial parameter guesses or search ranges. This is a significant practical benefit, as the performance of optimizers can be highly sensitive to the choice of starting point, often requiring extensive manual tuning or computationally expensive multi-start strategies to find a good solution. Our method is fully automated with reasonable default settings, removing a major source of user error and manual effort. Users may optionally provide physical constraints (such as positivity) or adjust the fit-quality tolerance if desired, but these are not required for the method to operate successfully.

Furthermore, because our method is rooted in solving algebraic systems, it has the capacity to find *all* possible parameter solutions for locally identifiable systems. This provides a global perspective on the parameter landscape that is simply unavailable to standard local optimizers, which are designed to find only a single minimum. For problems in systems biology and engineering where multiple parameter sets can explain the data, this feature is not just a convenience but a critical tool for correct model analysis.

The main trade-off, as suggested by our results on more complex systems (Tables 6 and 7), appears to be in the scalability of accuracy. While the method is highly reliable (high success rate), the average accuracy for the highly complex Crauste model was lower than that of the fine-tuned optimizer. This may reflect the increased sensitivity of the algebraic method on highly nonlinear systems, where even small errors in derivative estimates can propagate significantly through the polynomial system. However, even in these cases, our method provides a highly reliable way to find a good-quality solution automatically. Such a solution could then serve as an excellent, data-driven starting point for a local optimizer if maximum precision is the ultimate goal, combining the strengths of both approaches.

6.3 Limitations and Future Work

The current method has two main limitations that point toward avenues for future research. First, like all two-stage methods, it requires data that is sufficiently dense to allow for a reliable GPR fit. The performance on very sparse datasets has not been evaluated, and developing strategies to handle such data is an important next step. This could involve using priors informed by the ODE structure within the GPR model itself.

Second, the computational cost of the polynomial solving step scales with the number of parameters and states. While it performs well on the moderately sized problems in our benchmark, its applicability to extremely large-scale systems (e.g., ¿50 parameters) would require further investigation into the performance of the underlying algebraic solvers. Integrating more advanced, high-performance polynomial solvers could significantly extend the scale of problems our method can tackle.

7 Related Work

Our work is positioned at the intersection of several fields: parameter estimation for dynamical systems, differential algebra, and machine learning for scientific computing.

Traditional parameter estimation has long been dominated by optimization-based “shooting” methods, both local and global, as exemplified by the `SciML` and `AMIGO2` baselines used in this study. A thorough review of these methods can be found in works by Raue et al. [11] and Villaverde and Banga [26].

The differential-algebraic approach to structural identifiability and parameter estimation has a rich theoretical history, with key contributions from authors like Ljung and Glad [17]. Our work builds directly on the algorithmic framework presented in our previous paper [3], which in turn relies on the identifiability analysis tools developed by Hong et al. [14].

The use of Gaussian Processes in the context of differential equations is an active area of research. Some works focus on directly solving ODEs with GP-based methods, while others use GPs for parameter inference, often in a full Bayesian framework. Key papers in this area include the work of Calderhead et al. [7] and Girolami [10], which laid the groundwork for using GPs to connect ODE models with data. More recent work includes Bayesian approaches combining GPs with NeuralODEs [5] and variational multiple shooting methods [13]. In contrast to these methods, our approach is not fully Bayesian; we use GPR as a highly effective "black-box" smoother and differentiator within a larger algebraic framework.

The challenge of accurately estimating derivatives from noisy data was systematically addressed in a benchmark study by Bassik (2025) [2]. The study compared multiple numerical differentiation methods—including finite differences, Savitzky–Golay filters, and various splines—across thousands of test cases with diverse functions, noise levels ranging from 10^{-6} to 10^{-2}, and derivatives up to the 7th order. Gaussian Process Regression (GPR) with automatic differentiation of its posterior mean consistently emerged as the most robust approach. It achieved substantially lower median relative errors than competing methods, with a particularly strong advantage for higher-order derivatives where noise amplification is most severe. This superior performance, rooted in GPR's probabilistic noise modeling and automated hyperparameter tuning, provides direct empirical justification for our selection of GPR as the core differentiation engine in our parameter estimation framework.

Recently, Demin et al. [8] proposed a certified approach to parameter estimation using polynomial system solving, offering rigorous guarantees on solution accuracy. While their method provides theoretical certainty, our GPR-enhanced approach offers a practical balance between accuracy and computational efficiency for real-world noisy data.

8 Conclusion

We have presented a robust, automated method for parameter estimation in rational ODE systems that successfully bridges the gap between the theoretical power of differential algebra and the practical reality of noisy experimental data. By replacing the brittle interpolation step of the original algebraic method with a principled Gaussian Process Regression, we have created a method that is highly reliable, competitive in accuracy with standard optimization techniques, and retains the crucial advantages of being guess-free and capable of finding all solutions.

Our comprehensive benchmark demonstrates that the GPR-enhanced approach is a viable and powerful tool for parameter estimation, particularly in the low-to-moderate noise regimes common in many scientific and engineering disciplines. It transforms the algebraic method from a theoretical curiosity into a practical instrument for systems analysis. Future work will focus on extending the method to handle sparse data and investigating its scalability to very-large-scale models.

References

[1] Eva Balsa-Canto, David Henriques, Attila Gabor, and Julio R. Banga. AMIGO2, a toolbox for dynamic modeling, simulation and optimization in systems biology. *Bioinformatics*, 32(21):3357–3359, 07 2016.

[2] Oren Bassik. Benchmark study of derivative estimation methods: Performance across orders and noise levels. 2025. In preparation.

[3] Oren Bassik, Yosef Berman, Soo Go, Hoon Hong, Ilia Ilmer, Alexey Ovchinnikov, Chris Rackauckas, Pedro Soto, and Chee Yap. Robust parameter estimation for rational ordinary differential equations. *Applied Mathematics and Computation*, 509:129638, 2026.

[4] G. Bellu, M. P. Saccomani, S. Audoly, and L. D'Angiò. DAISY: a new software tool to test global identifiability of biological and physiological systems. *Computer Methods and Programs in Biomedicine*, 88(1):52–61, 10 2007.

[5] Mohamed Aziz Bhouri and Paris Perdikaris. Gaussian processes meet NeuralODEs: A Bayesian framework for learning the dynamics of partially observed systems from scarce and noisy data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 380(2229), 2022. Also available as arXiv preprint arXiv:2103.03385 (2021).

[6] Paul Breiding and Sascha Timme. HomotopyContinuation.jl: A Package for Homotopy Continuation in Julia. In *Mathematical Software – ICMS 2018*, volume 10931 of *Lecture Notes in Computer Science*, pages 458–465. Springer, 2018.

[7] Ben Calderhead, Mark A. Girolami, and Neil D. Lawrence. Accelerating bayesian inference over nonlinear differential equations with gaussian processes. In *Advances in Neural Information Processing Systems 21*, pages 217–224, 2009.

[8] Alexander Demin, Alexey Ovchinnikov, and Fabrice Rouillier. Parameter estimation in ODE models with certified polynomial system solving. *arXiv preprint arXiv:2504.17268*, 4 2025.

[9] Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1(6):445–466, 7 1961.

[10] Mark A. Girolami. Bayesian inference for differential equations. *Theoretical Computer Science*, 408(1):4–16, 2008.

[11] Attila Gábor and Julio R. Banga. Robust and efficient parameter estimation in dynamic models of biological systems. *BMC Systems Biology*, 9(74), 10 2015.

[12] K. J. Harvatine and M. S. Allen. Kinetic model of rumen biohydrogenation: fractional rates of fatty acid biohydrogenation and passage. *Journal of Animal and Feed Sciences*, 13(Suppl. 1):87–90, 2004.

[13] Pashupati Hegde, Çağatay Yıldız, Harri Lähdesmäki, Samuel Kaski, and Markus Heinonen. Variational multiple shooting for Bayesian ODEs with Gaussian processes. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pages 790–799. PMLR, 2022.

[14] Hoon Hong, Alexey Ovchinnikov, Gleb Pogudin, and Chee Yap. Sian: A software for structural identifiability analysis of ode models. In *Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation*, ISSAC '19, page 231–238, New York, NY, USA, 2019. Association for Computing Machinery.

[15] J. R. Jacobs, S. L. Shafer, J. L. Larsen, and E. D. Hawkins. Two equally valid interpretations of the linear multicompartment mammillary pharmacokinetic model. *Journal of Pharmaceutical Sciences*, 79(4):331–333, 4 1990.

[16] William Ogilvy Kermack and Anderson G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115(772):700–721, 8 1927.

[17] L. Ljung and T. Glad. On global identifiability for arbitrary model parameterizations. *Automatica*, 30(2):265–276, 1994.

[18] Alfred J. Lotka. *Elements of Physical Biology.* Williams & Wilkins, Baltimore, MD, 1925.

[19] J. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 10 1962.

[20] A. S. Perelson, D. E. Kirschner, and R. De Boer. Dynamics of HIV infection of CD4+ T cells. *Mathematical Biosciences*, 114(1):81–125, 3 1993.

[21] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[22] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, Cambridge, MA, 2006.

[23] Andreas Raue, Clemens Kreutz, Thomas Maiwald, Jörg Bachmann, Marcel Schilling, Ursula Klingmüller, and Jens Timmer. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, 25(15):1923–1929, 8 2009.

[24] Emmanuelle Terry, Jacqueline Marvel, Christophe Arpin, Olivier Gandrillon, and Fabien Crauste. Mathematical model of the primary CD8 T cell immune response: stability analysis of a nonlinear age-structured system. *Journal of Mathematical Biology*, 65(2):263–291, 8 2012. Epub 2011 Aug 13.

[25] B. van der Pol and J. van der Mark. Frequency demultiplication. *Nature*, 120(3019):363–364, 9 1927.

[26] Alejandro F. Villaverde and Julio R. Banga. Reverse engineering and identification in systems biology: strategies, perspectives and challenges. *Journal of the Royal Society Interface*, 11(91):20130505, 2 2014.

[27] Vito Volterra. Variations and fluctuations of the number of individuals in animal species living together. *ICES Journal of Marine Science*, 3(1):3–51, 4 1928. Translation of Volterra's 1926 work.

A Benchmark Model Details

This appendix presents the ODE systems used in the benchmarks, adapted from [3].

A.1 System Specifications and Observed Outputs

Table 9 provides the detailed specifications for each benchmark system, including the number of state variables, parameters, and which outputs were treated as observable in our experiments. This information is essential for understanding the observational scenario and partial observability constraints of each system.

Table 9: Observed outputs for each benchmark system. The measurement variables column lists the specific outputs that were treated as observable in the benchmark experiments.

System	# States	# Parameters	Observed Outputs
Biohydrogenation	4	6	$y1$, $y2$
Crauste	5	13	$y1$, $y2$, $y3$, $y4$
DAISY MaMil3	3	5	$y1$, $y2$
DAISY MaMil4	4	7	$y1$, $y2$, $y3$
FitzHugh-Nagumo	2	3	$y1$
Harmonic Oscillator	2	2	$y1$, $y2$
HIV	5	10	$y1$, $y2$, $y3$, $y4$
Lotka-Volterra	2	3	$y1$
SEIR	4	3	$y1$, $y2$
Van der Pol	2	2	$y1$, $y2$
Slow-Fast	5	3	$y1$, $y2$, $y3$, $y4$

A.2 Harmonic Oscillator Model

A model for harmonic oscillators without damping.

$$\begin{cases} \dot{x_1} = -a\, x_2 \\ \dot{x_2} = \frac{1}{b}\, x_1 \end{cases} \tag{4}$$

Outputs: $y_1 = x_1, y_2 = x_2$.

A.3 Van der Pol Oscillator Model

A classical nonlinear oscillator from electrical circuit theory [25].

$$\begin{cases} \dot{x_1} = a x_2 \\ \dot{x_2} = -x_1 - b(x_1^2 - 1)x_2 \end{cases} \tag{5}$$

Outputs: $y_1 = x_1, y_2 = x_2$.

A.4 FitzHugh-Nagumo Model

A two-dimensional simplification of spike generation in squid giant axons [9, 19].

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

Output: $y_1 = V$.

A.5 HIV Dynamics Model

Models HIV infection dynamics during interaction with the immune system [20].

$$\begin{cases} \dot{x} = \lambda - d\,x - \beta\,x\,v \\ \dot{y} = \beta\,x\,v - a\,y \\ \dot{v} = k\,y - u\,v \\ \dot{w} = c\,x\,y\,w - c\,q\,y\,w - b\,w \\ \dot{z} = c\,q\,y\,w - h\,z \end{cases} \tag{7}$$

Outputs: $y_1 = w, y_2 = z, y_3 = x, y_4 = y + v$.

A.6 Mammillary 3-Compartment Model

A 3-compartment pharmacokinetic model [15] from the DAISY identifiability software examples [4].

$$\begin{cases} \dot{x_1} = -(a_{21} + a_{31} + a_{01})\,x_1 + a_{12}\,x_2 + a_{13}\,x_3 \\ \dot{x_2} = a_{21}\,x_1 - a_{12}\,x_2 \\ \dot{x_3} = a_{31}\,x_1 - a_{13}\,x_3 \end{cases} \tag{8}$$

Outputs: $y_1 = x_1, y_2 = x_2$.

A.7 Lotka-Volterra Model

Models predator-prey interactions in an ecosystem [18, 27].

$$\begin{cases} \dot{r} = k_1\,r - k_2\,r\,w, \\ \dot{w} = k_2\,r\,w - k_3\,w \end{cases} \tag{9}$$

Output: $y_1 = r$.

A.8 Crauste Model

Models the behavior of CD8 T-cells [24].

$$\begin{cases} \dot{N} = -\mu_N\,N - \delta_{NE}\,N\,P \\ \dot{E} = \delta_{NE}\,N\,P - \mu_{EE}\,E^2 - \delta_{EL}\,E + \rho_E\,E\,P \\ \dot{S} = \delta_{EL}\,S - S\,\delta_{LM} - \mu_{LL}\,S^2 - \mu_{LE}\,E\,S \\ \dot{M} = \delta_{LM}\,S - \mu_M\,M \\ \dot{P} = \rho_P\,P^2 - \mu_P\,P - \mu_{PE}\,E\,P - \mu_{PL}\,S\,P \end{cases} \tag{10}$$

Outputs: $y_1 = N, y_2 = E, y_3 = S + M, y_4 = P$.

A.9 Biohydrogenation Model

Models kinetic processes in the biohydrogenation of fatty acids [12].

$$\begin{cases} \dot{x_4} = -\frac{k5\,x_4}{k_6+x_4} \\ \dot{x_5} = \frac{k_5\,x_4}{k_6+x_4} - \frac{k_7\,x_5}{k_8+x_5+x_6} \\ \dot{x_6} = \frac{k_7\,x_5}{k_8+x_5+x_6} - \frac{k_9\,x_6\,(k_{10}-x_6)}{k_{10}} \\ \dot{x_7} = \frac{k_9\,x_6\,(k_{10}-x_6)}{k_{10}} \end{cases} \tag{11}$$

Outputs: $y_1 = x_4, y_2 = x_5$.

Note: The state variable x_7 is structurally unidentifiable as it does not appear in the output equations nor influence any observed variables.

A.10 Mammillary 4-Compartment Model

A 4-compartment pharmacokinetic model [15] from the DAISY identifiability software examples [4].

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

Outputs: $y_1 = x_1, y_2 = x_2, y_3 = x_3 + x_4$.

A.11 SEIR Model

Models an epidemic with stages of disease progression [16].

$$\begin{cases} \dot{S} = -b\, S\, I/N \\ \dot{E} = b\, S\, I/N - \nu\, E \\ \dot{I} = \nu\, E - a\, I \\ \dot{N} = 0 \end{cases} \tag{13}$$

Outputs: $y_1 = I, y_2 = N$.