

# Robust Algebraic Parameter Estimation via Gaussian Process Regression

---

Oren Bassik<sup>1</sup>   Alexander Demin<sup>2</sup>   Alexey Ovchinnikov<sup>1,3</sup>

IFAC/COSY/SSSC Joint Conference

July 1, 2025

<sup>1</sup>CUNY Graduate Center

<sup>2</sup>HSE University

<sup>3</sup>CUNY Queens College

# Parameter Estimation for ODE Systems

## Nonlinear Optimization Methods

E.g., Levenberg-Marquardt, Gradient Descent

- **Pros:** Mature, widely used, can be highly accurate, may scale well to large systems, noise is handled naturally.
- **Cons:** Require good initial guesses, risk convergence to local minima, typically find only one solution, often require manual tuning.

## Differential Algebra Methods

Based on symbolic manipulation of system equations.

- **Pros:** No initial guesses required, can find all solutions (given local identifiability), fully automated, integrated with identifiability analysis.
- **Cons:** Historically very sensitive to measurement noise, scaling is a challenge.

## Research Question

Can we make the algebraic method robust enough for real-world noisy data,

# The Differential-Algebraic Method in a Nutshell

1. **Differentiate System Symbolically:** Relate parameters to higher-order derivatives of observable outputs  $(y, \dot{y}, \ddot{y}, \dots)$ .

# The Differential-Algebraic Method in a Nutshell

1. **Differentiate System Symbolically:** Relate parameters to higher-order derivatives of observable outputs  $(y, \dot{y}, \ddot{y}, \dots)$ .
2. **Approximate Derivatives from Data:** At a specific time point  $t_i$ , compute numerical values for  $y(t_i), \dot{y}(t_i), \ddot{y}(t_i), \dots$  from measurements.

# The Differential-Algebraic Method in a Nutshell

1. **Differentiate System Symbolically:** Relate parameters to higher-order derivatives of observable outputs  $(y, \dot{y}, \ddot{y}, \dots)$ .
2. **Approximate Derivatives from Data:** At a specific time point  $t_i$ , compute numerical values for  $y(t_i), \dot{y}(t_i), \ddot{y}(t_i), \dots$  from measurements.
3. **Form Polynomial System:** Combine the symbolic relations from (1) with numerical values from (2).

# The Differential-Algebraic Method in a Nutshell

1. **Differentiate System Symbolically:** Relate parameters to higher-order derivatives of observable outputs  $(y, \dot{y}, \ddot{y}, \dots)$ .
2. **Approximate Derivatives from Data:** At a specific time point  $t_i$ , compute numerical values for  $y(t_i), \dot{y}(t_i), \ddot{y}(t_i), \dots$  from measurements.
3. **Form Polynomial System:** Combine the symbolic relations from (1) with numerical values from (2).
4. **Solve:** Use a numerical polynomial solver to find all sets of solutions for the parameters.

# The Differential-Algebraic Method in a Nutshell

1. **Differentiate System Symbolically:** Relate parameters to higher-order derivatives of observable outputs  $(y, \dot{y}, \ddot{y}, \dots)$ .
2. **Approximate Derivatives from Data:** At a specific time point  $t_i$ , compute numerical values for  $y(t_i), \dot{y}(t_i), \ddot{y}(t_i), \dots$  from measurements.
3. **Form Polynomial System:** Combine the symbolic relations from (1) with numerical values from (2).
4. **Solve:** Use a numerical polynomial solver to find all sets of solutions for the parameters.
5. **Filter & Validate:** Use forward simulation to find the best-fitting parameter set(s).

The practical success of this method hinges on two challenging steps.

# Two Critical Bottlenecks

The practical success of the algebraic method hinges on two challenging steps:

## Challenge 1: Derivative Estimation

- **Problem:** Must be accurate, and also robust to measurement noise. Naive methods amplify noise.
- **Our Focus:** This is the primary challenge addressed in this work.

## Challenge 2: Polynomial Solving

- **Problem:** Must be efficient and stable for large, complex systems.
- **Status:** We currently use Gröbner basis methods, i.e. RUR, and pivot to homotopy continuation for large systems.

## Our Contribution

We solve the derivative estimation bottleneck using Gaussian Process Regression.



# The Challenge: Differentiating Noisy Data

## Problem: Interpolation Overfits Noise

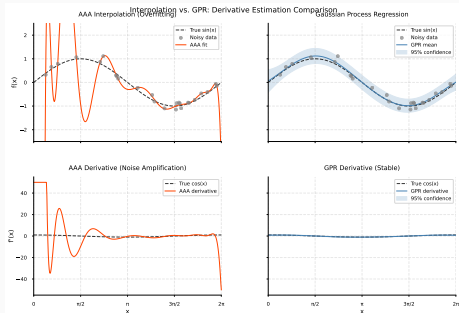
Methods like polynomial or rational interpolation (e.g., AAA) are designed to pass *through* data points. With noisy data, this leads to severe oscillations and unusable derivative estimates.

- Function RMSE: 135.38
- Derivative RMSE: 1278.42

## Solution: Probabilistic Regression

We need a method that smooths the data by modeling the underlying function and explicitly accounting for noise. This is a regression problem, not an interpolation problem.

- Function RMSE: 0.059
- Derivative RMSE: 0.082



*Figure included in left panel.*

# Our Contribution: Gaussian Process Regression

## Why Gaussian Process Regression (GPR)?

Instead of fitting a single function, GPR defines a *prior distribution over functions* and updates it to a *posterior distribution* based on the data.

- **Principled Smoothing:** A smoothness assumption is encoded in the prior via a kernel function (e.g., RBF). This is a natural fit for physical systems.
- **Noise Modeling:** GPR explicitly models measurement noise ( $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ ), learning the noise level from the data itself.
- **Analytic Derivatives:** The posterior mean function is a smooth, infinitely differentiable function whose derivatives can be computed analytically.

## Key Idea

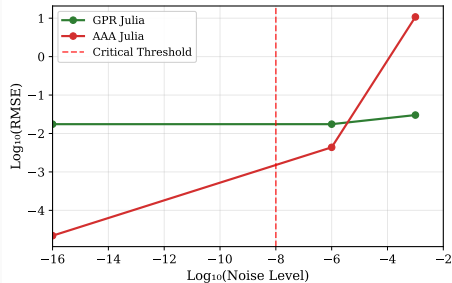
GPR replaces brittle interpolation with robust Bayesian inference.

# Differentiator Benchmark Results

## Noise Sensitivity

### RMSE vs. Noise Level

The "Noise Cliff"  
RMSE vs. Noise Level (1st Derivative)



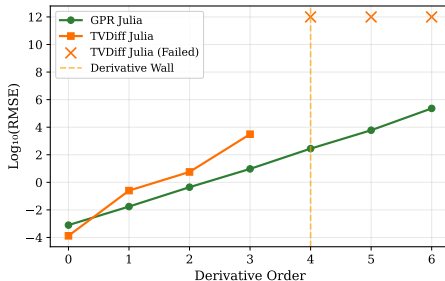
Analysis

- **GPR:** Error scales gracefully with noise.

## Higher-Order Derivatives

### RMSE vs. Derivative Order

The "Derivative Wall"  
RMSE vs. Derivative Order ( $\sigma = 1e-06$ )



Analysis

- **GPR:** Graceful degradation in accuracy.

# Application: End-to-End Parameter Estimation

## Benchmark on Nonlinear Systems (1% Relative Noise)

System	Original Alg. (AAA)	GPR-Algebraic	SciML (Optimizer)
Fitzhugh-Nagumo	100%	1.8%	0.5%
Lotka-Volterra	100%	2.1%	0.3%
SEIR Model	FAIL	1.2%	0.4%

*Values are Mean Relative Error (MRE) in parameter estimates. SciML is a standard Levenberg-Marquardt optimizer.*

## Interpretation

- The original algebraic method (using AAA) is unusable with noisy data.
- The GPR-enhanced method yields accurate results

## Key Advantage

Our method achieves this accuracy **without requiring initial guesses** for the parameters.

## Conclusion

- We addressed the key bottleneck of the differential-algebraic parameter estimation method by replacing interpolation with Gaussian Process Regression.
- The GPR-enhanced method is robust to realistic noise levels, making it a viable tool for practical applications.
- It preserves the essential advantages of the algebraic approach: it is fully automated and requires no initial parameter guesses.

## Future Work

- Propagate GPR uncertainty to get confidence intervals on parameters.
- Use uncertainty for optimal experimental design.
- Explore sparse GPs to improve computational scaling for large datasets.

# Thank You

## Questions?

# A1: Benchmark Results - Function Values ( $y$ )

## RMSE for Derivative Order 0

Method	Noise = 0	Noise = 1e-6	Noise = 1e-3
AAA_Julia	$6.2e - 7$	$1.5e - 4$	0.14
Butterworth_Python	0.81	0.81	0.85
Chebyshev_Python	0.12	0.12	0.16
FiniteDiff_Python	0.00	$1.3e - 4$	0.13
<b>GPR_Julia</b>	$7.8e - 4$	$7.9e - 4$	$0.02$
GP_RBF_Iso_Python	$1.3e - 3$	$1.4e - 3$	$0.01$
KalmanGrad_Python	0.19	0.15	0.22
LOESS_Julia	$0.09$	$0.09$	0.14
SVR_Python	0.56	0.56	0.58
SavitzkyGolay_Python	$0.05$	$0.05$	$0.09$
TVDiff_Julia	$2.5e - 14$	$1.3e - 4$	0.13

## Interpretation

For function value estimation (order 0), most methods perform reasonably well. Interpolation methods like AAA are extremely accurate on noise-free data, but GPR is more stable as noise increases.

## A2: Benchmark Results - 3rd Derivative ( $y'''$ )

### RMSE for Derivative Order 3

Method	Noise = 0	Noise = 1e-6	Noise = 1e-3
AAA_Julia	0.53	2.2e5	1.3e9
Butterworth_Python	28.90	28.90	28.90
Chebyshev_Python	60.56	8.9e2	1.8e5
FiniteDiff_Python	45.19	1.9e4	2.0e7
<b>GPR_Julia</b>	9.44	9.45	19.93
GP_RBF_Iso_Python	16.79	16.78	13.84
KalmanGrad_Python	37.80	27.62	34.38
LOESS_Julia	8.1e4	5.2e8	1.1e4
SVR_Python	28.58	28.58	28.58
SavitzkyGolay_Python	22.97	1.8e4	2.2e7
TVDiff_Julia	1.5e2	3.1e3	6.5e3

### Interpretation

For 3rd order derivatives, the difference is stark. GPR is the only tested method that remains robust and accurate across all noise levels. Other methods either fail catastrophically or, like SVR, have large but stable errors.