Robust Algebraic Parameter Estimation via Gaussian Process Regression

Oren Bassik[1] Alexander Demin[2] Alexey Ovchinnikov[1,3]

IFAC/COSY/SSSC Joint Conference
July 2, 2025

[1]CUNY Graduate Center
[2]HSE University
[3]CUNY Queens College

Supported by NSF grants CCF-2212460 and DMS-1853650

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, thereby combining the advantages of both approaches?

Supported by NSF grants CCF-2212460 and DMS-1853650

The Differential-Algebraic Method in a Nutshell

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

Supported by NSF grants CCF-2212460 and DMS-1853650

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.

Supported by NSF grants CCF-2212460 and DMS-1853650

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).

Supported by NSF grants CCF-2212460 and DMS-1853650

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.

Supported by NSF grants CCF-2212460 and DMS-1853650

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.

Supported by NSF grants CCF-2212460 and DMS-1853650

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.

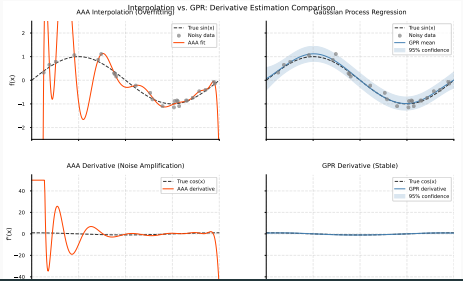
Supported by NSF grants CCF-2212460 and DMS-1853650

The Challenge: Differentiating Noisy Data

Problem: Interpolation Overfits Noise

Methods like polynomial or rational interpolation (e.g., AAA) pass *through* data points, causing severe oscillations with noisy data. Furthermore, many common methods (e.g., finite differences, local polynomials) struggle to produce the stable, higher-order derivatives required by the algebraic approach.

- 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.

Supported by NSF grants CCF-2212460 and DMS-1853650

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 could be computed analytically. In practice, we are using automatic differentiation.
- **Noiseless Performance:** In the absence of noise, GPR performance is competitive with high-accuracy interpolation methods, ensuring no significant trade-off.

Key Idea

GPR replaces brittle interpolation with robust Bayesian inference.

Supported by NSF grants CCF-2212460 and DMS-1853650

Application: End-to-End Parameter Estimation

Benchmark on Nonlinear Systems

	Lotka-Volterra			Van der Pol		
Noise	**GPR**	**AAA**	**SciML**	**GPR**	**AAA**	**SciML**
0.0%	**0.0%**	0.0%	9.1%	**0.0%**	0.0%	7.6%
1.0%	**4.4%**	29.0%	7.7%	**0.8%**	7.8%	13.5%
5.0%	**13.7%**	$> 229\%$	4.1%	**1.3%**	64.2%	6.7%

Values are Mean Relative Error (MRE) in parameter estimates. 'AAA' is the original algebraic method, 'SciML' is least-squares direct optimization driven by primarily by Optimization.jl.

Interpretation

- At zero noise, methods are comparable.
- As noise increases, the AAA-based algebraic method fails catastrophically.

Key Advantage

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

Supported by NSF grants CCF-2212460 and DMS-1853650

Conclusion & Future Work

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 derivative uncertainty to parameter estimates.
- Develop "physics-informed" priors for GPR using the ODE structure.
- Explore sparse GPs to improve computational scaling for large datasets.

Software: `github.com/orebas/ODEParameterEstimation`

Supported by NSF grants CCF-2212460 and DMS-1853650

Thank You

Questions?

Supported by NSF grants CCF-2212460 and DMS-1853650

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
GPR_Julia	$7.8e-4$	$7.9e-4$	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.

Supported by NSF grants CCF-2212460 and DMS-1853650

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
GPR_Julia	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.

Supported by NSF grants CCF-2212460 and DMS-1853650