ODEParameterEstimation.jl: Robust Parameter Estimation for ODEs

Oren Bassik

July 22, 2025

JuliaCon 2025

The Big Picture: What is ODEParameterEstimation.jl?

Many real-world systems are described by ODEs, but we often don't know the parameters.

`ODEParameterEstimation.jl` is a toolbox for discovering these unknown parameters directly from data.

Inputs:

1. ODE Model: $\mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{p})$
2. Noisy Data: $\{y(t_i)\}$

$\downarrow$ `ODEPE.jl`

Output:

Parameter Values $\mathbf{p}$

A Walkthrough of the Method (1/2)

Example: $x' = a^2x^2 + b$, **with output** $y = x^2 + x$

1. **Differentiate System Symbolically:**

A Walkthrough of the Method (1/2)

Example: $x' = a^2x^2 + b$**, with output** $y = x^2 + x$

1. **Differentiate System Symbolically:** Relate parameters to higher-order derivatives of outputs $(y, y', y'', \dots)$.

$$\begin{aligned} y' &= (2x+1)x' \\ y'' &= 2(x')^2 + (2x+1)x'' \end{aligned} \quad \text{where} \quad \begin{aligned} x' &= a^2x^2 + b \\ x'' &= 2a^2xx' \end{aligned}$$

2. **Approximate Derivatives from Data:**

A Walkthrough of the Method (1/2)

Example: $x' = a^2x^2 + b$**, with output** $y = x^2 + x$

1. **Differentiate System Symbolically:**Relate parameters to higher-order derivatives of outputs $(y, y', y'', \ldots)$.

$$\begin{aligned} y' &= (2x+1)x' \\ y'' &= 2(x')^2 + (2x+1)x'' \end{aligned} \quad \text{where} \quad \begin{aligned} x' &= a^2x^2 + b \\ x'' &= 2a^2xx' \end{aligned}$$

2. **Approximate Derivatives from Data:**At a time t_i, compute numerical values for $y(t_i), y'(t_i), \ldots$ from measurements. This is the critical step where GPR is used.

A Walkthrough of the Method (2/2)

Example: $x' = a^2x^2 + b$, **with output** $y = x^2 + x$

3. **Form Polynomial System:**

A Walkthrough of the Method (2/2)

Example: $x' = a^2x^2 + b$**, with output** $y = x^2 + x$

3. **Form Polynomial System:**Substitute numerical derivative values into the symbolic equations to create a polynomial system in the parameters and states.
4. **Solve:**

A Walkthrough of the Method (2/2)

Example: $x' = a^2x^2 + b$**, with output** $y = x^2 + x$

3. **Form Polynomial System:**Substitute numerical derivative values into the symbolic equations to create a polynomial system in the parameters and states.
4. **Solve:**Use a numerical polynomial solver to find all sets of solutions for the parameters (a, b) and states $(x(t_i))$.
5. **Filter & Validate:**

A Walkthrough of the Method (2/2)

Example: $x' = a^2x^2 + b$, **with output** $y = x^2 + x$

3. **Form Polynomial System:**Substitute numerical derivative values into the symbolic equations to create a polynomial system in the parameters and states.
4. **Solve:**Use a numerical polynomial solver to find all sets of solutions for the parameters (a, b) and states $(x(t_i))$.
5. **Filter & Validate:**Use forward simulation to find the best-fitting parameter set.

The Challenge: Real-World Data is Noisy

The Problem

The original method, based on AAA baryrational interpolation, is extremely accurate on clean data but fails catastrophically with noise. Interpolation overfits, leading to unstable derivatives.

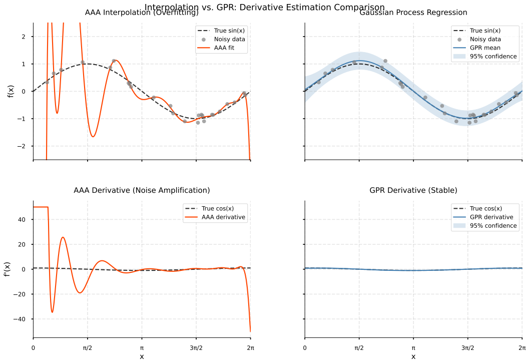

Figure 1: Interpolation vs. Regression

The Solution: Taming Noise with Gaussian Process Regression

Why Gaussian Process Regression (GPR)?
GPR defines a *prior distribution over functions* and updates it based on the data.

- **Principled Smoothing:** A smoothness assumption is encoded in the prior via a kernel function (e.g., RBF).
- **Noise Modeling:** GPR explicitly models measurement noise, learning the noise level from the data itself.
- **Analytic Derivatives:** The posterior mean function is smooth and can be differentiated reliably.

The Result

Robust and accurate derivative estimation, even with noisy data.

The Results: Robust and Accurate

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%

Key Takeaway

The GPR-enhanced method is robust to noise, making it practical for real-world applications.

Why Julia?

Julia is the perfect language for this kind of work!

- **Symbolic Manipulation:** Symbolics.jl
- **Numerical Methods:** DifferentialEquations.jl, GaussianProcesses.jl, Groebner.jl, HomotopyContinuation.jl
- **High Performance:** Fast enough for complex systems.
- **Composable:** Easy to combine different tools and packages.

Get Involved!

`github.com/orebas/ODEParameterEstimation`

(Formerly known as ParameterEstimation.jl)

Oren Bassik
obassik@gradcenter.cuny.edu

Thank You

Questions?