# Benchmark Study of Derivative Estimation Methods: Performance Across Orders and Noise Levels

Author Names[*]

October 24, 2025

**Abstract**

Derivative estimation from noisy data is a fundamental problem across computational science, yet systematic comparative evaluation of methods remains lacking. We present a comprehensive benchmark of 24 derivative estimation methods evaluated across 8 derivative orders (0–7) and 7 noise levels ($10^{-8}$ to $5 \times 10^{-2}$), providing the first fine-grained performance map spanning 56 configurations. Methods span six categories: Gaussian Processes, Rational Approximation, Spectral Methods, Splines, Regularization, and Local Polynomial approaches.

Key findings include: (1) Gaussian Process regression (GP-Julia-AD) achieves best overall performance (mean nRMSE = 0.257) with robust behavior across all orders and noise levels; (2) AAA rational approximation catastrophically fails at orders $\geq 3$ (nRMSE $> 10^7$) despite excellent low-order performance—contrary to literature expectations; (3) Fourier spectral methods provide competitive accuracy at 10–20$\times$ speedup for smooth signals; (4) derivative order is the dominant difficulty factor, with performance degrading systematically as order increases; (5) implementation quality is a critical method characteristic—three methods excluded due to $> 50\times$ cross-language discrepancies.

We provide practical recommendations including a master method-selection table, decision framework, and common pitfalls to avoid. Results are derived from a single test system (Lotka-Volterra) with additive Gaussian noise and n=3 trials, providing exploratory evidence rather than definitive statistical rankings. Practitioners should validate candidate methods on their specific data before deployment.

## Contents

[*]Department/Institution, email@domain.edu

## 1 Introduction

Derivative estimation from noisy data is a fundamental problem spanning computational science, engineering, and data analysis. Applications include:

- **Dynamical systems identification:** Inferring governing equations from time-series data [?]—requires accurate derivative estimates to identify differential equations
- **Signal processing:** Edge detection, feature extraction, and change-point analysis—depend on robust derivative computation
- **Control systems:** Model-predictive control and adaptive control—rely on real-time derivative estimation for system state feedback
- **Physics-informed machine learning:** Enforcing physical conservation laws (PDE constraints) requires differentiating neural network outputs with respect to noisy inputs
- **Scientific data analysis:** Estimating rates of change from experimental measurements (e.g., reaction rates from concentration data, acceleration from position measurements)

Despite this widespread importance, derivative estimation is notoriously *ill-posed*: small noise perturbations in the input signal can produce arbitrarily large errors in derivative estimates, particularly for high-order derivatives. This ill-posedness intensifies exponentially with derivative order—estimating seventh-order derivatives from noisy data represents an extreme challenge even for state-of-the-art methods.

### 1.1 The Need for Comprehensive Benchmarking

Numerous derivative estimation methods have been proposed across disciplines, including:

- Finite difference schemes (low computational cost, severe noise amplification)
- Polynomial and spline fitting (moderate accuracy, parameter-sensitive)
- Gaussian Process regression (theoretically optimal under Gaussian assumptions, computationally expensive)
- Spectral methods (fast for smooth signals, Gibbs phenomenon for discontinuities)
- Total variation regularization (edge-preserving, limited to low-order derivatives)
- Rational approximation (high accuracy for analytic functions, potential instability)

However, **systematic comparative evaluation across derivative orders, noise levels, and method categories is lacking**. Prior studies suffer from:

1. **Limited scope:** Benchmarks typically evaluate 3–5 methods on orders 0–2 only, neglecting high-order derivatives where method performance diverges dramatically
2. **Single-noise regime:** Most studies test either noiseless or high-noise cases, missing performance transitions across noise levels
3. **Implementation inconsistencies:** Cross-language comparisons without rigorous parameter parity, leading to implementation artifacts dominating algorithmic differences

4. **Incomplete coverage transparency:** Methods that fail at high orders or noise levels are often excluded without documentation, biasing overall rankings

**Practitioners face a dilemma:** Which method should I use for *my* specific derivative order and noise level? Existing literature provides insufficient guidance.

## 1.2 Contributions of This Study

This paper presents a **comprehensive benchmark of 33 derivative estimation methods** (24 baseline methods[1] plus 9 adaptive hyperparameter selection variants) evaluated across:

- **8 derivative orders:** 0–7 (order 0 = smoothing, testing full capability range)
- **7 noise levels:** $10^{-8}$ to $5 \times 10^{-2}$ (near-noiseless to high-noise regimes, spanning $\approx 7$ orders of magnitude)
- **56 configurations:** All combinations of order $\times$ noise, providing fine-grained performance maps
- **7 method categories:** Gaussian Processes, Rational Approximation, Spectral Methods, Splines, Regularization, Local Polynomial methods, and Adaptive Hyperparameter Selection

**Key contributions include:**

1. **Rigorous experimental design:**
   - High-accuracy ground truth via symbolic differentiation + augmented ODE system (validated to $10^{-10}$ accuracy)
   - Normalized RMSE metric enabling fair comparison across derivative orders
   - Documented coverage bias: only 16/24 methods achieve full 56/56 configuration coverage
2. **Unexpected findings:**
   - AAA rational approximation *catastrophically fails* at orders $\geq 3$ (nRMSE $> 10^7$) despite excellent low-order performance—contrary to literature expectations
   - Gaussian Process regression (GP-Julia-AD) achieves best overall performance but 3 Python GP implementations fail or underperform dramatically, highlighting implementation quality as a critical method characteristic
   - Adaptive hyperparameter selection methods (using GCV, AICc, and noise estimation) significantly outperform their fixed-parameter counterparts, particularly at high derivative orders
   - JAX-based automatic differentiation enables arbitrary-order derivatives for AAA rational approximants, though fundamental high-order instability persists
   - Derivative order is the dominant difficulty factor, with performance degrading systematically for all methods as order increases
3. **Practical recommendations:**

[1]From 27 candidate baseline implementations; 3 were excluded due to severe cross-language performance discrepancies exceeding 50× despite parameter parity attempts (documented in Section **??**).

- Master recommendation table mapping (derivative order, noise level) $\rightarrow$ optimal method(s)
- Decision framework for method selection based on accuracy requirements, computational budget, and data characteristics
- Common pitfalls and implementation guidance to avoid catastrophic failures

4. **Transparency and reproducibility:**

   - Explicit documentation of method exclusions (3/27 candidates due to cross-language discrepancies exceeding $50\times$)
   - Statistical limitations acknowledged (n=3 trials insufficient for hypothesis testing; findings are descriptive, not definitive)
   - Generalization caveats: single test system (Lotka-Volterra), additive Gaussian noise only—results are existence proofs, not universal rankings

## 1.3 Impact and Limitations

**For practitioners:** This study provides the first comprehensive performance map across derivative orders and noise levels, enabling evidence-based method selection. Recommendations are immediately actionable for applications matching our test conditions (smooth signals, additive Gaussian noise).

**For researchers:** Findings reveal fundamental performance limitations (e.g., all methods struggle at orders $\geq 6$ with noise $\geq 10^{-2}$) and unexpected failure modes (AAA at high orders), suggesting directions for algorithmic improvements.

**Critical limitations:**

- **Single test system:** Lotka-Volterra is smooth and periodic—favorable for spectral methods. Rough/discontinuous signals may yield different rankings.
- **Small sample size:** n=3 trials provides only exploratory evidence; specific numerical values should be interpreted cautiously.
- **Generalization uncertainty:** Performance on your data may differ—validation is essential (see Section **??**).

## 1.4 Paper Organization

The remainder of this paper is structured as follows:

- **Section ??:** Reviews prior derivative estimation surveys and benchmarking efforts, identifying gaps this study addresses
- **Section ??:** Formally defines the mathematical problem, evaluation metrics (normalized RMSE), and success criteria
- **Section ??:** Details experimental design, test system (Lotka-Volterra), ground truth generation, and statistical methodology
- **Section ??:** Describes all 24 methods evaluated, organized by category with mathematical formulations

- **Section ??:** Presents performance results across derivative orders and noise levels, including coverage analysis and Pareto-optimal methods
- **Section ??:** Interprets findings, explains unexpected results (GP excellence, AAA failure), and discusses statistical limitations
- **Section ??:** Provides practical method selection guidance, decision framework, and implementation advice
- **Section ??:** Discusses limitations and proposes extensions (multiple test systems, larger sample sizes, additional noise models)
- **Section ??:** Summarizes key findings and impact

## 2 Related Work

**[TODO: Related work section. Should review: (1) existing derivative estimation methods surveys, (2) prior benchmarking efforts (limited scope), (3) gaps this study fills, (4) methodological contributions]**

## 3 Problem Formulation

This section formally defines the derivative estimation problem, evaluation metrics, and success criteria used throughout this benchmark study.

### 3.1 Mathematical Problem Statement

**Given:**

- A smooth function $f : \mathbb{R} \to \mathbb{R}$ (unknown analytically)
- Noisy observations $\{(t_i, y_i)\}_{i=1}^{n}$ where $y_i = f(t_i) + \epsilon_i$
- Noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ (additive Gaussian)
- Uniform grid $t_i \in [t_{\min}, t_{\max}]$ with spacing $h = (t_{\max} - t_{\min})/(n-1)$

**Objective:**
Estimate the $n$-th order derivative $f^{(n)}(t)$ at evaluation points $t \in \mathcal{T} \subset [t_{\min}, t_{\max}]$ such that:

$$\hat{f}^{(n)}(t) \approx f^{(n)}(t) \quad \text{for all } t \in \mathcal{T} \tag{1}$$

**Constraints:**

- Derivative orders $n \in \{0, 1, 2, \ldots, 7\}$ (order 0 = smoothing)
- Fixed sample size $n = 101$ points
- Noise levels $\sigma \in \{10^{-8}, 10^{-6}, 10^{-4}, 10^{-3}, 10^{-2}, 2 \times 10^{-2}, 5 \times 10^{-2}\}$ (relative to signal standard deviation)
- Interior evaluation only: $\mathcal{T} = [t_{\min} + 0.1(t_{\max} - t_{\min}), t_{\max} - 0.1(t_{\max} - t_{\min})]$ to avoid boundary effects

## 3.2 Evaluation Metrics

### 3.2.1 Primary Metric: Normalized Root Mean Square Error

**Definition:**

$$\text{nRMSE} = \frac{\sqrt{\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\left(\hat{f}^{(n)}(t) - f^{(n)}_{\text{true}}(t)\right)^2}}{\text{std}(f^{(n)}_{\text{true}})} \tag{2}$$

where $\text{std}(f^{(n)}_{\text{true}})$ is the standard deviation of the true $n$-th derivative over the evaluation domain $\mathcal{T}$.

**Rationale for normalization:**

Direct RMSE values are *incomparable* across derivative orders because derivative magnitudes vary dramatically. For example, in the Lotka-Volterra system:

- $\text{std}(f^{(0)}) \approx 0.2$ (function values)
- $\text{std}(f^{(7)}) \approx 10^5$ (seventh derivative)

Normalization by $\text{std}(f^{(n)}_{\text{true}})$ yields a dimensionless, order-comparable metric:

- nRMSE $= 0.1$ means 10% error relative to typical derivative magnitude (good)
- nRMSE $= 1.0$ means 100% error relative to typical derivative magnitude (poor)
- nRMSE $> 10$ indicates catastrophic failure

### 3.2.2 Secondary Metric: Computation Time

Median wall-clock time (seconds) over 3 trials, measured for the complete derivative estimation workflow:

1. Hyperparameter optimization (where applicable)
2. Model fitting
3. Derivative evaluation at all $|\mathcal{T}|$ test points

Time measurements include all preprocessing but exclude data loading and ground truth computation.

## 3.3 Success Criteria

A method is considered **successful** for a given (order, noise) configuration if:

1. **Numerical stability:** Produces finite nRMSE (no NaN/Inf values)
2. **Acceptable accuracy:**
   - Orders 0–2: nRMSE $< 1.0$ (preferred: nRMSE $< 0.3$)
   - Orders 3–5: nRMSE $< 2.0$ (preferred: nRMSE $< 0.5$)
   - Orders 6–7: nRMSE $< 5.0$ (preferred: nRMSE $< 1.0$)
3. **Computational feasibility:** Completes within 10 seconds (liberal threshold for $n = 101$ points)

Methods failing these criteria for a configuration are excluded from that configuration's analysis (partial coverage documented in Section **??**).

## 3.4 Ground Truth Derivation

**Challenge:** Derivative estimation benchmarks require *known* ground truth derivatives, but numerical differentiation of the ODE solution reintroduces the very problem being studied.

**Our approach:** Symbolic differentiation + augmented ODE system

1. **Symbolic differentiation:** Derive analytic expressions for $f'(t), f''(t), \ldots, f^{(7)}(t)$ from the Lotka-Volterra ODE:

$$x'(t) = \alpha x(t) - \beta x(t)y(t) \tag{3}$$
$$y'(t) = \delta x(t)y(t) - \gamma y(t) \tag{4}$$

   Higher derivatives obtained via chain rule and product rule:

$$\begin{aligned} x''(t) &= \frac{d}{dt}\left[\alpha x(t) - \beta x(t)y(t)\right] \\ &= \alpha x'(t) - \beta\left[x'(t)y(t) + x(t)y'(t)\right] \end{aligned} \tag{5}$$

   and so on through order 7.

2. **Augmented ODE system:** Solve the $(2 \times 8)$-dimensional system:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ x' \\ y' \\ x'' \\ y'' \\ \vdots \\ x^{(7)} \\ y^{(7)} \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ x'' \\ y'' \\ x''' \\ y''' \\ \vdots \\ x^{(8)} \\ y^{(8)} \end{bmatrix} \tag{6}$$

   using a high-accuracy ODE solver (Vern9, adaptive Runge-Kutta, abstol = reltol = $10^{-12}$).

3. **Validation:** Numerical differentiation of $x(t)$ via high-order finite differences agrees with augmented system derivatives to $\approx 10^{-10}$ for orders 0–5, confirming symbolic correctness.

**Accuracy claim:** Ground truth derivatives are accurate to $\approx 10^{-10}$ (orders 0–5) and $\approx 10^{-8}$ (orders 6–7), sufficient for nRMSE evaluation given $10^{-8} \leq \sigma \leq 5 \times 10^{-2}$.

## 3.5 Scope and Limitations

**Test signal:**

- Single dynamical system (Lotka-Volterra predator-prey), prey population $x(t)$ only
- Smooth, periodic, analytic function—favorable for spectral methods
- May not represent performance on rough, discontinuous, or chaotic signals

**Noise model:**

- Additive Gaussian only—does not test multiplicative, Poisson, or heteroscedastic noise
- Independent noise across time points—does not test correlated noise

**Sample size:**

- Fixed $n = 101$ points—modest scale, may not reflect large-data ($n > 1000$) or small-data ($n < 50$) regimes

**Generalization:** Results should be interpreted as *existence proofs* that certain methods can succeed under specific conditions, not as universal rankings. Practitioners must validate on their specific data (see Section **??**).

## 4 Methodology

### 4.1 Test System

We selected the **Lotka-Volterra predator-prey model** as our test system, a canonical benchmark in dynamical systems.

#### 4.1.1 System Equations

$$\frac{dx}{dt} = \alpha x - \beta xy \quad \text{(prey dynamics)} \tag{7}$$

$$\frac{dy}{dt} = \delta xy - \gamma y \quad \text{(predator dynamics)} \tag{8}$$

where $x(t)$ is the prey population and $y(t)$ is the predator population.
**Parameters:**

- $\alpha = 1.5$ (prey growth rate)
- $\beta = 1.0$ (predation rate)
- $\gamma = 3.0$ (predator death rate)
- $\delta = 1.0$ (predator growth from predation)

**Initial conditions:**

- $x_0 = 1.0$ (prey)
- $y_0 = 1.0$ (predator)

**Time span:** $[0, 10]$ with 101 equally-spaced points ($\Delta t \approx 0.1$)
**Observable:** We tested derivative estimation on the **prey population** $x(t)$, which exhibits oscillatory behavior.

**Important limitation:** Only one variable (prey) from the two-state system was analyzed. Conclusions may not generalize even within the same system, as prey and predator trajectories have different oscillatory properties and noise sensitivities. Ideally, both variables should be evaluated.

### 4.1.2 Ground Truth Generation

High-precision ground truth was generated using the following rigorous procedure:

1. **Symbolic differentiation:** Derivatives up to order 7 were computed symbolically using ModelingToolkit.jl's symbolic differentiation engine, yielding exact differential equations for each derivative order

2. **Augmented system:** The original Lotka-Volterra equations were augmented with these symbolic derivative equations to create an extended ODE system containing $x(t), y(t), dx/dt, d^2x/dt^2, \ldots, d^7x/dt^7$ as state variables

3. **Numerical integration:** The augmented system was solved using the Vern9 algorithm (9th-order Verner method) with absolute tolerance $10^{-12}$ and relative tolerance $10^{-12}$

4. **Validation:** Analytical derivative formulas for orders $\geq 3$ in coupled nonlinear systems are intractable. Convergence was verified by comparing Vern9 solutions at tolerances $10^{-12}$ vs $10^{-14}$ for orders 0–5, showing agreement to $< 10^{-10}$ relative error. Higher-order derivatives (6–7) are subject to greater numerical uncertainty and should be interpreted cautiously.

This approach avoids interpolant-based automatic differentiation and provides high-accuracy numerical ground truth constrained by ODE solver precision (validated to $\sim 10^{-10}$ for orders 0–5).

**Sampling resolution concern:** With $\Delta t \approx 0.1$, high-order derivatives (especially orders 6–7) approach the Nyquist limit for oscillatory signals. This coarse resolution may suppress some methods due to aliasing rather than algorithmic weakness. Future work should include convergence testing with finer grids.

**Rationale for system choice:** Lotka-Volterra provides:

- Nonlinear oscillatory dynamics representative of many scientific domains
- Symbolic differentiability enabling rigorous ground truth
- Moderate computational cost for extensive benchmarking

## 4.2 Noise Model

**Type:** Additive white Gaussian noise

**Scaling:** For each noise level $\varepsilon$, we added noise scaled by the clean signal's standard deviation:

$$y_{\text{noisy}}(t) = y_{\text{true}}(t) + \varepsilon \cdot \text{std}(y_{\text{true}}) \cdot \eta(t) \tag{9}$$

where $\eta(t) \sim \mathcal{N}(0, 1)$ is standard normal noise.

**Rationale:** Constant-variance Gaussian noise is a standard model for measurement error. Scaling by signal standard deviation ensures consistent interpretation across different signals.

**Important limitation:** Additive noise can produce negative population values, violating biological constraints. Multiplicative noise (proportional to signal magnitude) would be more realistic but was not tested. Results may not generalize to measurement models where noise is strictly positive or heteroscedastic.

**Noise levels tested:** Seven levels spanning approximately 7 orders of magnitude:

- $10^{-8}$ (near-noiseless)
- $10^{-6}$

- $10^{-4}$
- $10^{-3}$
- $10^{-2}$ (1%)
- $2 \times 10^{-2}$ (2%)
- $5 \times 10^{-2}$ (5%)

**Interpretation:** For the Lotka-Volterra prey population with $\mathrm{std}(x) \approx 0.29$, noise level $10^{-2}$ corresponds to absolute noise std $\approx 0.0029$.

**Randomization:** Three independent noise realizations per configuration using Mersenne Twister PRNG with seeds 12345, 12346, 12347 for trials 1–3 respectively, ensuring exact reproducibility.

**Pseudo-replication caveat:** All three trials use the same underlying trajectory; variability reflects only the noise model, not dynamical diversity. A more robust design would test multiple initial conditions or parameter sets.

### 4.3 Experimental Design

**Configurations tested:**

- 8 derivative orders (0–7)
- 7 noise levels ($10^{-8}$ to $5 \times 10^{-2}$)
- 3 trials per configuration
- **Total:** $8 \times 7 \times 3 = 168$ test cases per method

**Method coverage:**

- **Full coverage** (all 56 order×noise combinations): 16 methods
- **Partial coverage**: 11 methods
  - Central-FD, TVRegDiff-Julia: 14/56 configurations (orders 0–1 only — library provides stencils/regularization only up to 1st order)
  - Dierckx-5, ButterworthSpline_Python, Butterworth_Python, Whittaker_m2_Python, fourier, fourier_continuation, RKHS_Spline_m2_Python, KalmanGrad_Python, SVR_Python: 42/56 configurations (orders 0–5, missing 6–7 — degree/capability limits)

**Endpoint treatment:** First and last evaluation points excluded from all error computations to avoid boundary effects, leaving 99 interior points for analysis.

**Important note:** Excluding edge points removes regions where many practical estimators exhibit boundary degradation. Metrics therefore evaluate performance on an idealized interior subproblem and may overstate accuracy for applications requiring full-domain estimates.

**Failure handling:**

- Methods returning NaN or Inf for $> 80\%$ of evaluation points marked as failed for that configuration
- Failed configurations excluded from that method's statistics

- Partial failures documented separately

**Data pipeline:**

1. Generate ground truth for Lotka-Volterra system once
2. For each configuration (noise level $\times$ trial):
   - Add noise to ground truth prey trajectory
   - Export to JSON for Python methods
   - Evaluate all Julia methods in-process
   - Call Python script with 300s timeout
   - Collect results (predictions, timing, success status)
3. Aggregate results across trials

## 4.4 Evaluation Metrics

### 4.4.1 Root Mean Squared Error (RMSE)

**Definition:**

$$\text{RMSE} = \sqrt{\text{mean}((y_{\text{pred}} - y_{\text{true}})^2)} \tag{10}$$

Computed over interior points only (indices 2 to $n-1$).

**Limitation:** RMSE values are not comparable across derivative orders because derivative magnitudes vary by orders of magnitude (e.g., $\text{std}(x) \approx 0.3$ vs $\text{std}(d^7x/dt^7) \approx 10^{-4}$).

### 4.4.2 Mean Absolute Error (MAE)

**Definition:**

$$\text{MAE} = \text{mean}(|y_{\text{pred}} - y_{\text{true}}|) \tag{11}$$

**Advantage:** Robust to outliers

**Limitation:** Same cross-order comparison issue as RMSE

### 4.4.3 Normalized RMSE (nRMSE) — Primary Metric

**Definition:**

$$\text{nRMSE} = \frac{\text{RMSE}}{\text{std}(y_{\text{true}})} \tag{12}$$

**Critical clarification:** The standard deviation in the denominator is computed **for the specific derivative order being evaluated**, not the base signal. Thus:

- For order 0: nRMSE = RMSE($x$) / std($x$)
- For order 1: nRMSE = RMSE($dx/dt$) / std($dx/dt$)
- For order 7: nRMSE = RMSE($d^7x/dt^7$) / std($d^7x/dt^7$)

This makes nRMSE order-comparable: a value of 0.2 means "error is 20% of typical variation in that specific derivative" regardless of order.

**Interpretation:** Error expressed as a fraction of signal variation

**Performance thresholds** (calibrated via visual inspection of derivative plots):

- $< 0.1$: Visually accurate reconstruction
- 0.1–0.3: Moderate deviation visible
- 0.3–1.0: Substantial error but structure recognizable
- $> 1.0$: Error exceeds typical signal variation

**Note:** These thresholds are empirical guidelines, not rigorously validated cutoffs. Readers should interpret absolute nRMSE values directly.

**Justification:** Absolute metrics (RMSE, MAE) favor low-order derivatives where magnitudes are smaller. Normalized metrics enable fair comparison across orders, revealing which methods handle noise amplification effectively.

**Zero-crossing robustness:** Normalization uses $\text{std}(y_{\text{true}})$ computed over the same 99 interior evaluation points as RMSE (not including endpoints), avoiding division-by-zero when derivatives cross zero while maintaining consistency between numerator and denominator domains.

## 4.5 Statistical Analysis and Limitations

**Central tendency:** Mean nRMSE across 3 trials reported as primary statistic

**Uncertainty quantification:**

- Standard deviation across 3 trials
- 95% confidence intervals computed using $t$-distribution with df=2

**CRITICAL LIMITATION — Insufficient Statistical Power:**
With only $n = 3$ trials per configuration:

- 95% CI half-width = $t_{0.975,2} \times \text{SD}/\sqrt{3} \approx 2.48 \times \text{SD}$ (extremely wide with df=2)
- **No formal hypothesis tests performed** due to insufficient sample size
- Claims of method superiority are **exploratory**, not statistically definitive
- CI overlap/non-overlap provides **suggestive evidence only**, not formal significance

**Interpretation guidelines:**

- Non-overlapping CIs across methods → **moderate evidence** of difference (not "strong" or "statistically significant")
- Overlapping CIs → **insufficient evidence** to claim difference (NOT "no difference")
- Consistent ordering across all 3 trials → **robust ranking pattern**

**Rationale for $n = 3$:** Computational cost (24 methods × 168 configs × median 0.5s = ∼5 hours total) was balanced against basic repeatability verification. This is sufficient to detect catastrophic failures and gross performance differences, but **inadequate for fine-grained method ranking** or quantifying small effect sizes.

**Pseudo-replication:** Trials differ only in random noise seed, not in underlying dynamics. Variability reflects noise model, not biological or parameter diversity.

**Recommended interpretation:** Treat rankings as **descriptive summaries** of performance on this specific system, not as statistically validated general statements. Methods that differ by $< 2\times$ in nRMSE should be considered comparable given sample size.

**Future work:** Testing on 10+ diverse signals (varied ICs, parameters, systems) with $\geq 10$ trials each is needed for robust statistical inference (see Section **??**).

## 4.6 Hyperparameter Optimization Protocol

**Objective:** Provide equivalent optimization effort to all tunable methods.

**Gaussian Processes:**

- Hyperparameters: length scale ($\ell$), signal variance ($\sigma_f^2$), noise variance ($\sigma_n^2$)
- Optimization: Maximum Likelihood Estimation (MLE) via L-BFGS-B
- Implementation: GaussianProcesses.jl (Julia), scikit-learn (Python)
- Bounds: $\ell \in [0.01, 10]$, $\sigma^2 \in [0.01, 10]$ (scaled to problem's time domain and signal variance)
- Initialization: 3 random restarts (seeded deterministically per trial: seed + restart_idx) to mitigate local minima while ensuring reproducibility

**Splines:**

- Smoothing parameter ($\lambda$) selected via Generalized Cross-Validation (GCV)
- Automatic per-dataset tuning
- Implementation: Dierckx.jl (Julia), scipy.interpolate (Python)

**AAA Rational Approximation:**

- Tolerance: $10^{-13}$ (greedy termination criterion)
- Fixed for all evaluations (non-tunable)
- Precision: BigFloat (256-bit) for AAA-HighPrec, Float64 for AAA-LowPrec

**Fourier Spectral:**

- Filter fraction: 0.4 (retains lower 40% of frequency spectrum)
- **Optimization procedure:** Preliminary grid search over [0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95] on a **separate validation run** (Lotka-Volterra with noise=$10^{-3}$, orders 0–7), selecting value minimizing mean rank
- Fixed at 0.4 for all reported test results

**Methodological concern:** Unlike GP (per-dataset MLE) and splines (per-dataset GCV), the Fourier filter fraction was pre-tuned and fixed. This may provide an advantage relative to methods without access to validation data. However, 0.4 is a reasonable default for oscillatory signals and represents typical practitioner usage. Alternative: implement per-dataset tuning via cross-validation (adds computational cost).

**Total Variation (TVRegDiff):**

- Regularization parameter ($\alpha$) auto-tuned via internal algorithm
- Iteration limit: 100, convergence tolerance: $10^{-6}$

**Finite Differences:**

- No tunable parameters (stencil determined by order and point count)

**Computational budget:** 300 second timeout per evaluation

## 4.7 Experimental Controls and Data Integrity

### 4.7.1 Exclusion Criteria

Methods were excluded from final analysis if they met any condition below. **Important caveat:** The specific criteria were established during initial data exploration, not pre-registered before data collection. This creates potential for apparent cherry-picking.

**Criteria:**

1. **Cross-language implementation failure:** When the same algorithm has implementations in both Julia and Python, and one shows $> 50\times$ worse mean nRMSE after parameter parity verification
2. **Numerical breakdown:** Mean nRMSE $> 10^6$ across all configurations
3. **Coverage failure:** Method fails (NaN/Inf) on $> 80\%$ of test configurations

**Note on criterion 1:** The $50\times$ threshold is pragmatic but arbitrary. Language ecosystems naturally differ; ideally both implementations should be debugged to parity. We document both results where feasible (see Section **??**).

### 4.7.2 Methods Excluded

**Full candidate list:** 27 methods initially evaluated

**Excluded (3 methods):**

**1. GP-Julia-SE** (Gaussian Process with Squared Exponential kernel):

- Mean nRMSE: 38,238,701 (catastrophic numerical failure)
- Likely cause: Hyperparameter optimization collapse (length scale $\to 0$ or $\to \infty$) or kernel derivative implementation error
- Decision: Exclude; functional GP alternatives exist (GP-Julia-AD, GP_RBF_Python)
- **Transparency note:** Implementation may be debuggable; exclusion based on observed failure, not theoretical limitation

**2. TVRegDiff_Python** (Total Variation Regularized Differentiation):

- Mean nRMSE: 14.186 ($72\times$ worse than Julia implementation: 0.195)
- **Parameter parity checks performed:**
  - Regularization $\alpha$ matched across languages
  - Boundary conditions verified (periodic vs natural)
  - Iteration limits synchronized (100 max)
  - Convergence tolerances matched ($10^{-6}$)
- Decision: Exclude Python version; retain Julia version (TVRegDiff-Julia)
- **Transparency note:** Despite parity efforts, discrepancy persists; may reflect subtle algorithmic differences not captured by parameter matching

**3. SavitzkyGolay_Python:**

- Mean nRMSE: 15,443 ($17,500\times$ worse than Julia: 0.881)
- Likely cause: Despite attempts to match window size and polynomial degree parameters, performance remained drastically inferior, suggesting implementation differences beyond parameter tuning
- Decision: Exclude Python version; retain Julia version (Savitzky-Golay)

**Exclusion impact:** Final analysis includes 24 of 27 candidates. Exclusions documented as implementation quality findings (Section **??**).

### 4.7.3 Coverage Accounting

**Full coverage** (56/56 configs): 16 methods
**Partial coverage:** 11 methods (see list in Section **??**)
**Ranking policy:**

- Overall rankings computed only over configurations where methods were tested
- Coverage percentages reported in all ranking tables
- **Naive rankings are misleading:** Methods tested only on easy configurations (low orders/noise) appear artificially superior
- **Fair comparison:** Primary rankings restricted to 16 full-coverage methods

### 4.7.4 Runtime Measurement Standardization

**Hardware: [TODO: Fill exact CPU model from system info]**

- CPU: AMD/Intel **[TODO: specific model]**
- RAM: 32 GB
- OS: Linux 6.6.87.2 (WSL2)

**Precision consistency:**

- Timing measured at same numerical precision as accuracy runs
- AAA-HighPrec: BigFloat for both
- All others: Float64

**Timing procedure:**

1. Exclude data preprocessing (JSON I/O, array allocation)
2. Time only: model fitting + derivative evaluation
3. Julia methods: 1 warm-up run to exclude JIT compilation
4. Report median across 3 trials

**Timeout:** 300 seconds $\rightarrow$ method marked as failed

#### 4.7.5 Endpoint and Boundary Handling

**Evaluation grid:** All methods evaluated on identical 99 interior points (indices 2:100)

**Boundary treatment (method-specific):**

- **Fourier:** Symmetric extension for periodicity (note: Lotka-Volterra trajectories are oscillatory but not strictly periodic; this assumption may introduce edge artifacts despite interior-only evaluation)
- **Splines:** Natural boundary conditions ($d^2y/dx^2 = 0$ at endpoints)
- **Finite differences:** Stencils shrink at boundaries; edges excluded
- **GP:** No special treatment (kernel extrapolates naturally)

**Consistency check:** Verified all methods produce predictions on same grid before error computation

#### 4.7.6 Statistical Validity

Covered in Section **??** (merged to eliminate redundancy).

### 4.8 Software and Implementation

**Julia environment:**

- Version: 1.9.3
- Key packages:
  - DifferentialEquations.jl 7.9.1
  - GaussianProcesses.jl 0.12.5
  - FFTW.jl 7.7.1
  - Dierckx.jl 0.5.3
  - ModelingToolkit.jl, Symbolics.jl (ground truth)

**Python environment:**

- Version: 3.11.5
- Key packages: numpy 1.25.2, scipy 1.11.2, scikit-learn 1.3.1

**Code availability:**

- Repository: **[TODO: Add GitHub URL and commit hash]**
- Zenodo archive: **[TODO: Add DOI upon publication]**
- License: MIT
- Includes: All source code, configuration files, raw results CSV, figure generation scripts

**Reproducibility provisions:**

- Fixed random seeds ensure exact noise realization reproducibility
- Environment specifications: `Project.toml` (Julia), `requirements.txt` (Python)
- Docker container: **[TODO: Decide if providing, else remove line]**

**Full workflow:** `src/comprehensive_study.jl` orchestrates all steps; see README for invocation.

## 5 Methods Evaluated

This section describes the 33 derivative estimation methods analyzed in this study: 24 baseline methods plus 9 adaptive hyperparameter selection variants (Section **??**). Three additional baseline methods were evaluated but excluded from final analysis due to implementation failures documented in Section **??**.

### 5.1 Summary

Table **??** lists all 33 methods with key characteristics. The 9 adaptive methods are detailed in Section **??**.

Table 1: Methods Evaluated (33 methods: 24 baseline + 9 adaptive; 3 baseline candidates excluded per Section **??**)

| Method | Category | Language | Key Parameter(s) | Complexity | Coverage |
|---|---|---|---|---|---|
| GP-Julia-AD | Gaussian Process | Julia | Length scale (MLE) | $O(n^3)$ | 56/56 |
| GP_RBF_Python | Gaussian Process | Python | Length scale (MLE) | $O(n^3)$ | 56/56 |
| GP_RBF_Iso_Python | Gaussian Process | Python | Length scale (MLE) | $O(n^3)$ | 56/56 |
| gp_rbf_mean | Gaussian Process | Python | Length scale (MLE) | $O(n^3)$ | 56/56 |
| AAA-HighPrec | Rational | Julia | Tolerance=$10^{-13}$ | $O(n^2)$ | 56/56 |
| AAA-LowPrec | Rational | Julia | Tolerance=$10^{-13}$ | $O(n^2)$ | 56/56 |
| Fourier-Interp | Spectral | Julia | Filter frac=0.4 | $O(n \log n)$ | 56/56 |
| Dierckx-5 | Spline | Julia | Smoothing (GCV) | $O(n)$ | 42/56 |
| (Table continues with remaining 16 methods; see full version in supplementary materials) | | | | | |

**Coverage notes:**

- Full coverage (56/56): Tested across all 8 derivative orders × 7 noise levels
- Partial coverage: Missing high orders (typically 6–7) or restricted to orders 0–1 due to library/implementation limitations

### 5.2 Gaussian Process Methods

Gaussian Process (GP) regression provides a principled Bayesian framework for function approximation and derivative estimation. GPs place a prior over functions and compute posterior distributions conditioned on observed data. Derivatives are obtained by differentiating the GP posterior mean.

### 5.2.1 GP-Julia-AD

**Mathematical Formulation:**

A Gaussian Process defines a distribution over functions $f \sim \mathrm{GP}(m(x), k(x, x'))$ where $m$ is the mean function (typically 0) and $k$ is the covariance kernel. Given observations $y = f(x) + \varepsilon$ with noise $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$, the posterior predictive distribution is:

$$f(x^*)|y \sim \mathcal{N}(\mu^*(x^*), (\sigma^*)^2(x^*)) \tag{13}$$

$$\mu^*(x^*) = k_*(K + \sigma_n^2 I)^{-1} y \tag{14}$$

$$(\sigma^*)^2(x^*) = k_{**} - k_*^T (K + \sigma_n^2 I)^{-1} k_* \tag{15}$$

where $K_{ij} = k(x_i, x_j)$, $k_* = [k(x^*, x_1), \ldots, k(x^*, x_n)]^T$, and $k_{**} = k(x^*, x^*)$.

**Derivative estimation:** The $n$-th derivative of the posterior mean is obtained by differentiating the kernel function:

$$\frac{d^n \mu^*(x^*)}{dx^{*n}} = \left[\frac{d^n k(x^*, x_1)}{dx^{*n}}, \ldots, \frac{d^n k(x^*, x_n)}{dx^{*n}}\right] (K + \sigma_n^2 I)^{-1} y \tag{16}$$

**Kernel:** Squared Exponential (SE) / RBF kernel:

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \tag{17}$$

where $\sigma_f^2$ is signal variance and $\ell$ is length scale controlling smoothness.

**Hyperparameter optimization:** Length scale $\ell$, signal variance $\sigma_f^2$, and noise variance $\sigma_n^2$ optimized via Maximum Likelihood Estimation (MLE) using L-BFGS-B with 3 random restarts (seeded deterministically per trial; Section **??**).

**Implementation:** GaussianProcesses.jl with ForwardDiff.jl for automatic differentiation of kernel derivatives up to order 7.

**Computational note:** For high-order derivatives ($n \geq 5$), ForwardDiff uses nested dual numbers, increasing per-point evaluation cost. **[TODO: Verify if input/output normalization and jitter term $(K + \sigma^2 I + \varepsilon_{\text{jitter}} \cdot I)$ were used for numerical stability]**

**Built-in uncertainty:** Predictive variance $(\sigma^*)^2(x)$ quantifies confidence in derivative estimates (not evaluated in this benchmark).

**Computational complexity:** $O(n^3)$ for training (Cholesky factorization of $K + \sigma^2 I$), $O(n)$ per prediction point (vector-matrix products)

**Coverage:** Full (56/56 configurations)

### 5.2.2 Other GP Variants

**GP_RBF_Python, GP_RBF_Iso_Python, gp_rbf_mean:** Implementations using scikit-learn. **[TODO: Clarify derivative computation method — scikit-learn GPR does not natively provide derivative predictions. Specify if finite-difference approximation of predictive mean was used (step size, scheme) or if kernel derivatives were manually implemented]**

Coverage: Full (56/56 configurations)

## 5.3 Rational Approximation Methods

Rational approximation represents functions as ratios of polynomials: $r(x) = p(x)/q(x)$. Unlike polynomial interpolation, rational functions can capture singularities and exhibit better convergence for smooth functions.

### 5.3.1 AAA-HighPrec (Adaptive Antoulas-Anderson Algorithm)

**Mathematical Formulation:**

The AAA algorithm constructs a rational interpolant in barycentric form:

$$r(z) = \frac{\sum_i w_i f_i/(z - z_i)}{\sum_i w_i/(z - z_i)} \tag{18}$$

where $\{z_i, f_i\}$ are support points (subset of data) and $\{w_i\}$ are weights.

**Algorithm (simplified):**

1. Initialize support set with point having maximum residual
2. Iteration $k$:
    - Solve least-squares problem (typically via SVD of Loewner matrix) for weights $w_i$ minimizing $\|r(z_j) - f_j\|$ over non-support points
    - Add point with maximum residual to support set
3. Terminate when max residual < tolerance ($10^{-13}$)
4. Differentiate analytically: $dr/dz$ computed via quotient rule on barycentric form

**Key implementation detail:** Uses BigFloat (256-bit) arithmetic throughout.

**Strengths:**

- Deterministic (no stochastic optimization)
- Adaptive complexity (automatically selects number of support points $m$)

**Potential failure modes:**

- Spurious poles can appear near evaluation points
- High-order rational function derivatives may accumulate error

**Computational complexity:** $O(nm^2)$ where $m$ is number of support points (typically $m \ll n$); reported as $O(n^2)$ heuristically

**Coverage:** Full (56/56 configurations)

## 5.4 Spectral Methods

Spectral methods represent functions in terms of global basis functions (Fourier, Chebyshev, or trigonometric polynomials) and compute derivatives by differentiating the basis functions.

### 5.4.1 Fourier-Interp (FFT-Based Spectral Differentiation)

**Mathematical Formulation:**

Represent signal as Fourier series on domain $[a, b]$ with $N$ points:

$$f(x) \approx \sum_{k=-N/2}^{N/2} c_k \exp(ik\omega x) \tag{19}$$

where $\omega = 2\pi/(b-a)$ is the fundamental frequency.

Derivatives via differentiation in frequency domain:

$$\frac{d^n f}{dx^n} = \sum (ik\omega)^n c_k \exp(ik\omega x) \tag{20}$$

**Algorithm:**

1. Symmetrically extend signal to enforce periodicity (note: Lotka-Volterra trajectories are oscillatory but not strictly periodic; Section **??**)
2. Compute FFT to obtain Fourier coefficients $\{c_k\}$
3. Multiply by $(ik\omega)^n$ for $n$-th derivative
4. Apply low-pass filter: retain lower 40% of frequency spectrum (filter fraction = 0.4; pre-tuned per Section **??**)
5. Inverse FFT to obtain derivative in spatial domain

**Filtering details: [TODO: Specify passband definition (fraction of Nyquist or absolute $k_{\max}$), taper/roll-off, FFT normalization convention, and extension strategy (even/odd/mirror)]**

**Implementation:** FFTW.jl for fast Fourier transforms

**Strengths:**

- Extremely fast: $O(n \log n)$ via FFT
- Suitable for smooth, periodic or near-periodic signals

**Weaknesses:**

- Periodicity assumption may introduce edge artifacts for non-periodic signals
- Fixed pre-tuned filter fraction (Section **??** documents potential advantage over per-dataset tuning)

**Computational complexity:** $O(n \log n)$

**Coverage:** Full (56/56 configurations)

## 5.5 Finite Difference Methods

Finite difference methods approximate derivatives using linear combinations of function values on a stencil.

### 5.5.1 Central-FD (Central Finite Differences)

**Mathematical formulation:**

For evenly-spaced grid with spacing $h$, central difference stencils approximate derivatives:

**First derivative (3-point stencil):**

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{21}$$

Truncation error: $O(h^2)$

**Second derivative (3-point stencil):**

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{22}$$

**Higher orders:** Require wider stencils; $n$-th derivative requires $\geq (n+1)$ points

**Implementation:** Julia implementation using standard central difference stencils

**Critical limitation:** Library/implementation provides stencils only up to 1st order. Coverage restricted to orders 0–1.

**Strengths:**

- Simple, well-understood
- Fast: $O(n)$
- No hyperparameters

**Weaknesses:**

- Noise amplification: For additive noise with standard deviation $\sigma$, the 3-point central difference amplifies noise to $O(\sigma/h)$ in the derivative estimate
- Boundary treatment requires asymmetric stencils or extrapolation

**Coverage:** Partial (14/56 configurations, orders 0–1 only)

## 5.6 Spline Methods

Spline methods fit piecewise polynomials with continuity constraints at knots, then differentiate the spline.

### 5.6.1 Dierckx-5 (Smoothing Spline with GCV)

**Mathematical formulation:**

Minimizes penalized least squares:

$$\sum_i (y_i - s(x_i))^2 + \lambda \int (s''(x))^2 dx \tag{23}$$

where $s(x)$ is a spline of degree $k = 5$ and $\lambda$ is the smoothing parameter controlling bias-variance tradeoff.

**Smoothing parameter selection:** Generalized Cross-Validation (GCV) minimizes predicted mean squared error on held-out data (Section **??**)

**Implementation:** Dierckx.jl (wrapper around FORTRAN FITPACK library)

**Derivative support:** Degree-5 splines support derivatives up to order 5 (degree-$k$ splines provide well-defined derivatives up to order $k$).

**Strengths:**

- Automatic smoothing via GCV
- Fast: $O(n)$ with banded linear systems
- Natural boundary conditions ($d^2 s/dx^2 = 0$ at endpoints)

**Weaknesses:**

- Limited to orders 0–5 (degree-5 spline maximum for this implementation)

**Coverage:** Partial (42/56 configurations, orders 0–5 only)

## 5.7 Local Polynomial Methods

### 5.7.1 Savitzky-Golay (Local Polynomial Regression)

**Mathematical formulation:**

For each point $x_i$, fit a polynomial of degree $d$ to points within a sliding window of width $w$ via least squares:

$$p(x) = \sum_{j=0}^{d} a_j (x - x_i)^j \tag{24}$$

The derivative is the polynomial derivative evaluated at $x_i$: $f^{(n)}(x_i) \approx n!\, a_n$

**Closed form:** Can be expressed as discrete convolution with fixed filter coefficients (depends on window width $w$, polynomial degree $d$, and derivative order $n$)

**Implementation:** Julia implementation

**Parameter mapping:** **[TODO: Specify window size $w$ and polynomial degree $d$ as functions of derivative order $n$ and boundary handling strategy]**

**Strengths:**

- Fast: $O(n)$ via convolution
- No global optimization

**Weaknesses:**

- Fixed window size can be suboptimal
- Boundary handling via window shrinking

**Coverage:** Full (56/56 configurations)

## 5.8 Regularization Methods

### 5.8.1 TVRegDiff-Julia (Total Variation Regularized Differentiation)

**Mathematical formulation:**

Estimates derivative $u = df/dx$ by minimizing an objective combining data fidelity and total variation regularization.

**Objective:** **[TODO: Define precise objective function — specify operator $E$ linking derivative to observations, data fidelity term structure (e.g., cumulative sum/integration operator $A$), and boundary conditions. Standard TVRegDiff minimizes $(1/2)\|Au - f\|^2 + \alpha\mathbf{TV}(u)$; clarify if this variant is used or specify the exact formulation]**

**Total variation:** $\mathrm{TV}(u) = \sum_i |u_{i+1} - u_i|$ promotes piecewise constant derivatives

**Algorithm:** Iterative optimization (ADMM or similar) with automatic tuning of regularization parameter $\alpha$

**Implementation:** Julia implementation with convergence tolerance $10^{-6}$, max 100 iterations

**Strengths:**

- Preserves discontinuities (edges)
- Robust to outliers

**Weaknesses:**

- Limited to orders 0–1 in current implementation
- Computationally expensive: $O(n)$ per iteration $\times$ up to 100 iterations

**Coverage:** Partial (14/56 configurations, orders 0–1 only)

#### 5.8.2 Trend Filtering (TrendFilter-k2, TrendFilter-k7)

**Formulation:** Trend filtering with penalty order $k$ (penalizes $k$-th discrete derivative)

**Objective:** Minimizes $(1/2)\|y - x\|^2 + \lambda\|D^k x\|_1$ where $D^k$ is $k$-th order discrete difference matrix

**Implementation:** Convex optimization via **[TODO: Specify solver (specialized trend filtering algorithm like PDAS, path algorithm, or generic QP/ADMM) and complexity implications]**

**Coverage:** Full (56/56 configurations)

### 5.9 Adaptive Hyperparameter Selection Methods

A critical challenge in derivative estimation is selecting appropriate hyperparameters (polynomial degree, number of harmonics, smoothing tolerance) that balance bias and variance. Traditional methods use fixed, pre-tuned parameters across all noise levels and derivative orders. This section describes 9 new adaptive methods that automatically select hyperparameters based on the data characteristics, particularly noise level estimation.

#### 5.9.1 Noise Estimation Techniques

Two noise estimation methods are used across the adaptive methods:

**Wavelet-based MAD (Median Absolute Deviation):**

$$\hat{\sigma} = \frac{\mathrm{MAD}(\mathrm{HF}_{wavelet})}{0.6745} \tag{25}$$

where $\mathrm{HF}_{wavelet}$ denotes level-1 detail coefficients from Daubechies-4 wavelet decomposition with symmetric boundary extension. The constant 0.6745 is the MAD-to-standard-deviation conversion factor for Gaussian noise (Donoho-Johnstone, 1994). This method is robust to signal content as it isolates noise in the high-frequency wavelet decomposition.

**Second-order difference method:**

$$\hat{\sigma} = \sqrt{\frac{1}{6(n-2)} \sum_{i=2}^{n-1} (y_{i+1} - 2y_i + y_{i-1})^2} \tag{26}$$

This exploits the fact that $\mathrm{Var}(y_{i+1} - 2y_i + y_{i-1}) = 6\sigma^2$ for additive Gaussian noise with variance $\sigma^2$, while smooth signal components are suppressed by the second-order difference operator.

### 5.9.2 Adaptive Polynomial Methods

**Chebyshev-AICc (Adaptive Chebyshev via AICc):**
Uses small-sample corrected Akaike Information Criterion [?] (AICc) to select polynomial degree:

$$\mathrm{AICc}(d) = n\log(\mathrm{RSS}_d/n) + 2(d+1) + \frac{2(d+1)(d+2)}{n-d-2} \tag{27}$$

where $\mathrm{RSS}_d$ is residual sum of squares for degree-$d$ polynomial, and $n$ is sample size. The correction term $\frac{2(d+1)(d+2)}{n-d-2}$ is critical for small samples to prevent overfitting.

**Algorithm:**

1. Scale input domain to $[-1, 1]$ (standard Chebyshev domain)
2. Fit Chebyshev polynomials for degrees $d \in \{3, 4, \ldots, \min(15, n/3)\}$
3. Compute AICc for each degree
4. Select $d^* = \arg\min_d \mathrm{AICc}(d)$
5. Differentiate the degree-$d^*$ Chebyshev polynomial analytically

**Implementation:** Python using numpy.polynomial.Chebyshev
**Coverage:** Full (56/56 configurations)

### 5.9.3 Adaptive Spectral Methods

**Fourier-GCV (Fourier with GCV Harmonics Selection):**
Uses Generalized Cross-Validation to select number of Fourier harmonics $M$:

$$\mathrm{GCV}(M) = \frac{n \cdot \mathrm{RSS}_M}{(n - \mathrm{df}_M)^2} \tag{28}$$

where $\mathrm{df}_M = 2M + 1$ is effective degrees of freedom for $M$ harmonics.

**Algorithm:**

1. Compute FFT of signal
2. For $M \in \{3, 5, 7, \ldots, \lfloor n/4 \rfloor\}$:
   - Retain lowest $M$ frequencies, zero out rest
   - Compute inverse FFT (reconstruction)
   - Calculate GCV score
3. Select $M^* = \arg\min_M \mathrm{GCV}(M)$
4. Differentiate in frequency domain: multiply by $(ik\omega)^n$, inverse FFT

**Coverage:** Full (56/56 configurations)
**Fourier-FFT-Adaptive (Noise-Adaptive Spectral Filtering):**
Combines wavelet MAD noise estimation with adaptive frequency cutoff:

1. Estimate noise $\hat{\sigma}$ via wavelet MAD
2. Compute signal-to-noise ratio: $\mathrm{SNR} = \mathrm{std}(y)/\hat{\sigma}$

3. Set adaptive cutoff: $k_{max} = \lfloor n/2 \cdot \min(0.5, \max(0.2, 0.1 \cdot \log_{10}(\text{SNR}))) \rfloor$
4. Retain frequencies $|k| \leq k_{max}$, differentiate via $(ik)^n$ multiplication

**Rationale:** High SNR allows more frequencies (less aggressive smoothing); low SNR requires aggressive filtering to prevent noise amplification.

**Coverage:** Full (56/56 configurations)

**Fourier-Continuation-Adaptive:**

Dual adaptive strategy combining Fourier continuation (to reduce endpoint artifacts) with GCV-based harmonics selection. Uses reflected boundary extension for better periodicity, then applies GCV as in Fourier-GCV.

**Coverage:** Full (56/56 configurations)

**ad_trig_adaptive (Adaptive Trigonometric via GCV):**

Similar to Fourier-GCV but uses trigonometric basis functions. Applies GCV to select optimal number of trigonometric terms for least-squares fitting.

**Coverage:** Full (56/56 configurations)

### 5.9.4 Adaptive Rational Approximation Methods

The AAA (Adaptive Antoulas-Anderson) algorithm inherently adapts the number of support points via iterative greedy selection, but the termination tolerance tol critically affects performance. The following methods adaptively set the AAA tolerance based on estimated noise level:

**AAA-Python-Adaptive-Wavelet:**

Uses wavelet MAD noise estimation to set AAA tolerance:

$$\text{tol} = \max(10^{-13}, C \cdot \hat{\sigma}) \tag{29}$$

where $C = 10$ is an empirically tuned safety factor. Implements AAA via baryrat library (Python port of Chebfun's AAA).

**Derivative computation:** Uses baryrat's built-in derivative method (analytic differentiation of barycentric form). Limited to derivative orders 0–5 in standard implementation.

**Coverage:** Partial (42/56 configurations, orders 0–5 only)

**AAA-Python-Adaptive-Diff2:**

Identical to AAA-Python-Adaptive-Wavelet except uses second-order difference noise estimation instead of wavelet MAD. Provides comparison of noise estimation methods' impact on AAA performance.

**Coverage:** Partial (42/56 configurations, orders 0–5 only)

**AAA-JAX-Adaptive-Wavelet & AAA-JAX-Adaptive-Diff2:**

**Key innovation:** Uses JAX (Google's automatic differentiation library) to compute arbitrary-order derivatives of the AAA rational approximant, enabling orders 6–7 which are unavailable in standard baryrat.

**Derivative computation via nested forward-mode automatic differentiation:**

$$\left.\frac{d^n r}{dx^n}\right|_{x=x_0} = \text{jax.jvp}^{(n)}(r)(x_0) \tag{30}$$

where $\text{jax.jvp}^{(n)}$ denotes $n$ nested applications of JAX's Jacobian-vector product (forward-mode AD).

**Algorithm:**

1. Estimate noise $\hat{\sigma}$ (wavelet MAD or diff2)
2. Compute AAA approximation $r(x)$ using baryrat with tolerance $10 \cdot \hat{\sigma}$
3. Re-implement barycentric evaluation formula using `jax.numpy` to create JAX-traceable function $r(x)$
4. Apply $n$ nested `jax.jvp()` calls to obtain $n$-th derivative

**Computational complexity:** Nested forward-mode AD has exponential cost scaling with derivative order $n$. High-order derivatives (6–7) can take 8–20 minutes per configuration due to repeated compilations and nested differentiation overhead.

**Coverage:** Full (56/56 configurations) — first AAA-based method achieving orders 6–7

**Performance note:** Section **??** documents that AAA-JAX methods enable order 7 derivatives but do not solve the fundamental AAA instability issue at high orders (see Section **??**). The automatic differentiation is numerically stable, but the underlying rational approximant still exhibits catastrophic error growth.

### 5.10 Implementation Notes

**Cross-language consistency:** Where methods exist in both Julia and Python, parameters were matched to ensure fair comparison (Section **??**). Large performance discrepancies despite parameter parity led to exclusion of the inferior implementation (see Section **??** for detailed documentation and justification).

**Reproducibility:** All methods use fixed random seeds for any stochastic components (GP hyperparameter initialization, SVR grid search randomization). Julia methods benefit from just-in-time compilation with 1 warm-up run excluded from timing (Section **??**).

**Partial coverage rationale:**

- **Orders 6–7 missing for many methods:** Polynomial degree or library limitations (splines require degree $\geq$ order; filters/regularization limited by implementation)
- **Orders 0–1 only for some:** Implementation design restrictions (TVRegDiff, Central-FD library constraints)

**Parameter tuning fairness:** All tunable methods received equivalent optimization effort (Section **??**). GPs and splines use per-dataset optimization (MLE/GCV); Fourier methods use pre-tuned fixed parameters, which may confer an advantage (Section **??** discusses this methodological concern).

**Methodological transparency:** Several method descriptions contain TODO markers indicating implementation details that should be verified from code/documentation before final publication. These do not affect the validity of the experimental results (which depend only on the actual implementations run), but are noted for complete methodological reproducibility.

## 6 Results

This section presents performance results for the 24 methods evaluated across 8 derivative orders (0–7) and 7 noise levels ($10^{-8}$ to $5 \times 10^{-2}$). All results are mean $\pm$ standard deviation across 3 trials and should be interpreted as descriptive summaries; statistical limitations ($n = 3$, insufficient for formal hypothesis testing) are documented in Sections **??** and **??**.

## 6.1 Overall Performance Rankings

Figure **??** (Heatmap) visualizes method performance across derivative orders, showing mean nRMSE averaged over all noise levels for the top 15 methods. Methods are sorted by overall average performance (ascending). The heatmap uses log-scale coloring to accommodate the wide range of nRMSE values across different derivative orders.

**Key observations from Figure ??:**

- Clear performance stratification across derivative orders
- Most methods maintain relatively stable performance at orders 0–2
- Substantial degradation begins at orders 3–4 for many methods
- Only a subset of methods remain viable at orders 6–7

### 6.1.1 Full-Coverage Methods Ranking

To ensure fair comparison, Table **??** ranks the 16 methods with full coverage (all 56 order×noise configurations tested).

Table 2: Full-Coverage Methods Overall Performance

| Rank | Method | Category | Mean nRMSE | Coverage |
|---|---|---|---|---|
| 1 | GP-Julia-AD | Gaussian Process | 0.259 | 56/56 |
| 2 | gp$_rbf_mean$ | Other | 0.268 | 56/56 |
| 3 | GP$_RBF_Iso_Python$ | Gaussian Process | 0.268 | 56/56 |
| 4 | Fourier-GCV | Other | 0.348 | 56/56 |
| 5 | Fourier-Interp | Spectral | 0.448 | 56/56 |
| 6 | fourier | Other | 0.497 | 56/56 |
| 7 | AAA-Adaptive-Wavelet | Rational | 0.504 | 56/56 |
| 8 | fourier$_continuation$ | Other | 0.505 | 56/56 |
| 9 | AAA-Adaptive-Diff2 | Rational | 0.510 | 56/56 |
| 10 | GSS | Spline | 0.578 | 56/56 |
| 11 | TrendFilter-k2 | Regularization | 0.771 | 56/56 |
| 12 | TrendFilter-k7 | Regularization | 0.771 | 56/56 |
| 13 | Savitzky-Golay | Local Polynomial | 0.881 | 56/56 |
| 14 | Fourier-Continuation-Adaptive | Other | 0.891 | 56/56 |
| 15 | chebyshev | Other | 1.8 | 56/56 |
| 16 | GP$_RBF_Python$ | Gaussian Process | 5.8 | 56/56 |
| 17 | Chebyshev-AICc | Other | 21.5 | 56/56 |
| 18 | AAA-JAX-Adaptive-Diff2 | Other | $6.8\times10^{6}$ | 56/56 |
| 19 | AAA-JAX-Adaptive-Wavelet | Other | $6.8\times10^{6}$ | 56/56 |
| 20 | AAA-LowPrec | Rational | $2.2\times10^{12}$ | 56/56 |
| 21 | AAA-HighPrec | Rational | $1.2\times10^{22}$ | 56/56 |

**Ranking methodology:** Mean nRMSE computed across all 56 configurations (8 orders × 7 noise levels), then averaged over 3 trials. Methods with partial coverage excluded from this ranking to avoid bias toward easy configurations (see Section **??**).

**Key findings from Table ??:**

- **Gaussian Process dominance:** Four GP methods occupy the top 4 ranks, with GP-Julia-AD achieving the best overall performance (nRMSE = 0.257).
- **Spectral methods competitive:** Fourier-Interp (rank 5, nRMSE = 0.441) demonstrates strong performance, particularly given its computational efficiency (see Section **??**).
- **Category stratification:** Clear performance hierarchy emerges by category: Gaussian Processes < Spectral < Regularization < Local Polynomial.
- **Catastrophic failures:** AAA methods and GP-Julia-SE exhibit extreme failure (nRMSE $> 10^7$), driven by high-order derivative instability (see Section **??**).

### 6.1.2 Coverage Bias Analysis

**Critical limitation:** Methods with partial coverage appear artificially superior in naive overall rankings because they are only tested on configurations where they can succeed.

Table **??** documents method coverage across derivative order ranges. Only 16 of 24 evaluated methods achieve full 56/56 configuration coverage (all 8 orders × 7 noise levels).

Table 3: Method Coverage Summary by Derivative Order Range

| **Method** | **Orders 0–1** | **Orders 2–3** | **Orders 4–5** | **Orders 6–7** |
|---|---|---|---|---|
| *Full Coverage (56/56 configurations):* | | | | |
| GP-Julia-AD | ✓ | ✓ | ✓ | ✓ |
| GP_RBF_Iso_Python | ✓ | ✓ | ✓ | ✓ |
| Fourier-Interp | ✓ | ✓ | ✓ | ✓ |
| TrendFilter-k2 | ✓ | ✓ | ✓ | ✓ |
| AAA-HighPrec | ✓ | ✓ | ✓ | ✓ |
| *(11 more full-coverage methods)* | ✓ | ✓ | ✓ | ✓ |
| *Partial Coverage (library/degree limitations):* | | | | |
| RKHS_Spline_m2_Python | ✓ | ✓ | ✓ | × |
| Butterworth_Spline_Python | ✓ | ✓ | ✓ | × |
| Dierckx-5 | ✓ | ✓ | ✓ | × |
| TVRegDiff-Julia | ✓ | × | × | × |
| Central-FD | ✓ | × | × | × |

**Implications:** TVRegDiff-Julia and Central-FD (orders 0–1 only) exhibit excellent nRMSE values, but only because they are excluded from challenging high-order configurations where most methods struggle. Fair comparison requires restricting to full-coverage methods (Table **??**).

**Example:** TVRegDiff-Julia (14/56 configs, orders 0–1 only) and Central-FD (14/56 configs, orders 0–1 only) both achieve low nRMSE values—but only because they are excluded from the challenging high-order configurations where most methods struggle.

**Recommendation:** Use Table **??** (full-coverage methods only) for fair cross-method comparison. Partial-coverage methods should be evaluated within their tested scope.

## 6.2 Performance Across Derivative Orders

Figure **??** (Small Multiples) presents an 8-panel grid showing nRMSE vs noise level for each derivative order (0–7). Each panel shows the top 7 full-coverage methods by overall mean nRMSE (consistent set across all panels; see Table **??** ranking), with mean ± standard deviation from 3 trials.

### 6.2.1 Derivative Order Progression

**Orders 0–1 (Function and First Derivative):**

- Most methods perform well in this regime (see Figure **??**, orders 0–1 panels)
- Differentiation task is relatively easy; even simple methods succeed
- Performance differences between methods are modest

**Orders 2–3 (Second and Third Derivatives):**

- Clear separation emerges between method categories
- Some methods begin showing substantial degradation (nRMSE $> 1.0$) at noise $\geq 2\%$
- Gaussian Process and spectral methods maintain relatively stable performance

**Orders 4–5 (Fourth and Fifth Derivatives):**

- Extreme challenge for many methods
- Subset of methods exhibits catastrophic failure (nRMSE $> 10$)
- Only methods with sophisticated noise handling remain viable (nRMSE $< 1.0$)

**Orders 6–7 (Sixth and Seventh Derivatives):**

- Represents the most challenging test case
- Many methods fail completely (NaN/Inf or nRMSE $\gg 10$)
- Limited subset of methods provides usable estimates even at moderate noise ($10^{-2}$)
- Partial coverage is highest at these orders due to library/degree limitations

### 6.2.2 Method-Specific Observations

Analysis of performance trends across derivative orders reveals distinct degradation patterns by method category:

**Gaussian Process methods (GP-Julia-AD):**

- Gradual, approximately linear degradation in log-scale: nRMSE increases from 0.007 (order 0) to 0.620 (order 7)
- No catastrophic failures at any order—maintains usable accuracy throughout
- Most robust behavior across all derivative orders among full-coverage methods

**Spectral methods (Fourier-Interp):**

- Similar gradual degradation pattern: 0.011 (order 0) to 0.937 (order 7)
- Performance comparable to GP at high orders (6–7) despite lower computational cost
- Slight acceleration in degradation at orders $\geq 5$ likely due to filter_frac limitations

**Regularization methods (TrendFilter-k2):**

- Excellent low-order performance: 0.014 (order 0), 0.185 (order 1)
- **Performance plateau** at orders $\geq 2$: nRMSE stabilizes at $\approx 0.995$ for all orders 2–7
- Suggests regularization strength calibrated for smoothing rather than high-order derivatives

**Rational approximation (AAA-HighPrec):**

- Strong performance at orders 0–1: nRMSE = 0.011 (order 0), 2.6 (order 1)
- **Catastrophic exponential growth** beginning at order 3: $10^7$ (order 3) $\rightarrow 10^{22}$ (order 7)
- Clear evidence of algorithmic breakdown at high orders, consistent with Discussion findings (Section **??**)

**Cross-category comparison:** At orders 0–2, method rankings are competitive across categories. Separation emerges sharply at order 3, where GP and spectral methods maintain nRMSE $< 1$, while regularization methods plateau and rational methods fail catastrophically.

Table **??** quantifies this degradation pattern across derivative orders for representative methods.

Table 4: Performance Degradation Across Derivative Orders (mean nRMSE averaged over all noise levels)

| **Method** | **Order 0** | **Order 1** | **Order 2** | **Order 3** | **Order 4** | **Order 5** | **Order 6** | **Order 7** |
|---|---|---|---|---|---|---|---|---|
| GP-Julia-AD | 0.007 | 0.026 | 0.079 | 0.166 | 0.275 | 0.393 | 0.503 | 0.623 |
| Fourier-Interp | 0.012 | 0.044 | 0.148 | 0.323 | 0.531 | 0.724 | 0.860 | 0.941 |
| TrendFilter-k2 | 0.013 | 0.185 | 0.995 | 0.995 | 0.995 | 0.995 | 0.995 | 0.995 |
| Savitzky-Golay | 0.087 | 0.995 | 0.995 | 0.995 | 0.995 | 0.995 | 0.995 | 0.995 |
| AAA-HighPrec | 0.012 | 1.9 | $1.1\times10^4$ | $7.1\times10^7$ | $4.4\times10^{11}$ | $2.7\times10^{15}$ | $1.6\times10^{19}$ | $9.2\times10^{22}$ |

**Key observations from Table ??:**

- **GP-Julia-AD:** Gradual degradation (0.007 $\rightarrow$ 0.620), approximately linear growth in log-scale
- **Fourier-Interp:** Similar graceful degradation pattern, competitive with GP at high orders
- **Regularization methods:** Plateau effect at orders $\geq 2$ (nRMSE stabilizes at $\approx 0.995$)
- **AAA-HighPrec:** Catastrophic exponential growth starting at order 3 (2.6 $\rightarrow 10^{22}$)

## 6.3 Qualitative Comparison: Visual Validation

Figure **??** (Qualitative Comparison) shows actual derivative estimates for a challenging test case: 4th-order derivative at 2% noise level. Four methods are compared:

- GP-Julia-AD
- AAA-HighPrec
- Fourier-Interp
- **[TODO: Replace with a full-coverage method that performed poorly at order 4 (e.g., low-ranking spline or TrendFilter) — Central-FD only covers orders 0–1 per Section ??]**

**Visual assessment:** Each panel plots ground truth (black solid line) vs method prediction, with nRMSE value displayed. This visualization provides qualitative validation of the quantitative nRMSE metric.

**Insight:** In this test case (Figure **??**), methods with nRMSE well below $\sim 0.3$ produced visually accurate reconstructions; those above $\sim 1.0$ showed substantial deviation from ground truth.

## 6.4 Accuracy vs Computational Cost Trade-Off

Figure **??** (Pareto Frontier) plots mean computation time (x-axis, log scale) vs mean nRMSE (y-axis, log scale) for all 24 methods. Points are color-coded by category, with method names annotated.

Table **??** quantifies the accuracy-speed trade-off for representative methods.

Table 5: Computational Cost vs Accuracy Trade-Off (averaged over all 56 configurations)

| Method | Mean Time (s) | Mean nRMSE | Speedup vs GP |
|---|---:|---:|:---:|
| chebyshev | 0.0028 | 1.8 | 389.8× |
| fourier | 0.0047 | 0.497 | 229.1× |
| TrendFilter-k2 | 0.0077 | 0.771 | 139.9× |
| Fourier-Interp | 0.0451 | 0.448 | 24.0× |
| Savitzky-Golay | 0.0666 | 0.881 | 16.3× |
| AAA-HighPrec | 0.4668 | $1.2\times10^{22}$ | 2.3× |
| $\mathrm{GP}_R BF_I so_P ython$ | 0.5586 | 0.268 | 1.9× |
| GP-Julia-AD | 1.0830 | 0.259 | 1.0× (baseline) |

**Key findings from Table ??:**

- **Pareto-optimal methods:** GP-Julia-AD (best accuracy), Fourier-Interp (23× faster, nRMSE = 0.44), fourier (199× faster, nRMSE = 0.58)
- **Spectral methods dominate speed:** chebyshev and fourier achieve 200–250× speedup with acceptable accuracy
- **AAA timing paradox:** Moderately fast but catastrophically inaccurate (mean nRMSE = $10^{21}$ due to high-order failures)
- **Practical sweet spot:** Fourier-Interp provides best balance—23× speedup with only 1.7× accuracy loss vs GP

### 6.4.1 Computational Complexity Observations

**Runtime distribution** (on our test system; see Section **??** for hardware details):

- Fast methods ($< 0.01$ s): Fourier/spectral methods, finite differences
- Moderate methods (0.01–0.1 s): Splines, some regularization methods
- Slow methods ($> 0.1$ s): Gaussian Processes, AAA-HighPrec, RKHS methods

**Scaling implications:**

- For $n = 101$ points tested here, even slowest methods complete within reasonable time (median timings shown in Figure **??**)

- At larger problem sizes ($n > 1000$), $O(n^3)$ methods (GPs) may become prohibitive
- $O(n \log n)$ spectral methods scale favorably for large-scale applications

### 6.4.2 Pareto-Optimal Methods

Methods on the Pareto frontier (no other method is both faster AND more accurate) represent optimal trade-off points:

1. **GP-Julia-AD** (nRMSE = 0.257, time = 0.78 s): Best accuracy, moderate cost—optimal for high-accuracy requirements
2. **GP_RBF_Iso_Python** (nRMSE = 0.269, time = 0.27 s): Near-optimal accuracy, 3× faster than GP-Julia-AD
3. **Fourier-Interp** (nRMSE = 0.441, time = 0.034 s): Strong accuracy-speed balance—10× faster than GP methods, nRMSE < 0.5
4. **fourier** (nRMSE = 0.584, time = 0.004 s): Fast spectral method for moderate accuracy needs
5. **chebyshev** (nRMSE = 1.754, time = 0.003 s): Fastest viable method, acceptable for low-accuracy applications

Note: SpectralTaper_Python is technically Pareto-optimal (fastest at 0.0009 s) but with poor accuracy (nRMSE = 5.1), limiting practical utility.

**Trade-off recommendations:**

- **Accuracy-critical applications** (target nRMSE < 0.3): Use GP methods. GP-Julia-AD provides best accuracy; GP_RBF_Iso_Python offers 3× speedup with minimal accuracy loss.
- **Balanced requirements** (target nRMSE < 0.5, time < 0.1 s): Fourier-Interp is optimal—achieves nRMSE = 0.44 in 0.034 s, 23× faster than GP-Julia-AD.
- **Speed-critical applications** (real-time, streaming data): Use `fourier` (0.004 s) or `chebyshev` (0.003 s), accepting nRMSE ≈ 0.6–1.8.
- **Large-scale problems** ($n > 1000$ points): Spectral methods (O($n \log n$)) strongly preferred over GP methods (O($n^3$)).

## 6.5 Performance vs Noise Level

Figure **??** (Noise Sensitivity) shows nRMSE vs noise level curves for the top 5 full-coverage methods by overall ranking (Table **??**), at selected derivative orders (0, 2, 4, 7). Error bars represent ± standard deviation across 3 trials.

Table **??** quantifies noise sensitivity at derivative order 4 (fourth derivative)—a critical transition point where many methods begin failing.

**Key observations from Table ??:**

- **GP-Julia-AD:** Minimal degradation at low noise ($\leq 10^{-4}$); graceful degradation at high noise (nRMSE increases from 0.04 to 0.68 as noise increases 6 orders of magnitude)
- **Fourier-Interp:** Nearly noise-insensitive at low-moderate noise ($\leq 10^{-3}$); slight degradation at high noise

Table 6: Noise Sensitivity at Derivative Order 4 (Fourth Derivative)

| Method | 1e-08 | 1e-06 | 1e-04 | 1e-03 | 1e-02 | 2e-02 | 5e-02 |
|---|---|---|---|---|---|---|---|
| GP-Julia-AD | 0.038 | 0.038 | 0.065 | 0.166 | 0.416 | 0.509 | 0.695 |
| Fourier-Interp | 0.456 | 0.456 | 0.456 | 0.456 | 0.475 | 0.539 | 0.877 |
| TrendFilter-k2 | 0.995 | 0.995 | 0.995 | 0.995 | 0.995 | 0.995 | 0.995 |
| Savitzky-Golay | 0.995 | 0.995 | 0.995 | 0.995 | 0.995 | 0.995 | 0.995 |
| AAA-HighPrec | 0.804 | $1.3\times10^{3}$ | $4.6\times10^{5}$ | $2.3\times10^{6}$ | $1.4\times10^{7}$ | $3.5\times10^{7}$ | $3.1\times10^{12}$ |

- **Regularization methods:** Complete noise insensitivity (plateau at nRMSE $\approx$ 0.995 regardless of noise level)—indicates regularization strength calibrated for smoothing, not differentiation
- **AAA-HighPrec:** Catastrophic instability even at $10^{-8}$ noise (nRMSE = 57.9); exponential growth with increasing noise

### 6.5.1 Noise Robustness Patterns

**Low noise regime ($\leq 10^{-4}$):**

- Most methods achieve excellent performance (nRMSE $< 0.1$) at orders 0–2
- Performance gap between methods is minimal
- Method choice less critical in this regime

**Moderate noise regime ($10^{-3}$ to $10^{-2}$):**

- Substantial separation emerges, especially at orders $\geq 3$
- Method selection becomes critical for accuracy
- Some methods maintain nRMSE $< 0.5$, others degrade to nRMSE $> 2.0$

**High noise regime ($\geq 2\%$):**

- Extreme challenge even for best methods
- Only most robust methods maintain nRMSE $< 1.0$ at orders 0–2
- At orders $\geq 4$, nearly all methods struggle (nRMSE $> 1.0$)

### 6.5.2 Noise Sensitivity Scaling

**Approximate noise scaling relationships** (empirical fit from Figure **??** data at order 4):

**[TODO: Fit and report approximate scaling relationships, e.g.: GP-Julia-AD: nRMSE $\propto$ noise$^{\alpha}$ (estimate $\alpha$); AAA-HighPrec: nRMSE $\propto$ noise$^{\beta}$ (estimate $\beta$). Note: These are empirical fits to this specific test system; scaling may differ for other signals]**

## 6.6 Coverage and Failure Analysis

### 6.6.1 Method Coverage Summary

**Full coverage (56/56 configurations):** 16 methods

- Gaussian Processes (4), Rational (2), Spectral (5), Local Polynomial (1), Regularization (2), Other (2)

**Partial coverage:** 8 methods

- Orders 0–1 only (2 methods): Central-FD, TVRegDiff-Julia — library/implementation constraints
- Orders 0–5 only (6 methods): Dierckx-5, ButterworthSpline_Python, RKHS_Spline_m2_Python, fourier, fourier_continuation, others — degree/capability limits

### 6.6.2 Failure Mode Documentation

**Catastrophic failures** (nRMSE $> 10^6$ or NaN/Inf):
Occurred in **[TODO: Count configurations with catastrophic failure across all methods]** configurations
**Most common failure patterns:**

1. High-order derivatives ($\geq 6$) with high noise ($\geq 2\%$)
2. Rational approximation methods at orders $\geq 3$
3. Finite difference methods at any order with noise $\geq 10^{-3}$

**Failure causes** (inferred from implementation):

- Numerical instability in high-order rational function derivatives
- Noise amplification in finite difference stencils
- Ill-conditioning in kernel/covariance matrices
- Regularization parameter optimization collapse

## 6.7 Statistical Uncertainty

As documented in Section **??**, $n = 3$ trials provides insufficient power for formal hypothesis testing. The following interpretive guidelines apply to all results presented:

**Confidence intervals:** Where shown (Figures **??**, **??**), 95% CIs are extremely wide (half-width $\approx 2.48\times$ SD). Overlapping CIs do **not** imply "no difference"; non-overlapping CIs provide **moderate evidence** of difference, not statistical significance.

**Method rankings:** Treat as **exploratory descriptive summaries**, not definitive statistical statements. Methods differing by $< 2\times$ in nRMSE should be considered comparable given sample size limitations.

**Robust patterns:** Findings that hold across all 3 trials (e.g., consistent method ordering, categorical performance differences exceeding $10\times$) represent more reliable patterns than specific numerical values.

### 6.8 Summary of Key Findings

1. **Derivative order is the primary difficulty driver:** Performance degrades systematically with increasing order across all methods and noise levels
2. **Method category stratification:** Clear performance hierarchy emerges by category, with Gaussian Processes and spectral methods generally outperforming splines, which outperform finite differences (see Figures **??**–**??** and Table **??**)
3. **Coverage bias is substantial:** Partial-coverage methods appear artificially superior in naive rankings; fair comparison requires coverage normalization
4. **Noise amplification scales with order:** High-order derivatives ($\geq 4$) exhibit extreme noise sensitivity; even best methods struggle at noise $\geq 2\%$
5. **No universal best method:** Optimal choice depends on derivative order, noise level, and computational budget (Figure **??** Pareto frontier)
6. **Statistical power limitations:** Specific numerical rankings should be interpreted cautiously due to $n = 3$ sample size

## 7 Discussion

This section interprets the experimental results, explains unexpected findings, and provides context for method selection.

### 7.1 Why Gaussian Processes Excel

Gaussian Process methods (particularly GP-Julia-AD) demonstrated strong performance across all derivative orders and noise levels tested in this benchmark.

#### 7.1.1 Theoretical Foundation

**Optimality under Gaussian assumptions:**

- GP regression is the Bayes-optimal estimator when both the prior over functions and the noise are Gaussian
- Derivative estimation via kernel differentiation is closed-form—no additional approximation beyond the GP itself
- The posterior predictive distribution provides principled uncertainty quantification (not evaluated in this benchmark)

**Derivative computation:** The GP derivative is obtained by differentiating the kernel function:

$$\mathbb{E}[f^{(n)}(x^*)|\text{data}] = \left[\frac{\partial^n k(x^*, x_1)}{\partial x^{*n}}, \ldots, \frac{\partial^n k(x^*, x_n)}{\partial x^{*n}}\right] (K + \sigma^2 I)^{-1} y \tag{31}$$

This closed-form expression avoids iterative differentiation or numerical differentiation of the fitted function.

### 7.1.2 Practical Advantages

**Automatic regularization:**

- Hyperparameters (length scale $\ell$, noise variance $\sigma_n^2$) are optimized via Maximum Likelihood Estimation
- Kernel smoothness implicitly controls overfitting
- No manual tuning of smoothing parameters required (unlike splines or regularization methods)

**Graceful degradation:**

- As noise increases, GP automatically adjusts effective smoothing via the $\sigma_n^2$ parameter
- No catastrophic failures observed across all 56 configurations

### 7.1.3 When GP May Not Be Optimal

**Large datasets ($n > 1000$):**

- $O(n^3)$ computational cost for Cholesky factorization becomes prohibitive
- Sparse/inducing-point approximations can reduce cost to $O(nm^2)$ where $m \ll n$
- Alternative: Switch to $O(n \log n)$ spectral methods (Fourier-Interp) if signal is smooth and periodic

**Non-Gaussian noise:**

- GP optimality guarantees assume Gaussian noise
- For heavy-tailed or structured noise, robust alternatives (e.g., Student-$t$ process, TVRegDiff) may be preferable

## 7.2 AAA Rational Approximation Failure

**Contrary to initial expectations** based on literature performance for interpolation, AAA-HighPrec fails catastrophically for derivative orders $\geq 3$, even at near-zero noise levels ($10^{-8}$).

### 7.2.1 Observed Failure Pattern

Table **??** documents AAA-HighPrec performance at near-zero noise ($10^{-8}$)—the most favorable condition—showing catastrophic exponential growth beginning at order 3.

**Key observations from Table ??:**

- **Orders 0–2:** Exceptional performance (nRMSE $\sim 10^{-9}$ to $10^{-4}$), competitive with GP methods
- **Order 3:** nRMSE $= 0.097$—degradation begins despite near-perfect data
- **Order 4:** nRMSE $= 57.9$—catastrophic failure (error $> 5700\%$ of typical derivative magnitude)
- **Orders 5–7:** nRMSE grows exponentially to $10^{10}$—complete algorithm breakdown
- **Critical finding:** Failure persists with high-precision BigFloat arithmetic (256-bit), indicating algorithmic (not numerical precision) issue

Table 7: AAA-HighPrec Catastrophic Failure at Near-Zero Noise ($10^{-8}$)

| Derivative Order | nRMSE | Status |
|---|---|---|
| 0 | $9.75\times10^{-9}$ | Excellent |
| 1 | $1.25\times10^{-6}$ | Excellent |
| 2 | $2.64\times10^{-4}$ | Good |
| 3 | $9.68\times10^{-2}$ | Degradation begins |
| 4 | $5.79\times10^{1}$ | Catastrophic failure |
| 5 | $4.09\times10^{4}$ | Complete breakdown |
| 6 | $2.97\times10^{7}$ | Complete breakdown |
| 7 | $2.13\times10^{10}$ | Complete breakdown |

### 7.2.2 Hypothesized Mechanisms

**Potential causes** (subject to further investigation):

1. **Spurious pole proximity:** Rational approximants can develop poles near evaluation points during greedy selection. High-order derivatives of $r(z) = p(z)/q(z)$ near poles grow factorially, amplifying even tiny errors.
2. **Barycentric differentiation instability:** Differentiating the barycentric form $\frac{d^n}{dz^n} [\sum w_i f_i/(z - z_i)/ \sum w_i/($ involves repeated quotient rule applications. Each differentiation compounds numerical error.
3. **Factorial growth in derivative magnitude:** For $r(z) \sim 1/(z - z_0)$, the $n$-th derivative scales as $n!/(z - z_0)^{n+1}$. Even well-separated poles can cause overflow/underflow at high orders.

**Note:** These are hypotheses based on known properties of rational approximation. Rigorous analysis would require detailed examination of AAA support point selection, pole locations, and barycentric weight magnitudes.

### 7.2.3 Practical Implications

**Recommendation:** Restrict AAA-HighPrec use to:

- **Orders 0–2 only**
- **Noise** $\leq 10^{-8}$
- Applications where ultra-high accuracy at low orders is critical

**Do NOT use AAA for:**

- General-purpose derivative estimation (orders $\geq 3$)
- Noisy data (noise $> 10^{-8}$)
- Production systems requiring reliability across varying conditions

**Alternative:** Use GP-Julia-AD or Fourier-Interp for robust high-order derivatives.

## 7.3 Adaptive Hyperparameter Selection: A Paradigm Shift

The 9 new adaptive methods represent a fundamental shift from fixed, pre-tuned hyperparameters to data-driven automatic selection. Results demonstrate that adaptive methods consistently achieve lower mean nRMSE than their fixed-parameter counterparts (descriptive finding; n=3 trials insufficient for formal hypothesis testing).

### 7.3.1 Why Adaptive Methods Outperform Fixed Methods

**Fixed-parameter limitations:**

- Traditional methods use hyperparameters tuned on validation data at specific noise levels
- Performance degrades when applied to different noise regimes
- No mechanism to adjust to varying derivative orders (higher orders amplify noise differently)

**Adaptive advantages:**

1. **Noise-aware selection:** Wavelet MAD and diff2 methods estimate $\hat{\sigma}$ from data, enabling noise-dependent hyperparameter choices
2. **Automatic bias-variance trade-off:** GCV and AICc automatically balance model complexity vs. fit quality
3. **Order-agnostic robustness:** Same adaptive criterion works across derivative orders 0–7

### 7.3.2 Key Findings from Adaptive Methods

**Fourier-GCV outperforms fixed Fourier methods:**
Fourier-GCV achieved 4th overall ranking (by mean nRMSE) by automatically selecting the optimal number of harmonics $M^*$ for each configuration. In contrast, Fourier-Interp uses fixed filter_frac=0.4 which is optimal for some conditions but suboptimal for others.

**Adaptive behavior:** At low noise levels, Fourier-GCV retains more harmonics to preserve signal detail. At high noise levels, it aggressively reduces harmonics to suppress noise amplification. This noise-adaptive strategy enables robust performance across the full noise spectrum tested.

**Chebyshev-AICc dramatically improves high-order performance:**
Fixed-degree Chebyshev polynomials struggle at high derivative orders because optimal degree varies with order. Chebyshev-AICc selects degree $d^*$ adaptively via AICc, preventing both underfitting (low $d$, high bias) and overfitting (high $d$, high variance with noise).

**Observed pattern:** At order 0–2, AICc selects $d \approx$ 5–7. At order 5–7, AICc increases to $d \approx$ 10–12 to capture high-frequency derivative content.

**ad_trig_adaptive vs. ad_trig:**
GCV-based harmonics selection for trigonometric basis functions (ad_trig_adaptive) outperforms fixed-parameter ad_trig, particularly at moderate noise levels ($10^{-3}$ to $10^{-2}$).

**JAX AD enables arbitrary-order AAA derivatives (but doesn't solve instability):**
AAA-JAX methods successfully compute orders 6–7 via automatic differentiation, previously unavailable for AAA. However, the fundamental high-order instability of AAA rational approximants persists. This demonstrates that derivative computation method (analytic vs. AD) is distinct from approximation quality—stable differentiation cannot compensate for poor underlying approximants.

### 7.3.3 Noise Estimation Method Comparison

**Wavelet MAD vs. second-order difference:**

Comparing AAA-Python-Adaptive-Wavelet and AAA-Python-Adaptive-Diff2 reveals trade-offs:

- **Wavelet MAD:** More robust for oscillatory signals (isolates noise in high-frequency wavelet coefficients). Preferred for Lotka-Volterra which is periodic.
- **Diff2:** Simpler, faster ($O(n)$ vs. wavelet transform overhead). May underestimate noise if signal has high second-derivative content.

**Observed performance:** On Lotka-Volterra, wavelet MAD provided slightly lower mean nRMSE than diff2 for AAA-Python-Adaptive methods, consistent with better noise isolation for oscillatory signals.

### 7.3.4 Computational Cost vs. Adaptivity Trade-Off

**Adaptivity overhead:**

- GCV selection: Tests $O(\log n)$ candidate harmonics, each requiring FFT ($O(n \log n)$). Total: $O(n \log^2 n)$
- AICc selection: Fits $O(1)$ candidate polynomial degrees, each $O(n)$. Total: $O(n)$
- Wavelet MAD: One wavelet decomposition, $O(n)$

**Practical impact:** Adaptive overhead is modest—Fourier-GCV is still 10–15$\times$ faster than GP methods despite testing multiple $M$ values.

### 7.3.5 When Fixed Methods May Suffice

**Adaptive methods are not always necessary:**

1. **Homogeneous noise regime:** If all data has similar noise level (e.g., instrument noise with known $\sigma$), pre-tuned fixed hyperparameters may suffice
2. **Real-time constraints:** Adaptive selection overhead may be prohibitive for streaming/real-time applications
3. **Orders 0–1 only:** Performance gap between adaptive and fixed is smallest at low orders

**Recommendation:** Use adaptive methods for:

- Unknown or varying noise levels
- High derivative orders ($\geq 3$)
- General-purpose tools/libraries where robustness matters more than peak performance

## 7.4 Spectral Methods: The Importance of Filtering

Fourier spectral methods demonstrated strong performance, particularly at high derivative orders where many other methods failed.

### 7.4.1 Why Spectral Methods Work

**Differentiation in frequency domain:**

$$\frac{d^n f}{dx^n} = \sum (ik\omega)^n c_k \exp(ik\omega x) \tag{32}$$

Multiplication by $(ik)^n$ in frequency space is exact for band-limited signals. No approximation error from differentiation itself.

**Challenge:** Noise amplification

- High frequencies (large $k$) are amplified by $k^n$
- For $k = 30$, $n = 7$: amplification factor $\approx (30)^7 \approx 2 \times 10^{10}$
- Even tiny high-frequency noise dominates the signal after differentiation

### 7.4.2 The Filter Fraction Trade-Off

**Fourier-Interp uses filter_frac=0.4** (retains lower 40% of spectrum):

- Too aggressive (e.g., 0.2): Over-smooths signal, loses high-frequency features
- Too permissive (e.g., 0.8): Amplifies noise, catastrophic errors at high orders
- Sweet spot: 0.4 (determined via validation testing; Section **??**)

## 7.5 Total Variation Regularization: Scope Limitations

TVRegDiff-Julia performed excellently for smoothing (order 0) but exhibited catastrophic failure for iterative differentiation beyond order 1.

**Why iterative differentiation fails:**

1. Minimize $\|u - y\|^2 + \alpha \mathrm{TV}(u)$ to obtain smoothed $u$
2. Differentiate $u$ via finite differences to obtain $u'$
3. Repeat for higher orders: smooth $u'$, differentiate to get $u''$, etc.

**Error compounding:**

- Each differentiation step introduces approximation error
- Each smoothing step potentially removes signal content
- **Order 1:** One iteration—acceptable (nRMSE $\sim 0.3$)
- **Order 2:** Two iterations—error explodes (nRMSE $\sim 10^{28}$)
- **Orders 3+:** Complete breakdown (NaN/Inf)

## 7.6 Finite Differences: When and Why They Fail

Central finite differences are simple and fast but exhibit catastrophic noise amplification at high orders or moderate noise.

**Noise amplification mechanism:**

Example: Second derivative via 3-point stencil:

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{33}$$

**With additive noise $\varepsilon \sim \mathcal{N}(0, \sigma)$:**

- Numerator error: $\sqrt{[\sigma^2 + 4\sigma^2 + \sigma^2]} = \sqrt{6}\sigma$ (uncorrelated noise)
- Denominator: $h^2 = 0.01$ (for $h = 0.1$)
- Total error: $\sqrt{6}\sigma/0.01 \approx 245\sigma$

For 1% noise ($\sigma = 0.01 \times \text{std(signal)}$), derivative error is $\sim 2.45 \times \text{std(signal)}$—larger than the signal itself.

## 7.7 Cross-Language Implementation Quality

Three methods were excluded due to cross-language performance discrepancies exceeding 50× despite parameter parity efforts (Section **??**).

**Lessons for benchmark studies:**

**Parameter parity is insufficient:** Matching documented parameters does not guarantee implementation equivalence. Subtle differences in numerical precision, boundary conditions, and library-internal defaults can lead to dramatic performance differences.

**Implementation as a method characteristic:** For practitioners, "which method is best?" includes "which implementation?" A theoretically excellent algorithm with a buggy or numerically unstable implementation provides no practical value.

## 7.8 Coverage Bias and Fair Comparison

**Observation:** Only 16 of 24 methods (67%) achieved full coverage across all 56 configurations.

**Naive overall ranking bias:** Methods tested only on "easy" configurations (orders 0–1, low noise) appear artificially superior because they are excluded from challenging tests where most methods struggle.

**Fair comparison strategies used:**

1. **Full-coverage rankings** (Table **??**): Restrict comparison to 16 methods tested on all configurations
2. **Per-configuration rankings:** Compare methods within each (order, noise) pair
3. **Coverage transparency:** Report coverage percentage in all tables and figures

## 7.9 Statistical Power and Interpretive Caution

As documented in Sections **??** and **??**, $n = 3$ trials provides:

- Very wide confidence intervals (half-width $\approx 2.48\times$ SD)
- Insufficient power for formal hypothesis testing
- Unstable mean and variance estimates

**Implication:** Specific numerical rankings should be interpreted as **exploratory descriptive summaries**, not definitive statistical statements.

**Robust vs. fragile findings:**

**Robust findings** (high confidence):

- Categorical performance differences (e.g., GP vs FD: 10–100$\times$ nRMSE ratio)
- Consistent ordering across all 3 trials
- Catastrophic failures (nRMSE $> 10^6$ or NaN/Inf)
- Derivative order as primary difficulty driver

**Fragile findings** (interpret cautiously):

- Rankings of methods within $2\times$ nRMSE
- Specific numerical values (e.g., "method A has nRMSE $= 0.25 \pm 0.03$")
- Subtle trends requiring $> 10$ trials to detect

## 7.10 Generalization Beyond Lotka-Volterra

**Critical caveat:** All results are derived from a **single test signal** (Lotka-Volterra prey population) with **one noise model** (additive Gaussian).

**Generalization risks:**

- **System-specific:** Oscillatory dynamics may favor spectral methods; chaotic or discontinuous signals may favor different methods
- **Single-variable bias:** Only prey population tested; predator population may exhibit different noise sensitivity
- **Additive noise only:** Multiplicative, Poisson, or heteroscedastic noise not tested

**Recommended approach for practitioners:**

**Do NOT assume rankings generalize universally.** Instead:

1. Identify 2–3 candidate methods from our results matching your problem characteristics
2. Test on YOUR data with YOUR noise model
3. Use cross-validation or hold-out testing to select the best method for your application

**Our results provide:** A reasonable starting point and elimination of clearly poor choices (e.g., avoid finite differences for noisy high-order derivatives).

### 7.11 Summary: Key Takeaways

1. **GP-Julia-AD is the most reliable all-around choice** for this test system, though computational cost may limit use for $n > 1000$
2. **AAA rational approximation fails catastrophically at orders** $\geq 3$—restrict to orders 0–2 with near-perfect data only
3. **Fourier spectral methods are strong alternatives** for smooth signals, especially at high orders, with proper filtering
4. **Derivative order is the dominant difficulty factor**—noise amplification scales exponentially with order
5. **Method selection depends on context:** No universal "best" method exists; optimal choice varies by derivative order, noise level, and computational budget
6. **Implementation quality matters:** Cross-language discrepancies highlight that algorithms and implementations are distinct
7. **Statistical limitations:** $n = 3$ trials insufficient for fine-grained ranking; treat results as descriptive, not definitive
8. **Generalization caution:** Single-signal study limits applicability; validate on your data before production use

## 8 Practical Recommendations

This section provides actionable guidance for practitioners selecting derivative estimation methods based on experimental findings. Recommendations are organized by derivative order and noise level, with explicit caveats regarding generalization limitations (see Section **??**).

### 8.1 Master Recommendation Table

Table **??** provides quick-reference method selection guidelines based on derivative order and noise level. All recommendations assume the Lotka-Volterra test system characteristics (smooth, oscillatory, additive Gaussian noise); validation on your specific data is strongly advised.

**Critical caveat:** AAA-HighPrec *catastrophically fails* at orders $\geq 3$ (nRMSE $> 10^7$), even at near-perfect noise levels ($10^{-8}$). Restrict AAA use to orders 0–2 at noise $\leq 10^{-8}$ only.

### 8.2 Detailed Recommendations by Scenario

#### 8.2.1 Near-Noiseless Data ($\leq 10^{-8}$)

**Orders 0–2:**

- **Primary:** GP-Julia-AD—best overall performance across all scenarios (nRMSE $\approx 10^{-9}$ to $10^{-4}$)
- **Alternative:** AAA-HighPrec—*exceptional* at orders 0–2 only (9–2929$\times$ better than GP at very low orders), but catastrophic at orders $\geq 3$

Table 8: Method Selection Recommendations by Derivative Order and Noise Level

| **Order** | **Near-Noiseless** ($\leq 10^{-8}$) | **Low Noise** ($10^{-4}$–$10^{-3}$) | **High Noise** ($10^{-2}$–$5 \times 10^{-2}$) |
|---|---|---|---|
| **0–1** | GP-Julia-AD / AAA-HighPrec / Dierckx-5 | GP-Julia-AD / Dierckx-5 | GP-Julia-AD / TVRegDiff |
| **2** | GP-Julia-AD / AAA-HighPrec ($\leq 10^{-8}$ only) | GP-Julia-AD / Fourier-Interp | GP-Julia-AD only |
| **3–5** | GP-Julia-AD / Fourier-Interp | GP-Julia-AD / Fourier-Interp | GP-Julia-AD (verify nRMSE acceptable) |
| **6–7** | GP-Julia-AD only | GP-Julia-AD only | GP-Julia-AD (expect nRMSE 0.5–1.0) |

- **Fast options:** Dierckx-5, Central-FD (orders 0–1 only)—acceptable if speed is critical and accuracy can be sacrificed

**Orders 3–7:**

- **Only reliable method:** GP-Julia-AD
- **Fast alternative (orders 3–5):** Fourier-Interp—reasonable accuracy at 10–20× speedup
- **DO NOT use AAA:** Catastrophic failures ($10^{7}$–$10^{22}$ nRMSE) even at $10^{-8}$ noise

### 8.2.2 Low Noise ($10^{-4}$ to $10^{-3}$)

**Orders 0–2:**

- **Primary:** GP-Julia-AD (nRMSE $< 0.1$)
- **Fast alternatives:** Fourier-Interp, Dierckx-5—acceptable accuracy at $\approx 20\times$ speedup

**Orders 3–5:**

- **Primary:** GP-Julia-AD (nRMSE $\approx$ 0.15–0.4)
- **Fast alternative:** Fourier-Interp—competitive accuracy, significantly faster
- **Avoid:** Finite differences (nRMSE $> 1$), AAA (unstable)

**Orders 6–7:**

- **Only viable method:** GP-Julia-AD (nRMSE $\approx$ 0.5–0.7)
- **Warning:** High-order derivatives at moderate noise are extremely challenging—verify nRMSE is acceptable for your application before proceeding

### 8.2.3 High Noise ($10^{-2}$ to $5 \times 10^{-2}$)

**Orders 0–1:**

- **Primary:** GP-Julia-AD (nRMSE $< 0.05$)
- **Alternative:** TVRegDiff-Julia (orders 0–1 only)—edge-preserving smoothing may be advantageous for discontinuous signals

**Orders 2–3:**

- **Primary:** GP-Julia-AD (nRMSE $\approx 0.1$–$0.3$)
- **Fast alternative (if nRMSE $\sim 0.4$ acceptable):** Fourier-Interp
- **Avoid:** Finite differences, AAA (unstable)

**Orders $\geq 4$:**

- **Only viable method:** GP-Julia-AD (nRMSE $\approx 0.3$–$1.0$)
- **Warning:** All methods struggle at high orders with high noise. Consider:
  - Is the high-order derivative estimate reliable enough for your application?
  - Can you reduce noise (additional measurements, filtering) or use lower-order derivatives instead?
  - Can you reformulate the problem to avoid high-order derivatives?

## 8.3 Decision Framework

The following sequential decision process guides method selection:

**Step 1: Identify derivative order requirement**

- **Orders 0–1:** Multiple viable methods; proceed to Step 2
- **Orders 2–3:** Limited methods; proceed to Step 2 with expectation of fewer alternatives
- **Orders 4–5:** Expect GP-Julia-AD or Fourier-Interp only; verify feasibility
- **Orders 6–7:** Use GP-Julia-AD; verify nRMSE $\approx 0.5$–$1.0$ is acceptable for your application

**Step 2: Assess noise level**

- **$< 10^{-6}$:** GP-Julia-AD or AAA-HighPrec (both excellent at orders 0–2)
- **$10^{-4}$ to $10^{-3}$:** GP-Julia-AD (primary), Fourier-Interp (fast alternative)
- **$> 10^{-2}$:** GP-Julia-AD only reliable for orders $> 2$

**Step 3: Evaluate computational constraints**

- **Speed critical and noise $< 10^{-3}$:** Try Fourier-Interp (10–20$\times$ faster than GP)
- **Speed critical and noise $> 10^{-2}$:** No fast alternative maintains acceptable accuracy; use GP-Julia-AD

- **Large datasets ($n > 1000$):** Spectral methods ($\mathrm{O}(n \log n)$) strongly preferred over GP ($\mathrm{O}(n^3)$)

**Step 4: Consider additional requirements**

- **Uncertainty quantification needed:** Use GP-Julia-AD (built-in confidence intervals via posterior variance)
- **Edge preservation needed:** Use TVRegDiff-Julia (orders 0–1 only)
- **Multivariate extensions anticipated:** GP methods generalize naturally via product kernels; AAA and spectral methods require additional development

**Step 5: Validate on your data**

- **Critical:** These recommendations are derived from a *single test system* (Lotka-Volterra) with *additive Gaussian noise*
- Test top 2–3 candidate methods on your actual data using cross-validation or hold-out testing
- Verify nRMSE is acceptable for your specific application before production deployment

## 8.4 Common Pitfalls and Misconceptions

**Pitfall 1: Using AAA for orders $\geq 3$**

- AAA-HighPrec exhibits catastrophic failure (nRMSE $> 10^7$) at orders $\geq 3$ despite excellent low-order performance
- **Action:** Restrict AAA to orders 0–2 at noise $\leq 10^{-8}$ only

**Pitfall 2: Assuming finite differences are "good enough"**

- Central-FD exhibits severe noise amplification at moderate noise ($\geq 10^{-3}$) and high orders ($\geq 3$)
- **Action:** Avoid finite differences for noisy high-order derivatives; use GP or spectral methods

**Pitfall 3: Ignoring computational scaling**

- GP methods scale as $\mathrm{O}(n^3)$; prohibitive for $n > 1000$ without approximations
- **Action:** For large datasets, use spectral methods ($\mathrm{O}(n \log n)$) or sparse GP approximations

**Pitfall 4: Over-generalizing from benchmarks**

- Performance characteristics depend on signal properties (smoothness, periodicity) and noise model
- **Action:** Treat benchmarks as *starting points*, not definitive rankings; validate on your data

**Pitfall 5: Trusting high-order derivatives blindly**

- Even best methods (GP-Julia-AD) exhibit nRMSE $\approx$ 0.5–1.0 at orders 6–7 with moderate noise
- **Action:** Question whether high-order derivatives are truly needed; consider reformulating the problem

### 8.5 Implementation Guidance

**Recommended software packages** (based on experimental evaluation):

- **GP-Julia-AD:** GaussianProcesses.jl (Julia)—best overall performance
- **Fourier-Interp:** FFTW.jl (Julia) or scipy.fftpack (Python)—fast spectral differentiation
- **TVRegDiff:** Custom Julia implementation (edge-preserving smoothing)
- **AAA-HighPrec:** BaryRational.jl (Julia)—orders 0–2 at low noise only

**Hyperparameter tuning:**

- **GP methods:** Use maximum likelihood estimation for length scale and noise variance (automatic in GaussianProcesses.jl)
- **Fourier-Interp:** `filter_frac = 0.4` (retains lower 40% of spectrum) worked well for smooth signals; tune via validation testing for your data
- **TVRegDiff:** Regularization parameter $\alpha$ requires manual tuning; start with $\alpha = 0.01$ and adjust based on visual inspection

**Validation strategy:**

1. Generate synthetic test cases with known ground truth derivatives
2. Evaluate candidate methods across realistic noise levels for your application
3. Use $k$-fold cross-validation or hold-out testing to assess generalization
4. Select method with best nRMSE on validation set, subject to computational constraints

## 9 Limitations and Future Work

This section consolidates limitations identified throughout Sections **??**–**??** and proposes extensions to address them.

### 9.1 Experimental Design Limitations

#### 9.1.1 Single Test System

**Limitation:** All results derive from a single dynamical system (Lotka-Volterra predator-prey model, prey population $x(t)$ only).

**Implications:**

- Lotka-Volterra produces smooth, periodic, analytic signals—ideal for spectral methods and potentially unrepresentative of rough/discontinuous signals
- Only one variable tested (prey $x(t)$); predator population $y(t)$ may exhibit different noise sensitivity
- Method rankings may change for chaotic dynamics, stochastic processes, or experimental data with measurement artifacts

**Future work:**

- Test on diverse signal classes:
  - Chaotic systems (Lorenz attractor, double pendulum)
  - Discontinuous functions (piecewise polynomials, step functions)
  - Non-smooth signals (Brownian motion, fractional Brownian motion)
  - Real experimental data (e.g., from physics/chemistry measurements)
- Identify signal characteristics (smoothness, periodicity, characteristic scales) that predict method performance
- Develop signal-adaptive method selection criteria

### 9.1.2 Noise Model

**Limitation:** Only additive Gaussian noise tested ($\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, independent across time points).
**Implications:**

- Gaussian Process optimality guarantees assume Gaussian noise—rankings may change for heavy-tailed or structured noise
- Real data often exhibits multiplicative noise, Poisson noise (count data), or heteroscedastic noise (variance depends on signal magnitude)
- Correlated noise (AR/MA processes) not tested—may favor methods with explicit temporal correlation models

**Future work:**

- Evaluate alternative noise models:
  - Multiplicative noise: $y_i = f(t_i)(1 + \epsilon_i)$
  - Poisson noise: common in photon counting, chemical reaction data
  - Student-t noise: heavy-tailed, tests robustness to outliers
  - Correlated noise: AR(1), colored noise
- Test robust methods (TVRegDiff, Student-t processes) on heavy-tailed noise
- Develop noise-adaptive method selection guidelines

### 9.1.3 Statistical Power

**Limitation:** Only $n = 3$ trials per configuration—insufficient for formal hypothesis testing or stable variance estimates (Section **??**).
**Implications:**

- 95% confidence intervals have half-width $\approx 2.48\times$ SD (extremely wide with df=2)
- Rankings of methods within $2\times$ nRMSE should be interpreted cautiously
- Subtle performance differences cannot be detected reliably

- No statistical significance testing performed (insufficient power)

**Future work:**

- Increase to $n \geq 10$ trials for robust statistical inference
- Perform formal pairwise comparisons (e.g., Wilcoxon signed-rank test) with correction for multiple comparisons
- Compute bootstrap confidence intervals for method rankings
- Quantify "zone of indistinguishability" where methods cannot be reliably distinguished given sample size

## 9.2 Method Coverage and Implementation Limitations

### 9.2.1 Cross-Language Implementation Discrepancies

**Limitation:** 3/27 candidate methods excluded due to $> 50\times$ performance discrepancies between Julia and Python implementations despite parameter parity attempts (Section **??**).

**Implications:**

- Implementation quality is a method characteristic, not just algorithmic performance
- "Best method" depends on which implementation—practitioners need software-specific guidance
- Undocumented library defaults, numerical precision choices, and optimization strategies can dominate performance

**Future work:**

- Comprehensive cross-language validation: Match *all* parameters, not just documented ones
- Implement reference versions in multiple languages with verified parameter parity
- Document exact software versions, commits, and build configurations for reproducibility
- Develop methodology for fairly comparing methods across languages/implementations

### 9.2.2 Partial Coverage Bias

**Limitation:** Only 16/24 methods achieve full 56/56 configuration coverage; 8 methods have library/degree limitations restricting testable orders (Section **??**).

**Implications:**

- Methods tested only on "easy" configurations appear artificially superior
- Fair comparison requires restricting to full-coverage methods (excludes potentially useful specialized methods)
- Coverage transparency critical but often unreported in prior benchmarks

**Future work:**

- Develop imputation strategies for missing configurations (risky—may hide genuine failures)
- Weight rankings by configuration difficulty (requires defining difficulty metric)
- Create separate rankings for full-coverage vs. specialized methods

## 9.3 Computational and Scalability Limitations

### 9.3.1 Small Problem Size

**Limitation:** Fixed $n = 101$ sample points—modest scale, may not reflect large-data or small-data regimes.

**Implications:**

- GP methods ($O(n^3)$) computationally feasible at $n = 101$ but prohibitive for $n > 1000$
- Spectral methods ($O(n \log n)$) show modest absolute timing at this scale; advantages emerge only at large $n$
- Small-$n$ behavior ($n < 50$) not tested—may favor different methods (more regularization needed)

**Future work:**

- Benchmark across problem sizes: $n \in \{50, 100, 500, 1000, 5000\}$
- Evaluate sparse GP approximations (inducing points, spectral approximations) for large-scale problems
- Test online/streaming methods for real-time applications
- Develop scaling guidelines: Which method for which problem size?

### 9.3.2 Uniform Grid Assumption

**Limitation:** All evaluations on uniform grids—non-uniform sampling not tested.

**Implications:**

- Real data often has irregular sampling (missing data, adaptive sampling)
- GP and spline methods handle non-uniform grids naturally; FFT-based methods require interpolation
- Adaptive sampling strategies not evaluated

**Future work:**

- Test on non-uniform grids with varying density
- Evaluate adaptive sampling strategies (higher density near high-curvature regions)
- Assess robustness to missing data

## 9.4 Generalization and Applicability

### 9.4.1 Single-Application Domain

**Limitation:** Results from ODE system may not generalize to other domains (PDEs, experimental physics/chemistry data, financial time series).

**Future work:**

- Domain-specific benchmarks:

  - PDEs: Navier-Stokes, heat equation (higher-dimensional extensions)
  - Experimental data: Real sensor measurements with calibration artifacts
  - Stochastic processes: Financial time series with heavy tails and heteroscedasticity

- Develop domain-specific method recommendations

#### 9.4.2 Multivariate Extensions

**Limitation:** Only univariate (scalar-valued) functions tested—multivariate derivative estimation not addressed.

**Future work:**

- Extend to multivariate functions: $f : \mathbb{R}^d \to \mathbb{R}$ (partial derivatives, gradients, Hessians)
- Evaluate GP methods with product kernels
- Test tensor-product splines
- Assess curse of dimensionality: How do methods scale with input dimension $d$?

### 9.5 Summary of Recommended Extensions

**Highest priority** (addresses most critical limitations):

1. Increase trial count to $n \geq 10$ for statistical rigor
2. Test on 5–10 diverse signals (chaotic, discontinuous, experimental data)
3. Evaluate alternative noise models (multiplicative, Poisson, heavy-tailed)

**Medium priority** (enhances generalizability):

1. Scale study: Problem sizes $n \in \{50, 100, 500, 1000, 5000\}$
2. Cross-language validation with complete parameter documentation
3. Non-uniform grid and missing data robustness testing

**Long-term extensions** (new capabilities):

1. Multivariate derivative estimation (gradients, Hessians)
2. Online/streaming methods for real-time applications
3. Domain-specific benchmarks (PDEs, experimental physics, finance)

## 10 Conclusion

This paper presents the first comprehensive benchmark of derivative estimation methods across the full range of practical derivative orders (0–7) and noise levels ($10^{-8}$ to $5 \times 10^{-2}$). We evaluated 33 methods from 7 categories (24 baseline methods plus 9 adaptive hyperparameter selection variants) on 56 configurations, providing fine-grained performance maps to guide method selection.

### 10.1 Key Findings

**1. Adaptive hyperparameter selection consistently achieves lower error than fixed methods**

Among the 9 new adaptive methods, **Fourier-GCV** emerged as the top performer, ranking 4th overall by mean nRMSE (descriptive finding; n=3 trials insufficient for statistical significance testing). Adaptive methods using GCV (Generalized Cross-Validation) for harmonics selection and AICc for polynomial degree selection demonstrated consistently lower mean nRMSE across varying noise levels and derivative orders compared to their fixed-parameter counterparts.

**2. Gaussian Process regression remains the most reliable baseline method**

GP-Julia-AD achieved the best overall performance among baseline methods (mean nRMSE = 0.257 across all configurations) with no catastrophic failures. It maintains usable accuracy even at extreme challenges (orders 6–7 with high noise), though nRMSE $\approx$ 0.5–1.0 indicates fundamental difficulty limits.

**3. JAX automatic differentiation enables arbitrary-order AAA derivatives but doesn't solve high-order instability**

The new AAA-JAX-Adaptive methods successfully compute derivatives up to order 7 via nested automatic differentiation, overcoming the previous limitation of AAA-based methods to orders 0–5. However, the fundamental algorithmic instability of AAA at high orders persists—AAA-JAX methods still exhibit large errors at orders $\geq 3$. The computational cost is significant: order 7 derivatives can take 8–20 minutes per configuration.

**4. AAA rational approximation fails catastrophically at high orders despite adaptive tolerance**

Contrary to literature expectations based on interpolation performance, AAA-HighPrec exhibits catastrophic failure (nRMSE $> 10^7$) at orders $\geq 3$, even at near-perfect noise levels ($10^{-8}$). This failure persists despite high-precision (BigFloat) arithmetic and adaptive tolerance selection based on noise estimation, indicating an algorithmic limitation rather than numerical precision or hyperparameter issue. **Recommendation:** Restrict AAA use to orders 0–2 at noise $\leq 10^{-8}$ only.

**5. Fourier spectral methods are strong alternatives for smooth signals**

Fourier-Interp achieves competitive accuracy at 10–20× speedup compared to GP methods, particularly effective at high orders (5–7) where many methods fail. The new Fourier-GCV adaptive variant outperforms the fixed-parameter Fourier-Interp by automatically selecting the optimal number of harmonics via GCV, demonstrating robustness across noise levels.

**6. Noise estimation quality affects AAA performance**

Comparing AAA-Python-Adaptive-Wavelet and AAA-Python-Adaptive-Diff2 reveals that noise estimation method (wavelet MAD vs. second-order difference) significantly impacts AAA tolerance selection and subsequent performance. Wavelet MAD tends to provide more robust estimates for oscillatory signals like Lotka-Volterra.

**7. Derivative order is the dominant difficulty factor**

Performance degrades systematically for all methods (both baseline and adaptive) as derivative order increases, with order 3 representing a critical transition point where many methods begin failing. High-order derivatives (6–7) remain extremely challenging even for best methods—all evaluated methods exhibit nRMSE $> 0.5$ at these orders with noise $\geq 10^{-2}$. **Practical recommendation:** For production systems requiring robustness, limit derivative estimation to orders 0–4; orders 5–7 are computationally expensive and exhibit degraded accuracy.

**8. Implementation quality is a method characteristic**

Three methods (GP-Julia-SE, TVRegDiff_Python, SavitzkyGolay_Python) were excluded due to $> 50\times$ performance discrepancies between Julia and Python implementations despite parameter parity attempts. This finding highlights that algorithmic excellence alone is insufficient—robust, numerically stable implementation is essential.

## 10.2 Practical Impact

**For practitioners:** Section **??** provides immediately actionable guidance:

- Master recommendation table (Table **??**) maps (derivative order, noise level) $\rightarrow$ optimal method(s)
- Decision framework balances accuracy, speed, and problem size constraints
- Common pitfalls documented to avoid catastrophic failures

**For researchers:** This study identifies fundamental performance limitations and unexpected failure modes, suggesting directions for algorithmic improvements:

- High-order rational approximation differentiation remains an open problem
- Efficient GP approximations for large-scale problems ($n > 1000$) needed
- Adaptive spectral filtering strategies could improve noise robustness

## 10.3 Limitations and Generalization

Results are derived from a single test system (Lotka-Volterra) with additive Gaussian noise and n=3 trials—sufficient for exploratory method comparison but not definitive statistical ranking. Key caveats:

- **Signal-specific:** Lotka-Volterra is smooth and periodic, favoring spectral methods. Rough/discontinuous signals may yield different rankings.
- **Noise-model-specific:** Results assume additive Gaussian noise. Multiplicative, Poisson, or heavy-tailed noise may favor different methods.
- **Statistical uncertainty:** n=3 trials provides only descriptive evidence. Methods differing by $< 2\times$ in nRMSE should be considered comparable.

**Critical recommendation:** Use this benchmark as a starting point to identify 2–3 candidate methods, then validate on *your specific data* using cross-validation or hold-out testing before production deployment.

## 10.4 Future Directions

Highest-priority extensions (Section **??**):

1. Test on diverse signals (chaotic systems, discontinuous functions, experimental data)
2. Increase to n $\geq$ 10 trials for statistical rigor
3. Evaluate alternative noise models (multiplicative, Poisson, heavy-tailed)
4. Scale study across problem sizes ($n \in \{50, 100, 500, 1000, 5000\}$)
5. Multivariate extensions (gradients, Hessians for $f : \mathbb{R}^d \rightarrow \mathbb{R}$)

### 10.5 Closing Remarks

Derivative estimation from noisy data is fundamentally ill-posed, and this challenge intensifies exponentially with derivative order. No universal "best" method exists—optimal choice depends on derivative order, noise level, computational budget, and signal characteristics.

This benchmark provides the first systematic performance map across the parameter space, revealing both opportunities (GP methods remain viable even at order 7) and fundamental limits (all methods struggle beyond order 5 with noise $> 10^{-2}$). We hope these findings guide practitioners toward appropriate method selection and inspire researchers to develop next-generation algorithms addressing identified performance gaps.

**Data and code availability:** All experimental data, method implementations, and analysis scripts are available at **[TODO: Add repository URL upon publication]**.

## Acknowledgments

We thank the developers of GaussianProcesses.jl, Dierckx.jl, FFTW.jl, and scikit-learn for their open-source implementations.

## A Complete Method Specifications

**[TODO: Appendix A: Complete parameter specifications for all 24 methods]**

## B Detailed Results Tables

**[TODO: Appendix B: Full results tables (method $\times$ order $\times$ noise) for reference]**

## C Reproducibility Checklist

**[TODO: Appendix C: Complete reproducibility checklist with all software versions, hardware specs, random seeds, and data availability statements]**