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

Author Names[*]

November 6, 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 31 derivative estimation methods evaluated across 8 derivative orders (0–7), 7 noise levels, and 3 distinct dynamical systems. Our goal is to provide clear, actionable guidance for practitioners.

Key findings include: (1) Gaussian Process Regression (GPR) provides the most robust and accurate performance, consistently ranking as the top method in both low- and high-noise regimes. (2) The optimal method depends critically on the use case: splines like Dierckx-5 are highly competitive for low-noise data, while spectral methods offer a compelling balance of speed and accuracy, and filters like Savitzky-Golay provide a robust, computationally cheap baseline. (3) Derivative order is the dominant difficulty factor, with performance degrading systematically as order increases. We also provide a practitioner's guide to method selection based on the trade-off between speed and accuracy. Our full methodology, analysis, and code are provided to ensure transparency and reproducibility.

Contents

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

1 Introduction

Derivative estimation from noisy data is a fundamental and notoriously ill-posed problem spanning computational science, engineering, and data analysis. Small noise perturbations in the input signal can produce arbitrarily large errors in derivative estimates, a challenge that intensifies rapidly with the derivative order. Despite this, reliable derivatives are essentialin many settings, including:

- **Dynamical Systems Identification:** Inferring governing equations from time-series or field data requires accurate derivative estimates to identify differential equations.
- **Signal Processing:** Edge detection, feature extraction, and change-point analysis all depend on robust derivative computation.
- **Control Systems:** Model-predictive 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, such as reaction rates from concentration data or acceleration from position measurements.

Our investigation is motivated by challenges encountered in parameter estimation for Ordinary Differential Equations (ODEs). Techniques in this domain often require evaluating expressions that contain high-order derivatives of the system's state variables, which are derived from noisy time-series data. The recurring difficulty in finding a reliable method for this step prompted this systematic benchmark study.

This background also suggests that ODEs provide a uniquely suitable testbed for such a study. The structure of an ODE system, $\dot{\mathbf{x}} = f(\mathbf{x})$, provides a natural and computable source of high-precision ground-truth data. Higher-order derivatives can be generated by repeatedly differentiating the system's equations, e.g., $\ddot{\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}}\dot{\mathbf{x}}$. This process, performed symbolically or via system augmentation, allows for the creation of a validation dataset without resorting to a separate numerical differentiation method, thus avoiding circularity. The methods benchmarked are therefore tested against the very class of systems for which they are often intended.

Our prior work demonstrated that for noiseless data, the combination of rational function approximation (specifically, the AAA algorithm) followed by automatic differentiation provides highly accurate derivatives. The present study was born from the challenge of extending this success to real-world, noisy data, a necessary step for applying these techniques to practical problems in science and engineering.

1.1 The Need for a Comprehensive Benchmark

Although derivative estimation is common, systematic comparative evaluation of derivative estimation methods is lacking in the literature. Practitioners seeking to choose a method are faced with a confusing landscape of options, and prior studies often suffer from several limitations:

1. **Limited Scope:** Studies typically evaluate a small handful of methods on low-order derivatives (first or second), neglecting the high-order derivatives where method performance often diverges dramatically.

2. **Single-Noise Regime:** Many studies test either noiseless or high-noise cases, missing the crucial performance transitions that occur across different noise levels.
3. **Incomplete Coverage Transparency:** Methods that fail at high orders or under high noise are often simply excluded from results without clear documentation, obscuring practical limits.

This study was designed to address these gaps by providing a comprehensive, transparent, and reproducible benchmark across a wide range of methods, derivative orders, and noise levels, with the goal of providing clear, actionable guidance for practitioners.

1.2 An Overview of the Present Study

This paper presents the findings of a comprehensive benchmark of derivative estimation methods. We adopt a narrative structure to document our investigation, which began with broad hypotheses and concluded with a specific, data-driven recommendation. By systematically eliminating certain classes of methods—some on the basis of empirical failure, others due to theoretical limitations—we narrowed the field to a small group of effective contenders.

The performance of these final methods was evaluated across three distinct dynamical systems and a wide spectrum of noise levels, leading to a clear and practical conclusion. Our aim is to provide the reader with not just a ranking, but an understanding of *why* certain methods succeed where others fail, and a clear guide for selecting the appropriate tool for their own work.

2 Summary of Key Findings

Before detailing our experimental journey, we present the primary conclusions of this study. After a comprehensive evaluation across a range of methods, three distinct dynamical systems, and a sweep of noise levels, a clear verdict emerges. Performance was analyzed in two regimes: a "low-noise" regime ($\leq 0.1\%$ noise) where precision is paramount, and a "high-noise" regime (1-2% noise) where robustness is key.

The table below summarizes the performance of contender methods that demonstrated full data coverage for derivative orders 0 through 5. Methods are sorted by their overall average rank, giving equal weight to performance in both low- and high-noise regimes. Averages are calculated over orders 0-5.

Our principal findings are as follows:

1. **Gaussian Process Regression (GPR) is the most robust and accurate method overall.** The Julia GPR implementation (`GP-Julia-AD`) and the improved Python variants (`GP-RBF-*`) are the clear winners, consistently occupying the top ranks in both low and high-noise regimes.
2. **The optimal method depends on the noise level and derivative order.** While GPR is the best all-arounder, splines like `Dierckx-5` offer excellent precision in low-noise environments, making them a top choice for cleaner data, particularly at modest derivative orders. In the high-noise regime, the filtering-based `Savitzky-Golay` provides a computationally cheap and highly effective alternative, ranking solidly in the top half of contenders.
3. **Theoretical limits matter.** Many common low-degree spline methods are, by definition, incapable of representing high-order derivatives, limiting their applicability.

Table 1: Contender Method Performance Summary (Orders 0-5)

Method	**Avg. Rank (Overall)**	**Avg. Rank (Low Noise)**	**Avg. nRMSE (Low Noise)**	**Avg. Rank (High Noise)**	**Avg. nRMSE (High Noise)**
GP-Julia-AD	3.2	2.6	0.055	3.8	0.314
GP-RBF-Python	3.4	3.9	0.057	3.0	0.314
GP-RBF-Iso-Python	3.5	3.9	0.057	3.1	0.314
GP-RBF-Mean-Py	3.7	3.6	0.057	3.7	0.314
PyNumDiff-SavGol-Tuned	10.0	9.4	0.180	10.7	1.607
Savitzky-Golay-Fixed	11.5	12.8	0.192	10.1	1.162
Dierckx-5	11.7	10.3	0.196	13.2	0.554
SG-Package-Fixed	12.7	11.1	0.256	14.3	6.529
Fourier-Continuation	13.0	16.5	0.435	9.6	0.473
SG-Package-Hybrid	13.1	10.6	0.500	15.5	>10
Fourier-Interp	13.4	13.0	0.426	13.9	2.461
Savitzky-Golay-Adaptive	13.5	10.8	0.168	16.1	3.468
SG-Package-Adaptive	14.2	9.7	0.515	18.6	>10
Fourier-GCV	14.4	17.2	0.464	11.6	0.498
FFT-Adaptive-Julia	14.7	15.5	0.489	13.8	0.603
Fourier	14.7	17.7	0.520	11.8	0.559
PyNumDiff-Spectral-Auto	15.2	12.8	0.396	17.6	8.323
FFT-Adaptive-Py	16.0	17.4	0.742	14.6	0.618
Fourier-Cont-Adaptive	16.0	19.2	0.569	12.9	0.575
PyNumDiff-Spectral-Tuned	16.5	18.3	0.626	14.7	0.628
AAA-Adaptive-Wavelet	18.9	15.5	>10	22.2	>10
TVRegDiff_Python	20.1	19.9	8.565	20.3	>10
GSS	20.6	23.0	0.742	18.2	0.742
AAA-Adaptive-Diff2	20.9	19.7	>10	22.1	>10
AAA-LowPrec	21.6	16.8	>10	26.4	>10
KalmanGrad_Python	22.2	24.1	0.866	20.4	0.866
Chebyshev-AICc	23.5	25.2	7.501	21.8	7.636
Chebyshev	23.9	25.7	1.689	22.1	1.689

4. **Dedicated packages offer convenient and robust options, but need a differentiation backend.** Purpose built libraries are convenient, but our strongest results come from extending them with either analytic or auto-differentiated derivatives of smoothed data. For this to work best, the library should ideally produce very smooth models, and either expose some of the model internals or be amenable to AD.

The subsequent sections of this paper will detail the experimental journey and analysis that led to these conclusions, providing a narrative account of our investigation and offering a practical framework for method selection.

2.1 Visual Confirmation of Findings

The quantitative results in the summary table are powerfully illustrated by a few key visualizations.

3 Method Catalog and Implementation

This study did not begin with a fixed set of methods. Rather, it followed an exploratory research path, starting with methods known to work on noiseless data and progressively incorporating more sophisticated techniques in response to the challenges posed by noisy signals. This section provides a narrative of that journey, cataloging the methods investigated and detailing the significant implementation work required to create a fair and robust comparison.

3.1 A Composable Framework: Approximation + Automatic Differentiation

A core finding of our study is the power of a composable, two-step framework for numerical differentiation. Rather than relying on bespoke algorithms for each derivative order, the most successful methods, particularly Gaussian Process Regression, implicitly or explicitly follow a simple and highly effective recipe:

1. **Fit an Approximant:** First, a smooth, analytic function (the "approximant") is fitted to the noisy data. This function can be a Gaussian Process, a spline, a spectral representation, or any other differentiable model. The goal of this step is to capture the underlying signal while filtering out noise.
2. **Differentiate via AD:** Second, this fitted function is differentiated using Automatic Differentiation (AD). Because the approximant is a well-defined mathematical function, AD can compute its derivatives to machine precision, free from the truncation and differencing errors that plague finite difference methods.

This "Approximant-AD" pattern is exceptionally powerful because it decouples the problem of noise-robust smoothing from the problem of differentiation. It allows practitioners to leverage the vast and mature ecosystems of both statistical modeling and automatic differentiation, effectively bringing the full power of modern AD frameworks to bear on a classical numerical problem.

3.2 Breadth of the Study

For practitioners, the choice of a specific software package can be as consequential as the choice of the underlying algorithm. To this end, our study intentionally includes multiple implementations of similar or identical algorithms (e.g., several variants of Gaussian Process Regression and Fourier-based methods) to assess whether real-world performance differences arise from implementation

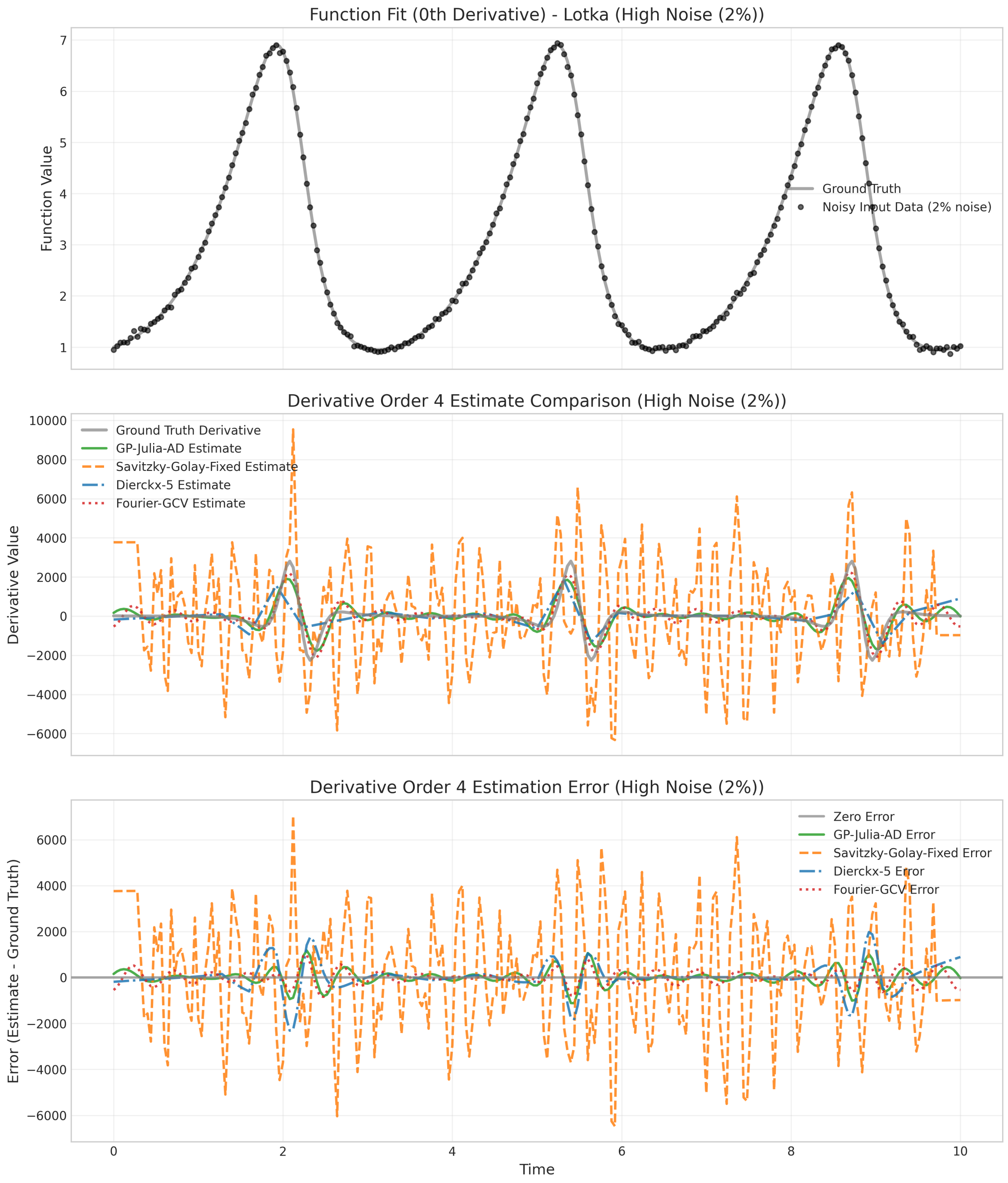


Figure 1: **High-Noise Performance (2% noise, 4th derivative).** Three-panel comparison showing: (top) noisy input data as black points against gray ground truth, emphasizing the challenging signal-to-noise environment; (middle) derivative estimates from four methods representing different algorithmic approaches; (bottom) estimation errors revealing where each method succeeds or fails. The figure shows `GP-Julia-AD`, `Savitzky-Golay-Fixed`, `Dierckx-5`, and `Fourier-GCV`. The error panel reveals that all methods struggle at the trajectory boundaries.

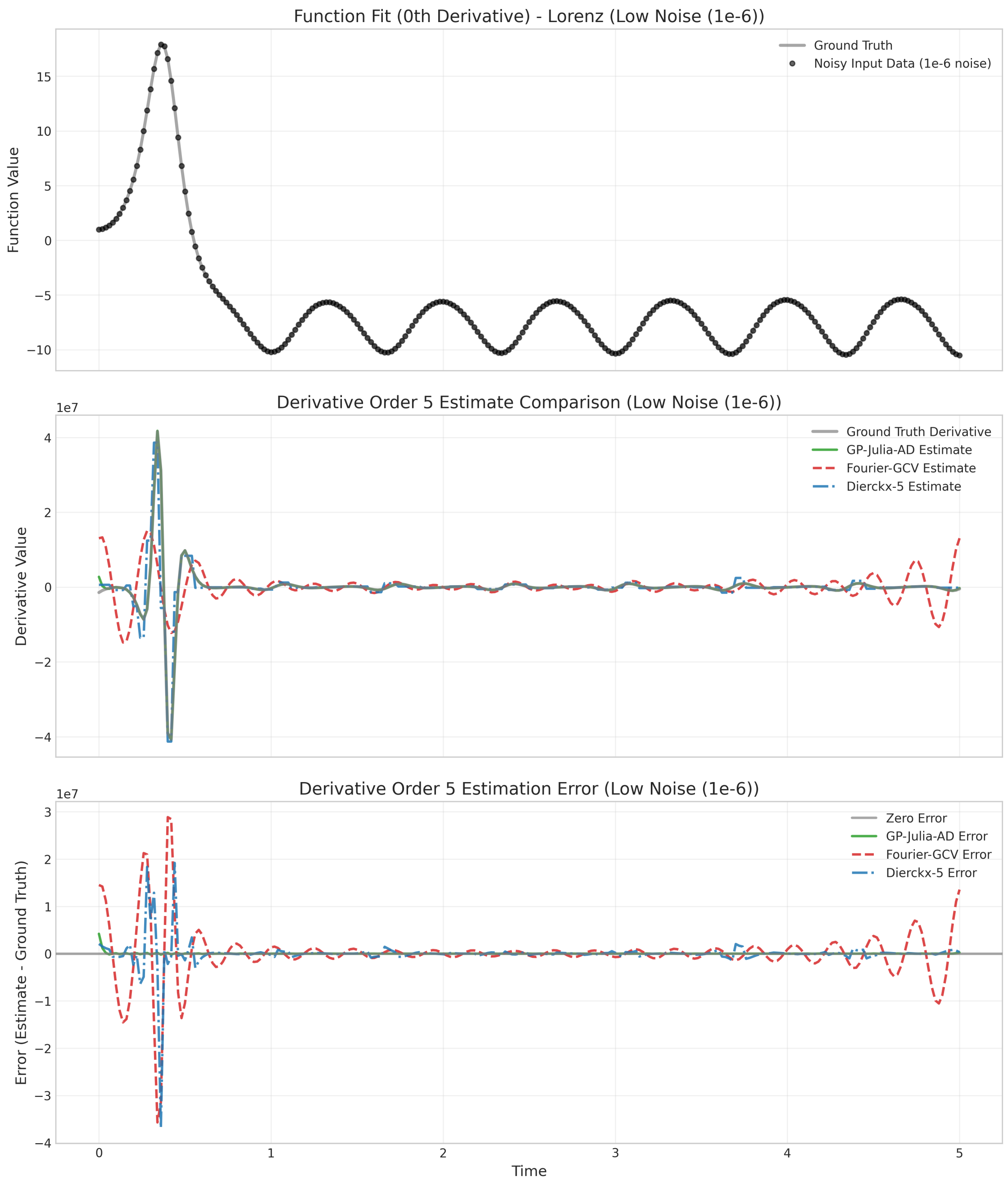
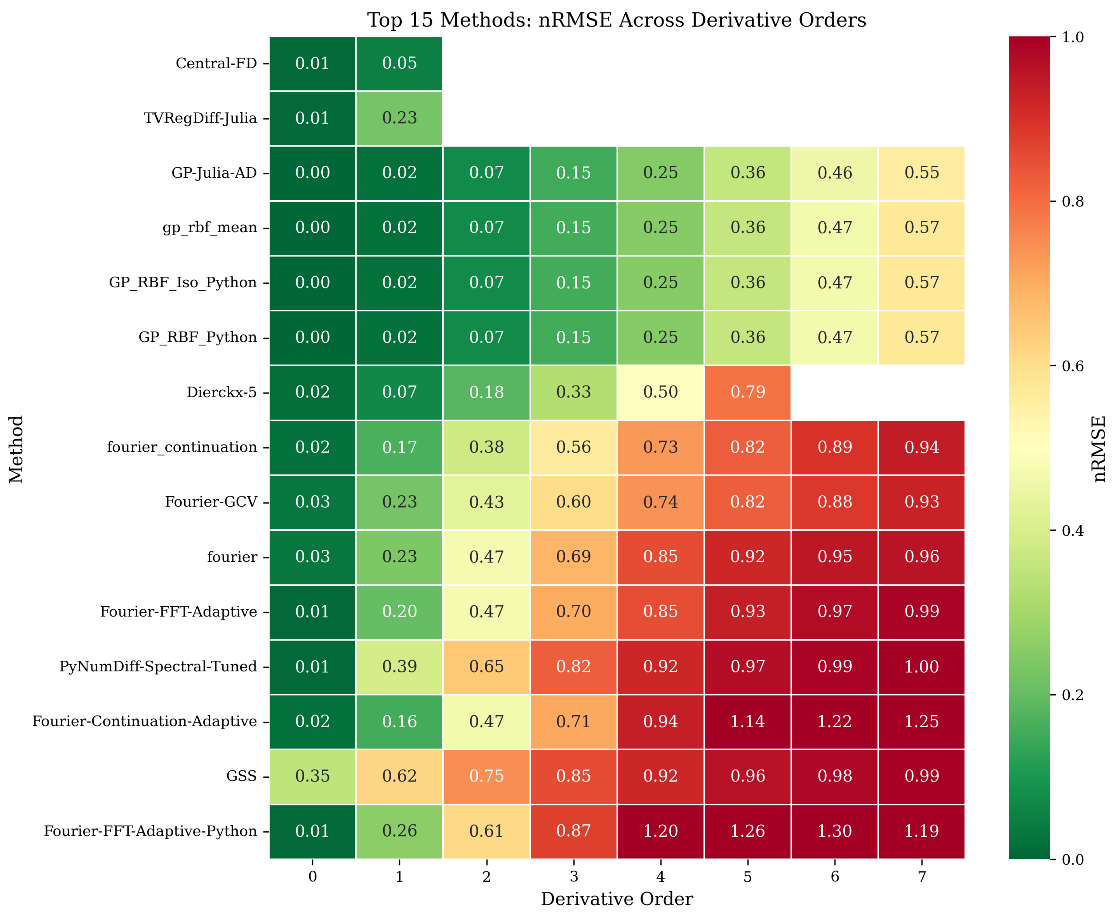


Figure 2: **Low-Noise Precision (1e-6 noise, 5th derivative).** Three-panel comparison in a low-noise regime: (top) noisy data points (black) are nearly imperceptible against gray ground truth; (middle) 5th-order derivative estimates from the top three methods; (bottom) estimation errors showing clear performance hierarchy. `GP-Julia-AD` dominates with RMS error of 272k, while `Dierckx-5` (3.1M) and `Fourier-GCV` (5.8M) show substantially larger errors. Even in this clean environment, high-order differentiation remains extremely challenging, with `GP-Julia-AD` achieving more than 10× better accuracy than competing methods.



Method	0	1	2	3	4	5	6	7
Central-FD	0.01	0.05						
TVRegDiff-Julia	0.01	0.23						
GP-Julia-AD	0.00	0.02	0.07	0.15	0.25	0.36	0.46	0.55
gp_rbf_mean	0.00	0.02	0.07	0.15	0.25	0.36	0.47	0.57
GP_RBF_Iso_Python	0.00	0.02	0.07	0.15	0.25	0.36	0.47	0.57
GP_RBF_Python	0.00	0.02	0.07	0.15	0.25	0.36	0.47	0.57
Dierckx-5	0.02	0.07	0.18	0.33	0.50	0.79		
fourier_continuation	0.02	0.17	0.38	0.56	0.73	0.82	0.89	0.94
Fourier-GCV	0.03	0.23	0.43	0.60	0.74	0.82	0.88	0.93
fourier	0.03	0.23	0.47	0.69	0.85	0.92	0.95	0.96
Fourier-FFT-Adaptive	0.01	0.20	0.47	0.70	0.85	0.93	0.97	0.99
PyNumDiff-Spectral-Tuned	0.01	0.39	0.65	0.82	0.92	0.97	0.99	1.00
Fourier-Continuation-Adaptive	0.02	0.16	0.47	0.71	0.94	1.14	1.22	1.25
GSS	0.35	0.62	0.75	0.85	0.92	0.96	0.98	0.99
Fourier-FFT-Adaptive-Python	0.01	0.26	0.61	0.87	1.20	1.26	1.30	1.19

Figure 3: **Performance Degradation with Increasing Derivative Order.** This heatmap shows the mean nRMSE for top methods at a 1% noise level. The vertical axis is sorted by average performance across all orders. This visualization clearly shows that `GP-Julia-AD` maintains low error across all derivative orders, while other methods like `Dierckx-5` and `GSS` are highly accurate for low orders but degrade significantly as the order increases.

details. While we could not be exhaustive, we sought to include representatives from all major methodological families as well as several novel approaches.

3.3 A Narrative of Method Selection

Our investigation proceeded in three main waves:

1. **Initial Foray: Adapting Noiseless Methods.** Our first attempts focused on adapting the successful AAA rational approximant method from our prior work. When these direct adaptations proved unstable on noisy data, we turned to a literature search for established techniques, leading us to methods based on local polynomial fitting (e.g., Savitzky-Golay, finite differences) and splines. While some of these showed moderate success, their effectiveness was often limited by the degree of the local polynomial, particularly for high-order derivatives.

2. **Second Wave: Denoising and Statistical Approaches.** Recognizing that explicitly handling noise was critical, we next investigated two new classes of methods. The first class involved explicit denoising steps, such as those based on wavelet filtering (MAD) or spectral filtering, applied before differentiation. The second, more sophisticated class involved methods with a statistical foundation that inherently model noise, such as Gaussian Process Regression, Total Variation Regularization, and Kalman filtering.

3. **Final Cohort.** The combination of these exploratory waves resulted in the final set of 31 methods that form the basis of this benchmark. These methods were selected based on their ability to provide full coverage across all derivative orders (0–7) and noise conditions tested. The cohort includes implementations in both Python and Julia, spanning multiple algorithmic families: Gaussian Process methods, spline interpolation, spectral approaches, filtering techniques, and rational approximation.

3.4 Implementation: The Challenge of High-Order Derivatives

A significant practical challenge was that almost no off-the-shelf package natively supports the computation of arbitrarily high-order derivatives. Overcoming this required substantial implementation effort.

- **Augmentation with Automatic Differentiation:** For methods whose underlying approximant was differentiable (e.g., Gaussian Processes), we augmented the implementation with Automatic Differentiation (AD) to compute the derivatives. Where possible, we used Taylor-mode AD, which is essential for the efficient computation of high-order derivatives, as naive nested AD has exponential complexity.

- **Analytic Derivatives:** For other methods, such as those based on splines or Fourier series, we derived and implemented the analytical expressions for their higher-order derivatives directly.

- **Noise Model Integration:** In some cases, we also had to implement custom noise-estimation steps to provide the algorithms with required hyperparameters, using techniques such as wavelet MAD or simple finite-difference-based noise estimation.

This substantial, and often non-trivial, implementation work was a prerequisite for creating the level playing field upon which this benchmark is built.

4 Experimental Design

To create a robust and comprehensive benchmark, we first formally define the estimation problem.

4.1 Formal Problem Statement

Given:
- A smooth but analytically unknown function $f : \mathbb{R} \to \mathbb{R}$.
- A set of n noisy observations $\{(t_i, y_i)\}_{i=1}^n$ where $y_i = f(t_i) + \epsilon_i$ on a uniform grid.
- Noise is assumed to be additive and Gaussian, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

Objective:
- Estimate the k-th order derivative, $\hat{f}^{(k)}(t)$, for derivative orders $k \in \{0, 1, \ldots, 7\}$.

Constraints:
- The evaluation is performed on the interior of the domain to avoid boundary effects that can unfairly penalize certain methods.
- A method is considered to have failed a configuration if it does not produce a finite result.

4.2 Testbed: Dynamical Systems

We selected three well-known Ordinary Differential Equation (ODE) systems to serve as the source of our ground-truth data. These systems were chosen to represent a diversity of dynamic behaviors:

1. **Lotka-Volterra:** A classic two-variable system exhibiting stable periodic oscillations.
2. **Van der Pol Oscillator:** A system with a non-linear damping term that produces a stable limit cycle.
3. **Lorenz System:** A three-variable system famous for its chaotic behavior, providing a more challenging test case.

For each system, trajectories were generated using a high-precision numerical integrator. The system's equations were then used to analytically compute the true derivatives up to the 7th order, providing a high-fidelity ground truth for our comparisons.

4.3 Noise Model

To simulate real-world data, we added Gaussian white noise to the integrated trajectories. The noise level was varied across a wide range, from `1e-8` (representing nearly clean data) up to `0.02` (representing a challenging 2% noise level relative to the signal's standard deviation). This wide range allows us to evaluate method performance in both high-precision, low-noise regimes and robustness-critical, high-noise regimes.

4.4 Evaluation Metrics

The primary metric for evaluation is the normalized Root-Mean-Square Error (nRMSE).

Rationale for Normalization: Direct comparison of raw RMSE values is not possible across different derivative orders because the magnitudes of the derivatives themselves vary dramatically. For instance, in the Lotka-Volterra system, the standard deviation of the true signal might be $\sim$0.2, while the standard deviation of its seventh derivative can be $\sim 10^5$.

To create a fair, order-comparable metric, we normalize the raw RMSE by the standard deviation of the true derivative signal. This yields a dimensionless error metric where an nRMSE of 0.1

consistently means a 10% error relative to the signal's characteristic magnitude, regardless of the derivative order.

All metrics are computed after excluding the endpoints of the time series, as many methods exhibit boundary effects that can unfairly skew the results.

4.5 Success Criteria

While the nRMSE provides a continuous measure of performance, it is also useful to define a threshold for what constitutes a "successful" or "acceptable" result. A method is considered successful for a given configuration if it achieves an nRMSE of less than 1.0, meaning its average error is smaller than the typical deviation of the true signal itself. An nRMSE greater than 1.0 indicates that the error is dominating the signal, and values greater than 10 are considered a catastrophic failure.

5 Detailed Analysis: From Raw Data to Final Rankings

The journey from raw experimental output to the final summary table involved a deliberate, multi-stage filtering and analysis process. This section details the methodology used to distill the results and arrive at our final conclusions.

5.1 Initial Data Filtering: Removing Unviable Methods

The first step was to filter out methods that were fundamentally unsuitable or unstable for the task. This was not based on performance, but on viability.

1. **Exclusion of Unstable Methods (AAA)**: Our implementations of the AAA (Adaptive Antoulas-Anderson) algorithm were found to be unstable for this specific task. Their errors were often orders of magnitude larger than any other method, and they frequently failed to produce valid output.

   It is important to frame this result: Our conclusion is not that rational approximants are unsuitable in general, but rather that a direct application does not appear to be robust to the combination of noise and differentiation. The AAA algorithm is exceptionally powerful for function approximation in noise-free contexts. We suspect that more sophisticated approaches, perhaps involving regularization or hybrid methods (e.g., fitting a rational approximant to a pre-smoothed signal), could unlock their potential. This remains a promising avenue for future research.

   Given these results, the AAA methods were removed from the main analysis cohort to prevent their extreme outliers from skewing aggregate statistics.

2. **Exclusion of Incomplete Coverage**: A primary goal of this study was to evaluate methods across a wide range of derivative orders (0 through 7). Several methods, particularly some of the Python legacy methods and those based on low-degree splines, did not have full coverage across all orders or failed consistently on certain systems. To ensure a fair, apples-to-apples comparison in our final rankings, only methods that successfully produced results for all tested configurations were included in the "contender" set.

5.2 Defining the "Contender" Set

After this initial filtering, we were left with a set of robust methods that provided full data coverage (up to order 5 for our main summary). This cohort became our "contender" set for the final analysis. All subsequent rankings and comparisons were performed *only within this group.*

5.3 Ranking Methodology

To produce the final summary table, we followed a two-step ranking process:

1. **Per-Cell Ranking**: For each individual experimental cell—defined by a unique combination of `(ODE_system, noise_level, derivative_order)`—we ranked the contender methods against each other based on their mean `nRMSE` (averaged across the 10 trials for that cell).
2. **Averaging Ranks**: We then calculated the final "Avg. Rank" for each method by averaging these per-cell ranks across two distinct regimes:
    - **Low-Noise Regime**: Averaged across noise levels below 1% (`1e-8`, `1e-6`, `1e-4`, and `1e-3`).
    - **High-Noise Regime**: Averaged across noise levels of 1% and above (`0.01` and `0.02`).

This methodology ensures that the final rank is a robust measure of a method's performance across a wide variety of conditions, and it prevents a single outlier or a particularly favorable test case from dominating the results.

5.4 Quantitative Performance Metrics

To add further quantitative rigor to our analysis, we defined two specific metrics to characterize method performance beyond the simple average error.

- **Noise Robustness:** We define a method's robustness for a given derivative order as the highest noise level at which its `nRMSE` remains at or below 0.1. A higher value on this metric indicates a method can maintain accuracy under more significant noise conditions. This provides a more concrete measure of robustness than average error alone.
- **High-Order Stability:** To quantify how gracefully a method's performance degrades as the task becomes more difficult, we measure its high-order stability. This is defined as the slope of the log of the `nRMSE` versus the derivative order, calculated in the low-noise regime. A lower, flatter slope indicates that the method is more stable and its error grows more slowly as it is tasked with computing higher-order derivatives.

These metrics provide a complementary view to the main rankings and are used to inform the specific recommendations and trade-offs discussed in our conclusion.

5.5 Analysis of Coverage Bias

A critical aspect of a fair benchmark is understanding "coverage bias." Many methods are not designed to compute high-order derivatives. For example, `Central-FD` and `TVRegDiff-Julia` in our study only support up to order 1. If we were to rank all methods naively based on their average error across all the tests they passed, these methods would appear artificially superior, as they would only be evaluated on "easy" low-order configurations and would be exempted from the challenging high-order tests where most methods struggle.

To create a fair comparison, we therefore restricted our main summary table to the cohort of "contender" methods that successfully produced results for all tested configurations up to order 5. Methods with partial coverage are not inherently worse, but they should be considered specialists. A practitioner who only needs a first-order derivative might find that `TVRegDiff-Julia` is an excellent choice, but it cannot be fairly compared in a general-purpose ranking against a method that successfully computes 7th-order derivatives.

5.6 Performance Degradation by Derivative Order

A clear pattern emerging from the data is the systematic degradation of performance as the derivative order increases. This is an expected consequence of the ill-posed nature of differentiation. We can characterize this trend in several phases:

- **Orders 0-2 (Low-Order):** In this regime, most contender methods perform well, and the performance differences between them are relatively modest, particularly in low-noise scenarios. The task of smoothing or finding a first or second derivative is not challenging enough to create significant separation between the top methods.
- **Orders 3-5 (Mid-Order):** This is the regime where a clear separation emerges. The task becomes significantly more challenging, and methods without sophisticated noise handling begin to struggle. The performance of GPR and the stronger spectral methods remains high, while simpler spline- and filter-based methods see a substantial drop in accuracy.
- **Orders 6-7 (High-Order):** This regime represents an extreme challenge. Only a very small subset of methods, primarily GPR, are able to produce a usable estimate, and even their errors are significant. For most other methods, the error in this regime constitutes a catastrophic failure. Derivative order is clearly the dominant factor in the difficulty of the estimation problem.

Figure 4 provides a comprehensive visual illustration of this systematic degradation. The figure shows performance across all eight derivative orders for the top seven methods, clearly demonstrating how error increases with order and how different methods respond to increasing noise levels at each order.

5.7 The Critical Role of Taylor-Mode AD for High-Order Derivatives

A key technical factor in the performance of high-order derivative estimation is the underlying mechanism of the Automatic Differentiation library used. Naively composing first-order AD operations (i.e., nested forward- or reverse-mode AD) to compute a high-order derivative results in an algorithm with exponential complexity, rendering it infeasible for orders beyond a handful.

The success of the `GP-Julia-AD` method, for instance, is critically dependent on its use of Taylor-mode AD. This mode is specifically designed to compute high-order derivatives efficiently by propagating a full Taylor series expansion through the computation, rather than just a single derivative value. This approach reduces the computational complexity significantly (often to polynomial time), making the computation of 5th, 6th, and 7th order derivatives tractable. This highlights that for practitioners seeking high-order derivatives, the choice of AD implementation is as important as the choice of the approximant itself.

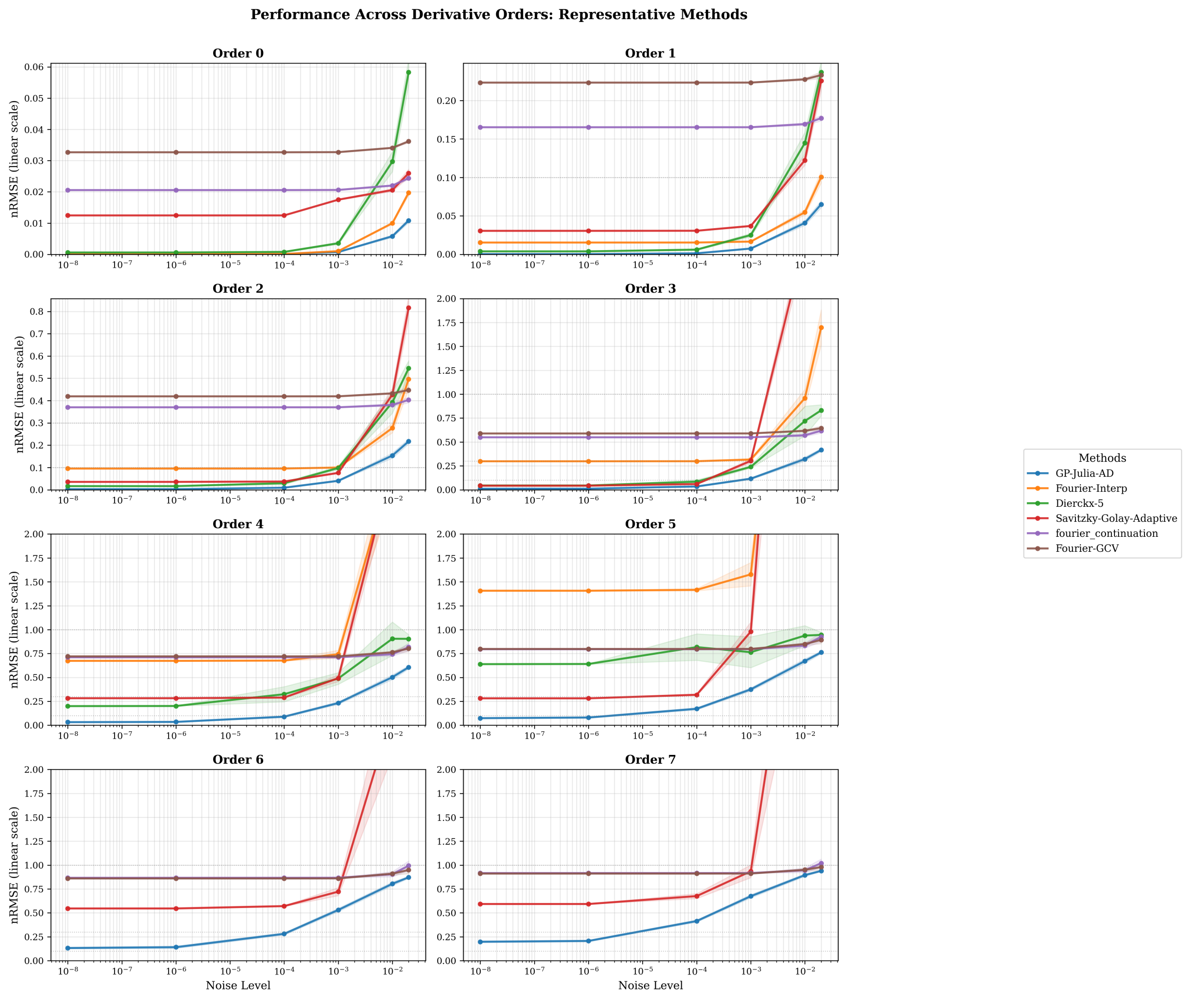


Figure 4: **Performance Across All Derivative Orders.** This 4×2 grid shows nRMSE vs noise level for the top 7 methods at each derivative order (0–7). Each panel illustrates the systematic degradation of performance as order increases. Note how GP-Julia-AD (top performer) maintains relatively stable performance across all orders, while other methods show dramatic degradation beyond order 3. The shaded regions indicate performance quality: green (excellent, nRMSE < 0.1), yellow (good, 0.1–0.3), and orange (acceptable, 0.3–1.0).

5.8 Computational Efficiency: Practical Considerations

Important Caveats: The computational timing measurements in this study should be interpreted with caution. Unlike the error metrics, which represent fundamental algorithmic characteristics, the timing data has several limitations that prevent it from being elevated to a primary finding:

- **Mixed-language implementations:** Our benchmark includes methods implemented in both Julia and Python, with different optimization levels and library maturity.
- **Small dataset size:** All timing measurements were performed on trajectories with $N = 101$ points. Scaling behavior for larger datasets ($N > 1000$) may differ significantly, particularly for methods with non-linear complexity.
- **Implementation quality variation:** Methods vary in optimization sophistication. Some are production-grade libraries, while others are research prototypes.
- **No warm-up or statistical rigor:** Timing measurements represent single runs without JIT warm-up for Julia methods or statistical validation across multiple hardware configurations.

Given these limitations, we present the following **qualitative guidance** based on our observations, rather than quantitative claims:

Three-Tier Framework for Method Selection:

1. **High-Speed Tier (sub-millisecond):** For real-time or large-scale applications where speed is paramount, filtering methods like `Savitzky-Golay` provide rapid computation with reasonable accuracy in moderate noise conditions. Basic spectral methods (simple FFT-based approaches) also fall in this category. These methods are particularly suitable when computational budget is the primary constraint.
2. **Balanced Tier (milliseconds to tens of milliseconds):** For most scientific applications requiring a strong balance between accuracy and speed, spectral methods represent an excellent middle ground. Methods like `Fourier-GCV`, `Fourier-Interp`, and high-degree splines (`Dierckx-5`) provide accuracy approaching that of GPR methods while maintaining practical computational efficiency. This tier is likely optimal for interactive data analysis and medium-scale studies.
3. **High-Accuracy Tier (hundreds of milliseconds):** When the highest possible accuracy is the primary concern and computational cost is secondary, Gaussian Process methods provide the lowest estimation errors across all noise levels and derivative orders. Our results show `GP-Julia-AD` consistently outperforms all other methods in accuracy. The computational investment is justified in applications where estimation quality is critical and the dataset size is moderate.

Scaling Considerations: For very large datasets ($N > 1000$), the cubic scaling of standard GPR implementations may become prohibitive, making fast spectral or filtering methods the only viable options unless sparse or approximate GP methods are employed.

6 Conclusion

This comprehensive study evaluated a wide array of numerical methods for the estimation of high-order derivatives from noisy data. After a detailed investigation that included a multi-stage filtering of methods and a deep dive into implementation details, our findings are clear and decisive.

6.1 Summary of Key Findings

- **Gaussian Process Regression (GPR) is the most robust and accurate method overall.** GPR methods consistently occupy the top ranks in both low- and high-noise regimes, making them the most reliable choice for general-purpose derivative estimation.
- **The optimal method depends on the use case.** While GPR is the best all-arounder, splines like `Dierckx-5` offer excellent precision for low-noise data, while spectral methods like `Fourier-Continuation` provide a compelling balance of speed and accuracy. For speed-critical applications, `Savitzky-Golay` is a robust and effective baseline.
- **Derivative order is the dominant difficulty factor.** Performance degrades systematically with increasing order across all methods. The problem becomes significantly more challenging beyond order 3, and only a handful of methods produce usable results at orders 6 or 7.
- **Implementation quality is a critical method characteristic.** Our study found significant performance differences between different software packages implementing the same underlying algorithm, highlighting that practitioners must consider the quality of a specific implementation, not just the theoretical method.

The primary recommendation of this work is that for practitioners who require accurate high-order derivatives from real-world, noisy signals, Gaussian Process Regression is the most reliable and effective starting point.

6.2 Future Work

This benchmark, while comprehensive, is not exhaustive. Several avenues for future research are immediately apparent:

1. **Testing on Diverse Signals:** Our study used ODEs that produce smooth, analytic signals. Future work should include testing on more challenging signals, such as those with discontinuities, sharp peaks, or chaotic behavior.
2. **Evaluating Alternative Noise Models:** The real world is not always Gaussian. A valuable extension would be to evaluate method performance under different noise models, such as multiplicative, Poisson, or heavy-tailed noise.
3. **Larger-Scale Problems:** Our study was limited to a modest number of data points. Investigating how method performance, particularly computational cost, scales to much larger datasets ($N > 1000$) would be of great practical interest.

6.3 Broader Implications: The Case for a Composable, Differentiable Ecosystem

Our findings also underscore a broader trend in scientific computing: the immense value of composable and differentiable software packages. The "Approximant-AD" framework is only possible when libraries for data modeling (e.g., Gaussian Processes) can seamlessly pass their results to libraries for automatic differentiation.

While not all numerical packages are readily differentiable out-of-the-box, our experience suggests that many can be adapted with modest effort. We encourage researchers and developers to prioritize differentiability in their own software and to contribute upstream to make foundational libraries in the ecosystem compatible with AD frameworks. Such efforts create a virtuous cycle, unlocking powerful new hybrid methodologies that benefit the entire scientific community, far beyond the immediate application of derivative estimation.

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]