Derivative-Enhanced Training for Data-efficient Surrogate Modeling

Paul Horvath[1,2], **Marian Staggl**[1,2], **Stefan Posch**[1,2]
[1] CD Laboratory for Physics-driven Machine Learning in Industrial Applications, Graz, Austria
[2] Institute of Thermodynamics and Sustainable Propulsion Systems,
Graz University of Technology, Austria
Corresponding author: Paul Horvath (paul.horvath@tugraz.at)

Abstract

Accurate surrogate modeling in engineering is often constrained by the high computational cost of generating training data from large scale numerical simulations. In many industrial applications, only a limited number of simulations can be afforded, which severely restricts the achievable surrogate accuracy, particularly in high dimensional parametric spaces. A promising approach to mitigate this curse of dimensionality is the incorporation of derivative information into surrogate training, which can be obtained efficiently via graph based implementations or adjoint calculations. This additional information captures local function structure, offering the potential to significantly improve data efficiency. In this work, we quantify the potential gains of derivative-enhanced training both theoretically and numerically, using a representative linear elasticity problem and an analytical benchmark. The findings provide guidance on the efficiency improvements achievable and the order of derivatives that yields the greatest benefit.

1 Introduction

The use of surrogate models is a state-of-the-art approach in modern engineering to avoid repeated evaluations of computationally expensive numerical simulations in applications such as optimization or uncertainty quantification [1]. However, the quantity of data required to train purely data-based surrogates, such as neural networks or Gaussian processes, can still be considerably high, leading to a large number of numerical simulations a priori. To reduce these data demands, several measures have been proposed to enhance the training of machine-learning-based surrogate models by injecting structure or auxiliary information. Among these, physics-informed neural networks (PINNs) [2] are particularly popular due to their relatively simple formulation and implementation. By using automatic differentiation (AD), PINNs introduce the residual of the governing differential equations as an additional loss term. In this way, PINNs can be used to identify unknown parameters of differential equations from data and to learn or even solve differentiable models by enforcing initial and boundary conditions through soft or hard constraints. PINN-based surrogates have been successfully applied to Bayesian inverse problems [3] and design optimization [4]. Complementary to physics-based regularization, another approach to improve data efficiency is to exploit derivative information available from simulations or adjoint solvers. Gradient-enhanced training, often referred to as Sobolev training [5], augments standard data fitting by incorporating sensitivity information, which can sharpen surrogate accuracy and generalization in low-data regimes [6]. In case of Gaussian processes (GP) for surrogate modeling, since differentiation is a linear operator, the derivative of a GP is another GP [7]. Thus, the use of derivative information to train GPs can have a significant influence on the data quantity requirements. Semler and Weiser [8] demonstrated the benefits of gradient-enhanced Gaussian process surrogates for inverse problems compared with GPs that rely solely on function values. They employed an adaptive selection of evaluation locations and tolerances

The Third Austrian Symposium on AI, Robotics, and Vision (AIROV26).

using a greedy heuristic, and assessed performance on both an analytical test case and a numerical example based on the finite element method. Their parameter reconstruction results consistently indicated that incorporating gradient information improves the quality of the inferred parameters, emphasizing the value of derivative data in surrogate-based inversion. Bouhlel and Martins [9] motivate gradient-enhanced Kriging with the availability of efficient sensitivities, e.g. from adjoint methods, but their engineering demonstrations rely on benchmark engineering functions rather than on gradients generated by a high-fidelity adjoint solver. Further prior work has shown that gradient information from high-fidelity solvers can substantially improve surrogate quality, both for neural surrogates and Gaussian-process-based models. In particular, gradient histories from adjoint-based optimization have been used to train multi-fidelity neural surrogates [10], while adjoint-based sensitivities have also been incorporated into gradient-enhanced deep GP aerospace-related applications [11]. More recent works extend this idea also to operator learning [12].

There are relevant works that use automatically differentiable simulators or differentiable physics for optimization and for training neural components [13], but explicit studies that use AD-generated gradients from PDE solvers as supervised signals for surrogate training remain comparatively scarce. This appears to be especially true for Gaussian-process-based surrogates, where gradient-enhanced formulations are well established, but the gradients are often not explicitly attributed to AD. Building on this perspective, we investigate gradient-enhanced strategies that combine differentiable physics solvers and AD to supply derivative information systematically. Two representative applications, namely the Rosenbrock function and a structural mechanics case, are considered to show how gradient augmentation affects sample-efficiency scaling in theory. Our results indicate that incorporating first-order derivatives provides a substantial and robust performance gain, whereas the value of higher-order derivatives is more problem-specific and depends on factors such as model smoothness.

1.1 Motivation

In many engineering applications, generating training data via numerical simulations represents the primary computational bottleneck, often taking significantly longer than the subsequent training of surrogate models. Standard data driven approaches require sufficiently large datasets to accurately capture complex physical behaviors, severely limiting their scalability. Consequently, our primary objective in applying derivative-enhanced modeling is to fundamentally reduce this data generation effort. By extracting more information from each individual simulation run, we aim to maximise data efficiency by reducing the high computational cost of acquiring training sets.

2 Methodology

To systematically evaluate this data efficiency, we introduce a theoretical scaling framework designed to weigh the information gain of higher order derivatives against the increased cost of their generation. Derivative-enhanced modeling enriches the surrogate by incorporating derivatives of the simulation output with respect to the input parameters. Because the underlying simulation model is formulated in a differentiable manner, these derivatives can be obtained efficiently via AD at a computational cost comparable to a single additional forward run. The following sections detail how this derivative information is incorporated into the respective training pipelines of neural network and Gaussian Process regression surrogates.

2.1 Sobolev training of neural networks

When learning a function u using classic neural network training, we typically aim to minimize a loss using function values $u(x_i)$ for training points x_i. Having access to first- or higher order derivative information, we extend this idea to a Sobolev space formulation by training not only with function values, but also matching first- or higher order derivatives of the target function [5]. Given a surrogate model $u_\theta(\mathbf{x})$ and reference data $u(\mathbf{x})$, the general Sobolev loss can be written as

$$\mathcal{L}_{\text{Sob}} = \sum_{k=0}^{r} \lambda_k \left\| \nabla^k (u_\theta(\mathbf{x}) - u(\mathbf{x})) \right\|_{L_p}, \tag{1}$$

where λ_k controls the relative weighting of the derivative information. Incorporating derivative terms constrains the local shape of the learned function and may reduce the number of required training samples, or increase model accuracy, as explained in section 2.3.

2.2 Gradient-enhanced GP regression

GP models naturally incorporate derivative observations by extending their covariance structure [7]. Because differentiation is a linear operator, the joint prior's covariance can be expanded to include not only function values but also their full gradient vectors by simply differentiating the base kernel $k(\mathbf{x}, \mathbf{x}')$. For an input space of dimension d, the complete extended joint covariance matrix combining function values and gradients can be expressed as [9]:

$$K_{\text{GE}}(\mathbf{x}, \mathbf{x}') = \begin{bmatrix} k(\mathbf{x}, \mathbf{x}') & (\nabla_{\mathbf{x}'} k(\mathbf{x}, \mathbf{x}'))^\top \\ \nabla_{\mathbf{x}} k(\mathbf{x}, \mathbf{x}') & \nabla_{\mathbf{x}} \nabla_{\mathbf{x}'}^\top k(\mathbf{x}, \mathbf{x}') \end{bmatrix}$$

Here, the off diagonal blocks represent the cross covariance column and row vectors between a function value and its gradients. The bottom right block captures the $d \times d$ covariance matrix between two gradient vectors via the mixed second derivatives. By augmenting the training data with these gradient observations, the GP posterior becomes more informative, leading to improved predictive accuracy and reduced uncertainty under limited data conditions.

2.3 Theoretical efficiency scaling

Evaluating the data efficiency of derivative-enhanced models requires weighing the information gained per calculation against its computational cost. This section theoretically assesses the potential for reducing the computational effort required to generate training data by establishing an equivalent information assumption. This approach suggests that each piece of extracted data, whether a function value or a partial derivative of an arbitrary degree, provides equally valuable information. Such an assumption holds perfectly, for instance, when fitting an n-coefficient polynomial surrogate to a multivariate polynomial system, which simply requires n pieces of information regardless of their source. Under this premise, the total information $I(d, r)$ extracted from one calculation is determined by the problem's dimensionality d and the maximum derivative degree r:

$$I(d, r) = \binom{d + r}{r} = \frac{(d + r)!}{d!r!}$$

A standard forward calculation ($r = 0$) yields exactly one piece of information. Including the gradient ($r = 1$) yields $1 + d$ pieces, and so forth. For a fair comparison, the escalating computational cost of higher-order derivatives must be accounted for. Assuming a single forward calculation has a normalized computational cost of $C = 1$ (requiring time Δt), evaluating the full gradient via AD also takes $\approx \Delta t$, resulting in a combined cost of $C = 2$. Computing the Hessian and higher-order derivatives requires differentiating the gradient entries, leading to cost scaling $\propto d^{r-1}$. The normalized computational cost $C(d, r)$ relative to a standard forward calculation is thus:

$$C(d, r) = 1 + \binom{d + r - 1}{r - 1} = 1 + \frac{r}{d + r} \frac{(d + r)!}{d!r!}$$

To estimate the computational savings of incorporating derivatives, we define $R(d, r)$ as the ratio of computational cost to the amount of information gained per sample:

$$R(d, r) = \frac{C(d, r)}{I(d, r)} = \frac{d!r!}{(d + r)!} + \frac{r}{d + r}$$

Effectively, $R(d, r)$ quantifies the data generation cost efficiency of using a derivative-enhanced method compared to relying solely on function values. Notably, for both gradients ($r = 1$) and Hessians ($r = 2$), this ratio is identical ($R = \frac{2}{d+1}$). However, for higher-order derivatives, the cost advantage deteriorates. Thus, only first and second derivatives appear theoretically beneficial for reducing simulation calls, though the vulnerability to numerical errors often reduces the practicality of evaluating the Hessian. As illustrated in Figure 1, the relative generation cost $R(d, r)$ reaches its minimum for first and second derivatives but worsens for higher orders. Furthermore, the efficiency advantage of the derivative-enhanced approach grows significantly as the dimensionality of the parameter space d increases.

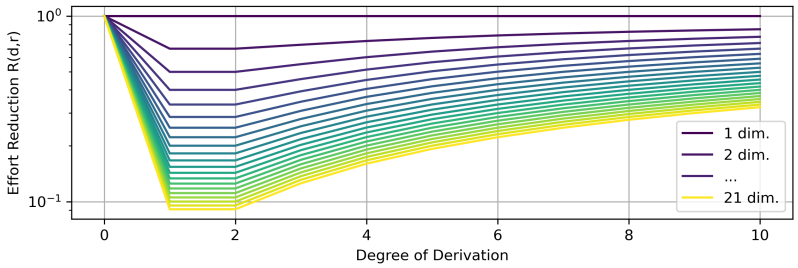

Figure 1: Reduction of computational effort following the assumption of equivalent information content with respect to different dimensions and derivative degrees.

3 Numerical studies

In this section, we assess how incorporating derivative information into surrogate model training impacts overall data demand. We evaluate the proposed approach using both a representative engineering application and an analytical benchmark. We investigate the effects of Sobolev training and gradient-enhanced Gaussian Process Regression (GE-GPR) on a structural example utilizing a differentiable finite element (FE) solver. This demonstrates the practical utility of derivative-enhanced learning within simulation environments, highlighting its potential to scale to industrial applications where finite element methods are already well established. Additionally, we use the Rosenbrock function as a benchmark to analyze how these methods perform on functions characterized by strong nonlinear characteristics. In order to evaluate how well the different models scale for higher dimensional parametric spaces, every model is trained for an increasing number of datasets while the accuracy of the trained models is evaluated (see Figure 2). The accuracy of each model is measured in terms of a R2 score, defined as

$$score = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{2}$$

where $\hat{y}_i$ denotes the predicted values, y_i the reference solution, and $\bar{y}$ the mean of the reference data. A value of $score = 1$ corresponds to a perfect prediction. Increasing the number of training

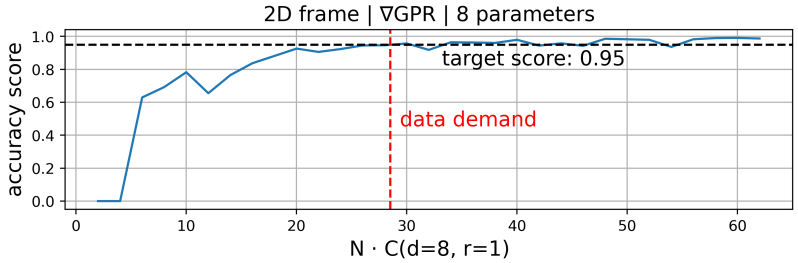

Figure 2: Computational effort of data generation for the eight parameter FE case using the gradient-enhanced GPR. The plotted computational cost represents the number of training samples multiplied by the evaluation cost per sample.

samples N generally improves model accuracy, however, it also increases the computational cost of data generation. We define total computational effort as the number of samples N multiplied by the relative cost per evaluation. Using a standard forward simulation as the baseline cost of 1, the total effort for standard data generation is simply N. Because extracting first order derivatives via AD requires roughly one additional pass through the computational graph, the effort per sample doubles, resulting in a total computational effort of $2N$. The figures below therefore present the obtained accuracy as a function of this total computational effort. From each study, the amount of data required

to reach a specified target accuracy is extracted, enabling the underlying data generation effort to be analyzed as a function of the parameter space dimensionality (see Figures 5 and 6).

3.1 Experimental Setup

Data Generation and Experimental Design. Datasets containing analytical derivatives are generated offline for both a structural FE problem (detailed in Section 3.2) and an analytical Rosenbrock benchmark. For the structural case, we use torchFEM, a differentiable FE library, utilizing mesh morphing to map a parameterized base mesh to target geometries. This system is defined by up to eight parameters: two representing the applied loads and six defining the geometry. For the analytical benchmark, a standard PyTorch implementation of the Rosenbrock function is utilized to evaluate scaling across varying dimensionalities. In both scenarios, input configurations are generated via uniform random sampling within predefined bounds. Following the forward evaluation of the target function, exact gradients and Hessians are extracted via AD. The target function is defined as the displacement magnitude at the load position for the FE model, and as the scalar output for the Rosenbrock case. As a preprocessing step, all input parameters and target function values are normalized via min-max scaling, and the corresponding gradients and Hessians are scaled accordingly. From a total generated dataset of 600 offline samples, 20% (120 samples) are reserved as a fixed test set to evaluate final model accuracy score and 10% (60 samples) are used for validation. Training subsets, ranging from 1 to 300 samples, are then randomly drawn from the remaining pool of 420 samples to systematically evaluate data demand and scaling behavior. Because the validation set is used exclusively for function value early stopping (requiring no gradients), it represents a constant across all NN models. Therefore, visualizations in the subsequent sections focus strictly on the computational effort required to generate the active training data.

Mesh Morphing. In order to deform the mesh of the parameterized FE example, a control-mesh is defined and wrapped over the computational mesh. This control mesh is discretized into triangular elements and uses the minimal number of nodes required to accurately define the geometric contours. During the setup of the parameterized model, each node of the FE mesh is mapped to a specific control element. When the control nodes are displaced, the internal FE nodes are updated via a piecewise linear transformation (see Fig. 3). Following this spatial deformation, the shape function derivatives and Jacobian determinants of the FE model are recomputed to reflect the new geometry.

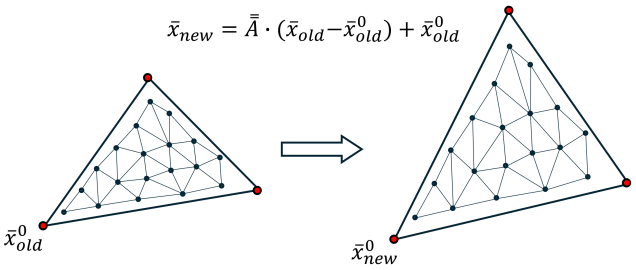

Figure 3: Mesh deformation methodology

GP Surrogate. The Gaussian Process regression models are implemented using GPyTorch. The model utilizes a constant mean and a Radial Basis Function (RBF) base kernel with varying length scales per dimension together with a scale kernel. The length scales are constrained to the interval [0.01, 1000]. A Gaussian likelihood is used, with the observation noise fixed at $1e-7$ and excluded from gradient updates to reflect the deterministic nature of the finite element solver. Model hyperparameters are optimized by maximizing the Exact Marginal Log-Likelihood using the Adam optimizer with a learning rate of 0.1 for 200 iterations.

NN Surrogate. The neural network surrogate is a compact Multilayer Perceptron (MLP) mapping input parameters to the target displacement. The architecture consists of two hidden layers with Tanh activation functions, utilizing four neurons per layer for the FE example and 16 for the Rosenbrock function. Training is executed with a learning rate of 0.005 and an L2 weight decay of 0.001. The Sobolev training loss function applies equal weighting (1.0) to the mean squared errors of the function values, gradients, and Hessians. The maximum number of training epochs is set to 100000, with an early stopping criterion triggered either upon reaching a validation score of 0.997 or after 10000 consecutive epochs without improvement.

3.2 FE example

As a structural reference model, a parameterized two dimensional frame structure is considered, as illustrated in Figure 4. The frame is fixed at its base, and an external force is applied at the upper right node. This system is governed by an eight dimensional parameter space: six geometric parameters control the individual bar thicknesses as well as the overall height and width of the frame, while two load parameters define the horizontal and vertical magnitudes of the applied force. The parameter bounds used for data generation can be found in Table 1. As the target function, the displacement magnitude at the load position is extracted. To evaluate surrogate performance, standard function only approaches are compared against the derivative-enhanced GE-GPR and Sobolev models across these respective training pipelines (see Sections 2.1 and 2.2).

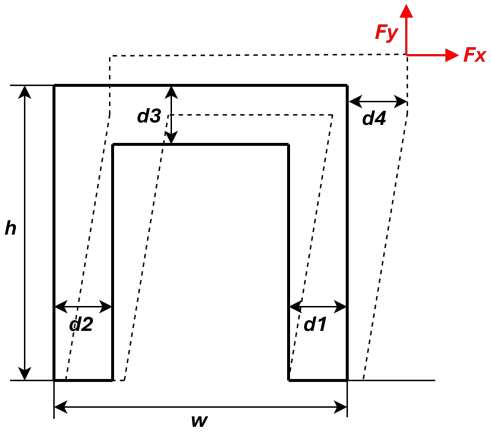

Figure 4: Illustration of the frame model with six geometric and two load parameters. The Young's modulus is $E = 1000$ MPa.

Table 1: Parameter bounds

Parameter	Min	Max	Unit
w	5.0	7.0	[m]
h	5.0	7.0	[m]
d1	0.5	1.5	[m]
d2	0.5	1.5	[m]
d3	0.5	1.5	[m]
d4	-0.5	0.5	[m]
Fx	0.1	1.0	[N]
Fy	0.1	1.0	[N]

Figure 5 presents the results for the GPR (left) and neural network (right) surrogates at target scores of 0.9 and 0.99. The blue and orange curves correspond to the classical GPR and the GE-GPR models, respectively. Similarly, the green and red curves represent the neural network surrogates trained with the standard loss formulation and the first order Sobolev loss. As expected, data demand increases exponentially with the number of parameters, and to some extent, this growth accelerates when higher target accuracies are required. The derivative-enhanced surrogates outperform their standard counterparts, requiring less computational effort in data generation to achieve the target scores. Moreover, the data demand based on theoretical efficiency scaling, introduced in Section 2.3, is plotted for both models (black dotted line). This line represents the theoretical reduction in computational effort based on the equivalent information assumption. The observed data demand for the neural network trained with first order Sobolev loss follows this theoretical prediction, demonstrating an approximate $2/(1+d)$ reduction in computational effort compared to the classical NN model.

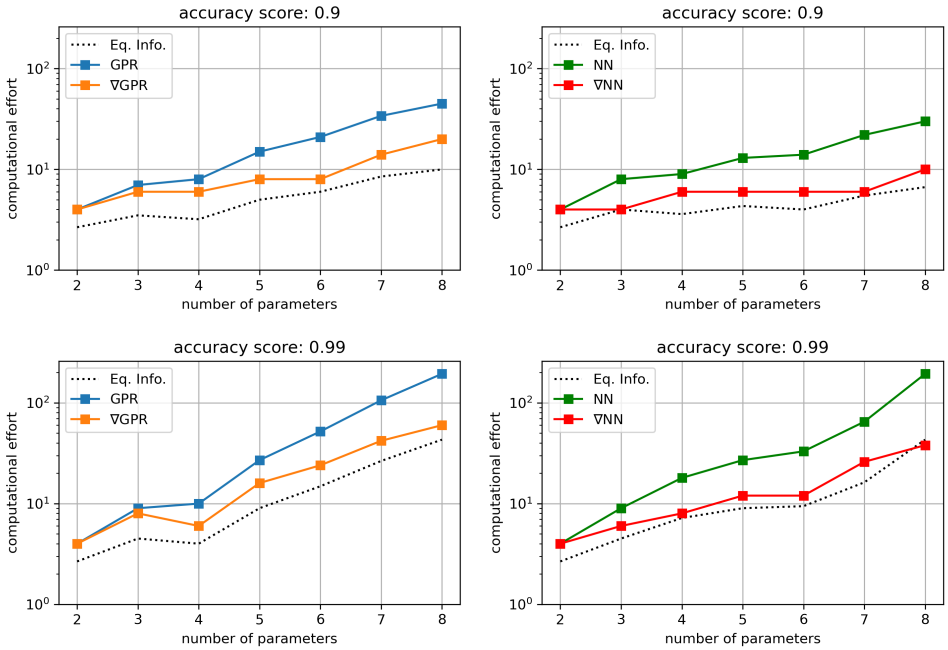

Figure 5: Computational-effort-corrected performance of surrogate models on the parameterized FE example for target accuracies of 0.9 and 0.99 (left: GPR, right: neural networks). Incorporating first order derivatives successfully reduces data generation effort. Furthermore, the neural network trained with first order sobolev loss exhibits similar scaling behavior to the theoretical baseline of the theoretical equivalent information assumption (black dotted line).

3.3 Rosenbrock function

As a second example, we consider the n-dimensional Rosenbrock function evaluated over the domain $[-1, 1]^n$. Although the computational effort required for data generation is negligible in this case, the problem provides a convenient benchmark for analyzing the proposed methods on a smooth yet highly nonlinear function. Evaluating the same surrogate models for target scores of 0.9 and 0.99 (Figure 6), we observe consistent trends: the first order derivative-enhanced surrogates (see red and orange curves) outperform their standard counterparts, lowering the required data generation effort. Furthermore, much like the neural network in the FE example, the GE-GPR closely follows the theoretical efficiency scaling (black dotted line) at the higher target accuracy. Besides first order derivatives, we also investigated the influence of incorporating Hessian information via a second order Sobolev loss for training the neural network. While the Hessian-informed model (purple curve) required more computational effort than the standard model at lower target accuracies and parameter dimensions, its data efficiency improved in higher dimensional spaces and at increased accuracy thresholds. However, it did not surpass the overall data efficiency of the first order Sobolev model. Exploring even higher dimensional parameter spaces may eventually reveal a comparative advantage for this second order approach.

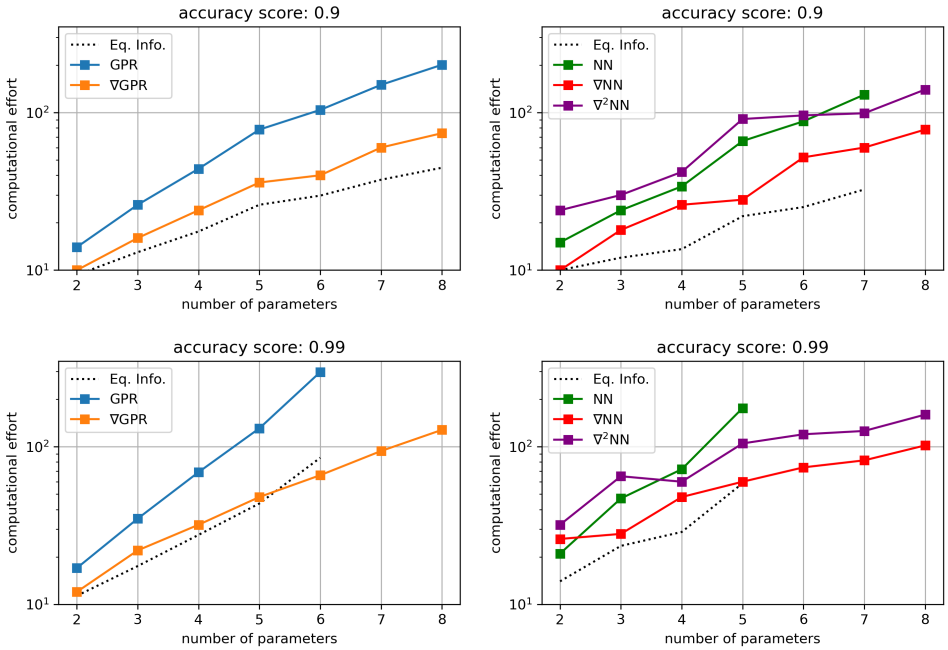

Figure 6: Computational-effort-corrected performance on the Rosenbrock function for target accuracies of 0.9 and 0.99 (left: GPR, right: neural networks). Consistent with the FE example, first order derivative-enhanced models outperform standard counterparts at higher dimensions. Here, the GE-GPR model follows the overall trend of the theoretical efficiency scaling (black dotted line). The Hessian-informed neural network demonstrates improved efficiency at higher accuracies but does not surpass the first order model.

4 Discussion and conclusion

This study investigates the benefits of enhancing surrogate models with derivative information obtained from differentiable numerical simulations to reduce data demand. By formulating simulation models in a differentiable manner, derivatives of the simulation output with respect to input parameters can be obtained efficiently via AD. A critical challenge, however, is determining the optimal derivative order to incorporate and balancing the reduced data demand against the computational cost of generating this data. To systematically evaluate these trade offs, a theoretical model is introduced in Section 2.3 and two benchmark problems are investigated: an FE-based structural example in Section 3.2 and the Rosenbrock function in Section 3.3. To evaluate surrogate performance, we measure the computational effort of generating data required to reach a specified target score for problems with up to eight parameters. The incorporation of higher order derivative information into surrogate training is, however, not straightforward. For example, the computational cost of fitting a GPR model scales with approximately $\mathcal{O}(N^3)$, where N denotes the total number of observations. For a first order GE-GPR model, this includes all function evaluations and gradients, causing the data vector to scale linearly with the parameter dimension. Extending this to second order methods incorporates the Hessian matrix, causing the observation count per sample to scale quadratically with the number of parameters, severely bottlenecking the training process. While Sobolev training of neural networks scales more efficiently with higher order derivatives, balancing the corresponding loss terms remains a significant challenge. In the Rosenbrock example, we observed that incorporating Hessian information yielded a benefit over the standard model at high target accuracies, however, it did not surpass the overall efficiency of the first order models. The successful application of higher order Sobolev training to structural examples remains an open subject for future investigation. Overall, our numerical studies show that using first order derivatives in surrogate training can reduce the computational effort of generating data compared to the standard training methods. The results therefore highlight the potential of such approaches to scale well to industrial sized problems, particularly for applications constrained by the high cost of simulation data.

Acknowledgments and Disclosure of Funding

The financial support by the Austrian Federal Ministry of Economy, Energy and Tourism, and the Christian Doppler Research Association is gratefully acknowledged.

References

[1] G.I. Schuëller. On the treatment of uncertainties in structural mechanics and analysis. *Computers & Structures*, 85(5):235–243, 2007. Computational Stochastic Mechanics.

[2] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[3] Yongchao Li, Yanyan Wang, and Liang Yan. Surrogate modeling for bayesian inverse problems based on physics-informed neural networks. *Journal of Computational Physics*, 475:111841, 2023.

[4] Yubiao Sun, Ushnish Sengupta, and Matthew Juniper. Physics-informed deep learning for simultaneous surrogate modeling and pde-constrained optimization of an airfoil geometry. *Computer Methods in Applied Mechanics and Engineering*, 411:116042, 2023.

[5] Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *Advances in neural information processing systems*, 30, 2017.

[6] Rhys Newbury, Jack Collins, Kerry He, Jiahe Pan, Ingmar Posner, David Howard, and Akansel Cosgun. A review of differentiable simulators. *IEEE Access*, 12:97581–97604, 2024.

[7] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[8] Phillip Semler and Martin Weiser. Adaptive gradient-enhanced gaussian process surrogates for inverse problems. *Mathematical Optimization for Machine Learning: Proceedings of the MATH+ Thematic Einstein Semester 2023*, page 59, 2025.

[9] Mohamed A Bouhlel and Joaquim RRA Martins. Gradient-enhanced kriging for high-dimensional problems. *Engineering with Computers*, 35(1):157–173, 2019.

[10] Tao Zhang, Mark Woodgate, George Barakos, and Yu Luo. A multi-start aerodynamic shape optimisation approach via multi-fidelity neural networks. *Aerospace Science and Technology*, 168:111006, 2026.

[11] Viv Bone, Chris van der Heide, Kieran Mackle, Ingo Jahn, Peter M. Dower, and Chris Manzie. Gradient-enhanced deep gaussian processes for multifidelity modeling. *Journal of Computational Physics*, 520:113474, 2025.

[12] Chanik Kang, Joonhyuk Seo, Ikbeom Jang, and Haejun Chung. Adjoint method-based fourier neural operator surrogate solver for wavefront shaping in tunable metasurfaces. *iScience*, 28(1):111545, 2025.

[13] Philipp Andelfinger. Differentiable agent-based simulation for gradient-guided simulation-based optimization. In *Proceedings of the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 27–38, 2021.