Reducing training costs with derivative informed data from hypercomplex automatic differentiation

Krishna Logakannan, Zhitong Xu, Shandian Zhe, Mike Kirby, Jacob Hochhalter, University of Utah
Mauricio Aristizabal, Harry Millwater, University of Texas at San Antonio
Geoffrey Bomarito, NASA Langley Research Center

https://ceid.utsa.edu/HYPAD/

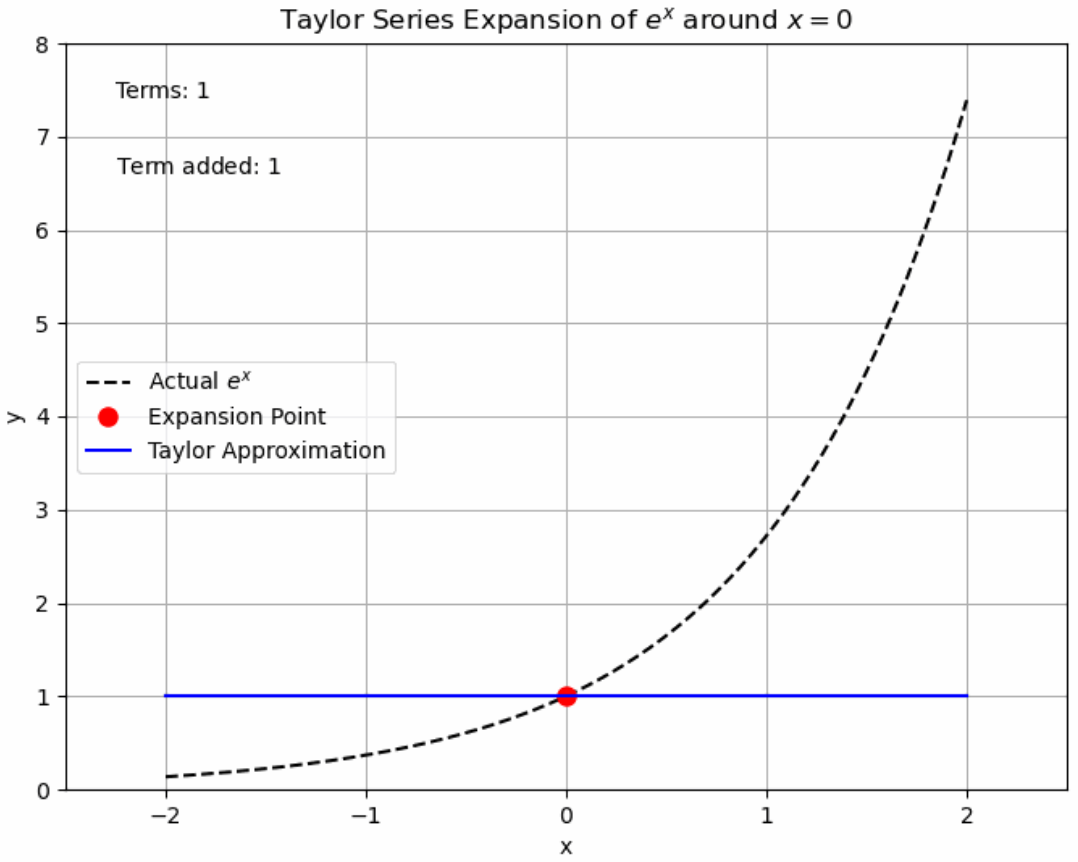
https://github.com/nasa/bingo

Outline

- Significance of derivatives in training data
- Efficient calculation of derivatives
- Results
  - Numerical experiments
  - Mechanics application
  - Computational efficiency

Motivation for derivatives in SciML

- Scientific and engineering applications are often characterized by **small** and **expensive data**
- Physics guides help in small data scenarios and aid explainability
- Derivatives of labels can often be obtained and provide similar benefits as physics guides
- Time to acquire **derivatives of training data** and computing **derivatives of models** is prohibitive



Derivative Informed Training Data (DITD)

- A typical training set includes $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $\boldsymbol{x}^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}\}$ is input feature vector, $y^{(i)}$ is the output label, and N is the number of training points.

- The objective of the Machine Learning (ML) model, $m(\boldsymbol{x}^{1:N}; \theta)$, is to find a function form with parameters (θ), such that the loss ($\mathcal{L}$) is minimized

Traditional loss

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left\| y^{(i)} - m(\boldsymbol{x}^{(i)}; \theta) \right\|_2^2$$

DITD extension

$$+ \sum_{k=1}^{d} \frac{1}{N}\sum_{i=1}^{N} \left\| \nabla^k y^{(i)} - \nabla^k (m(\boldsymbol{x}^{(i)}; \theta) \right\|_2^2$$

where *d* is the desired order of derivative

- The cost of obtaining derivatives of training data up to arbitrary order is expensive.
- The cost of calculating derivatives during training is expensive.
- We use a hypercomplex algebra method called Order Truncated Imaginary (OTI) numbers

Hypercomplex algebra

HYPERCOMPLEX AUTOMATIC DIFFERENTIATION (HYPAD)

Finite Differentiation Method (FD)

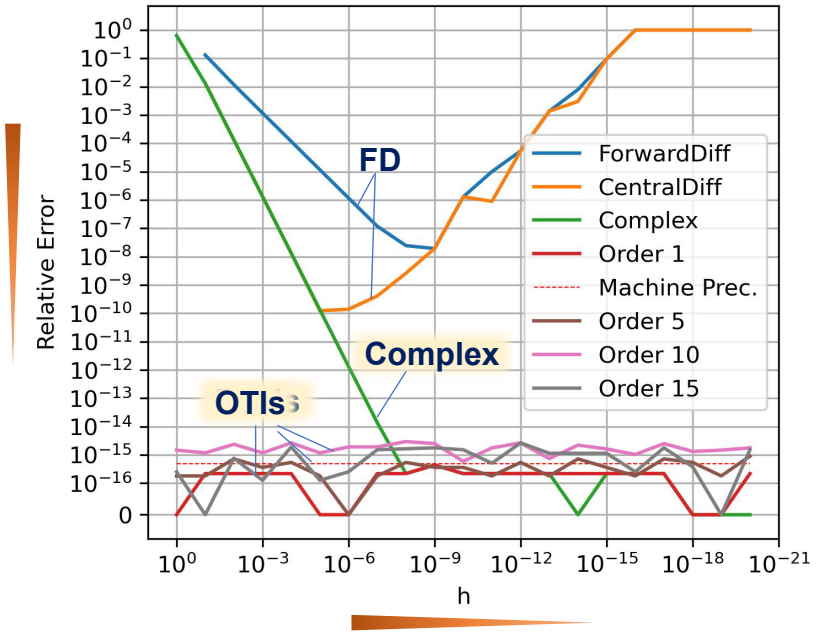
$$\frac{df(x_o)}{dx} \approx \frac{f(x_o + h) - f(x_o)}{h}$$

Complex Taylor Series Expansion (CTSE)

$$\frac{df(x_o)}{dx} \approx \frac{Im\left(f(x_o + \boldsymbol{i}h)\right)}{h}$$

HYPAD

$$\frac{df(x_o)}{dx} = \frac{Im\left(f(x_o + \boldsymbol{\epsilon}h)\right)}{h}$$



HYPAD features:

- High accuracy
- Insensitive to perturbation step-size
- Can be integrated with finite element models and other high-fidelity models.
- OTI is an efficient form of HYPAD
- *All* higher-order sensitivities can be computed in a single analysis with **OTI numbers** [1].
- Open-source library **otilib**: https://github.com/mauriaristi/otilib

OTI with Symbolic Regression (SR)

- In SR, equations are represented and evaluated using a directed acyclic graph (DAG), for example, evaluation of $f(x_o) = 2x_o^2$ is performed as shown in the table

i	Node	Param. 1	Param. 2	Conventional expression	OTI expression
0	Constant	0	-	$w_0 = 2$	$w_0^* = 2$
1	Variable	0	-	$w_1 = x_0$	$w_1^* = x_0 + \epsilon_1$
2	Multiply	0	1	$w_2 = w_0 \times w_1$	$w_2^* = w_{0_r} w_{1_r} + w_{0_r} w_{1_{\epsilon_1}} \epsilon_1$
3	Multiply	1	2	$w_3 = w_2 \times w_1$	$w_3^* = w_{1_r} w_{2_r} + \left(w_{1_r} w_{2_{\epsilon_1}} + w_{1_{\epsilon_1}} w_{2_r} \right) \epsilon_1 + 2 w_{1_{\epsilon_1}} w_{2_{\epsilon_1}} \epsilon_1^2$

Numerical Experiments

- Multi-dimensional optimization function (Griewank Function)

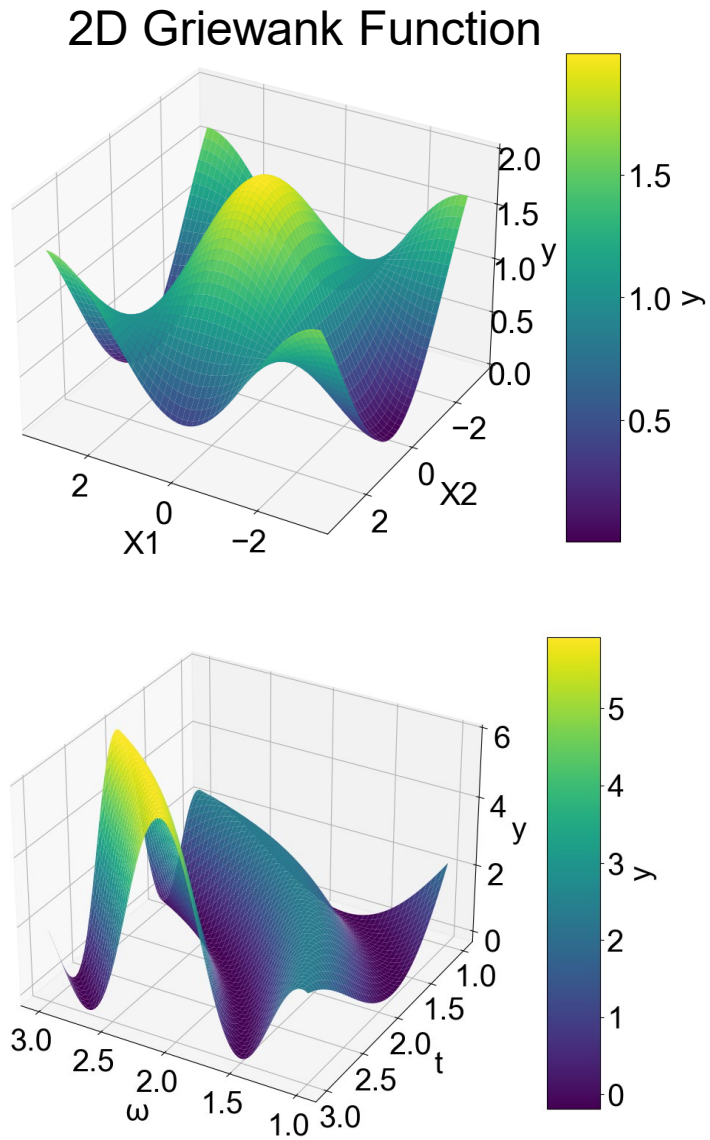
$$y = \sum_{i=1}^{dim} \frac{x_i^2}{4000} - \prod_{i=1}^{dim} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

where x_i is input features and dim represents the dimensionality of the function

- Non-linear oscillatory system (Feynman equation)

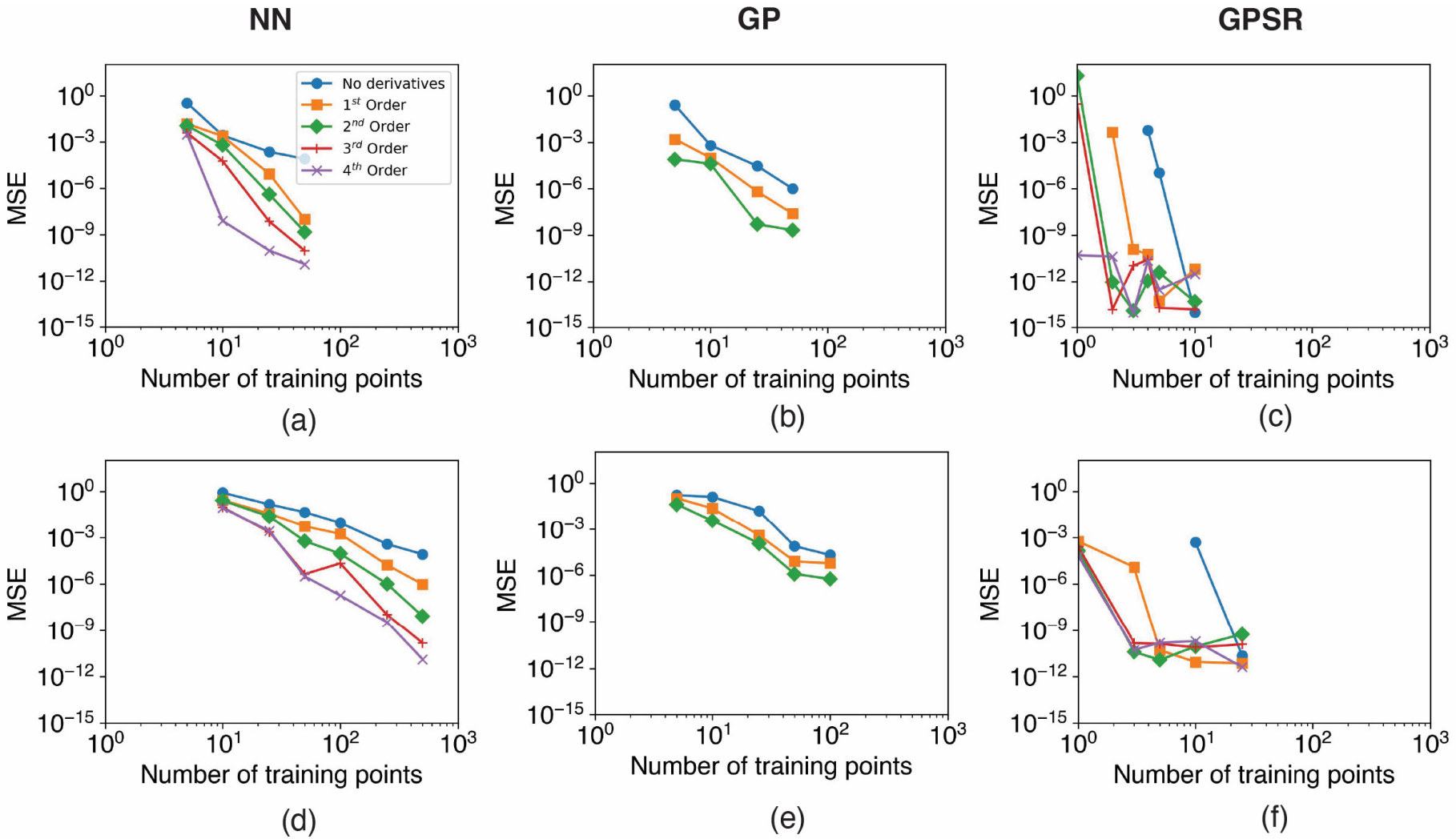
$$y(X, \omega, t, \alpha) = X[\cos(\omega t) + \alpha \cos^2(\omega t)]$$

where X, ω, t, α are features, and y is the output label



Numerical Experiments - Error

Multi-dimensional optimization problem



- Noticeable reduction in error for higher-order derivatives
- GPSR requires less training data

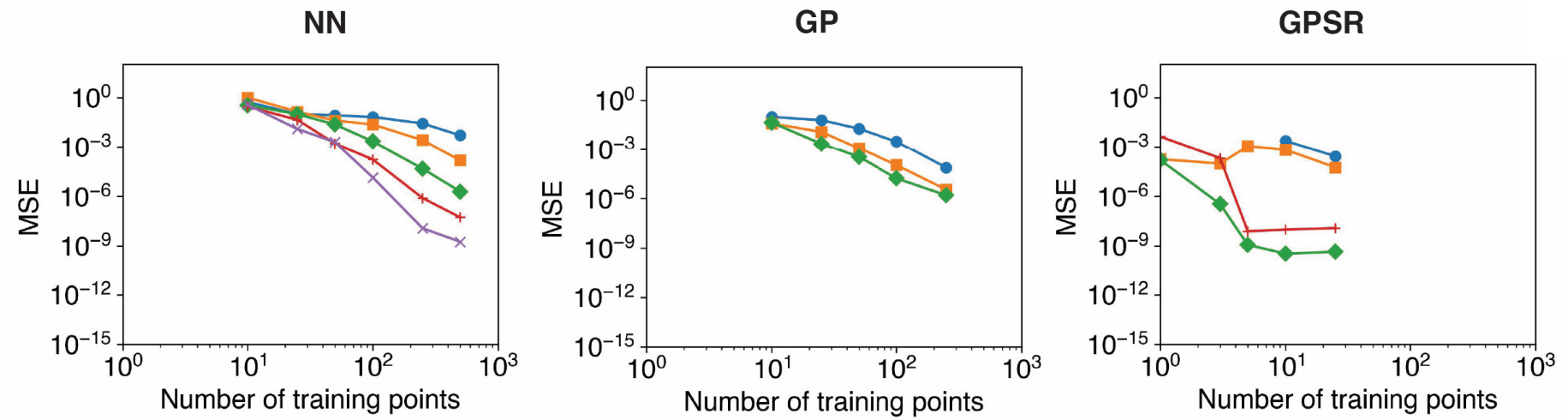
NN: Neural Networks

GP: Gaussian Process

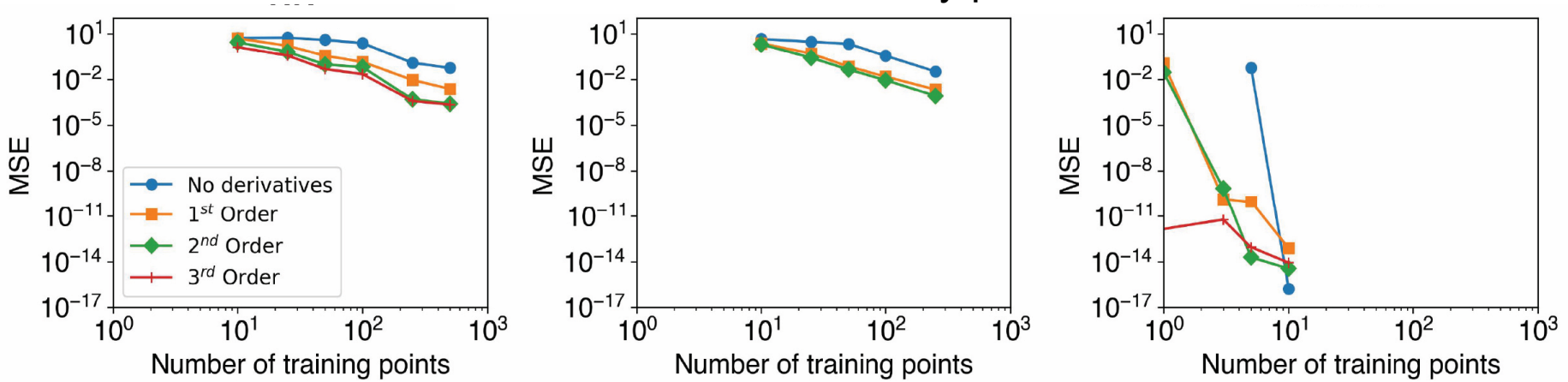
GPSR: Genetic Programming based SR

Numerical Experiments - Error

3D



Non-linear oscillatory problem



- Higher-order derivatives influential for complex problems

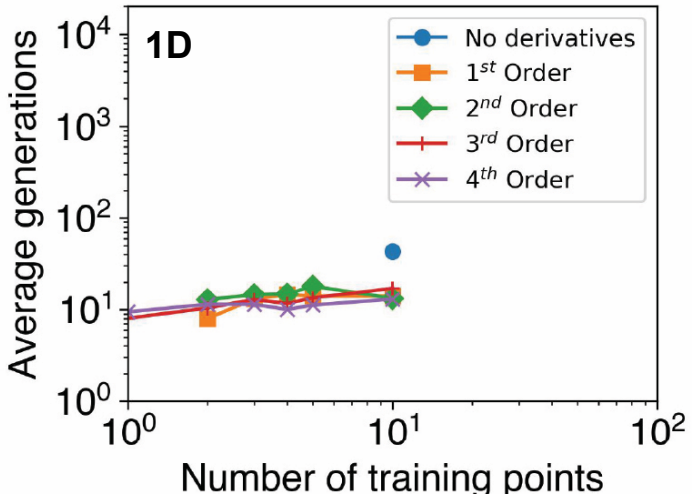
NN: Neural Networks

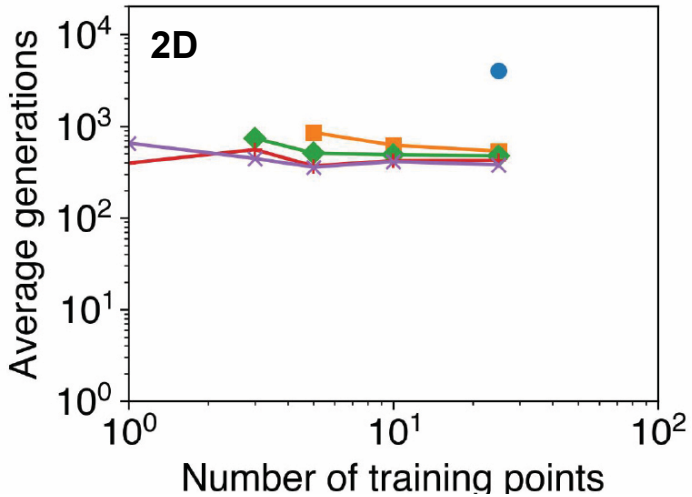
GP: Gaussian Process

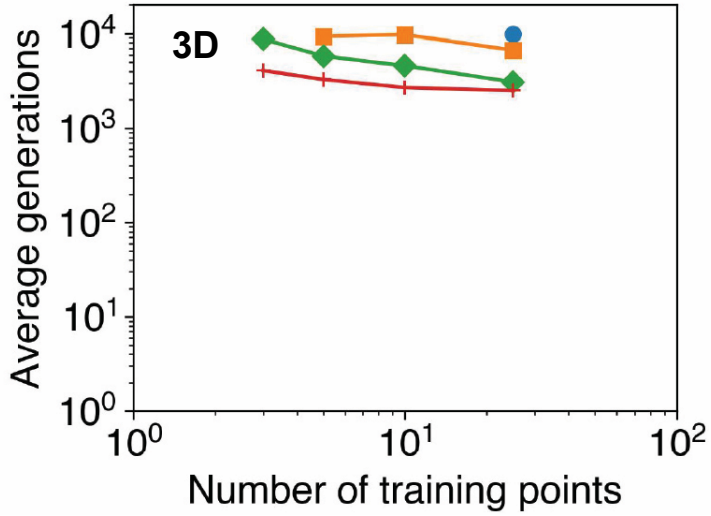
GPSR: Genetic Programming based SR

Numerical Experiments - Generations

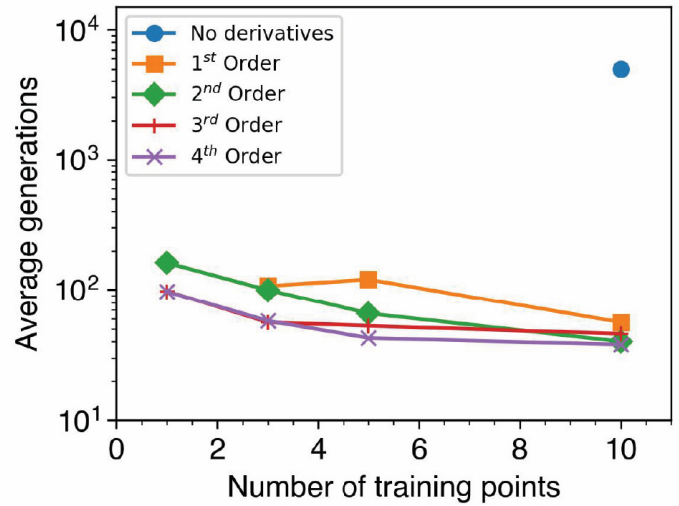
Multi-dimensional optimization problem







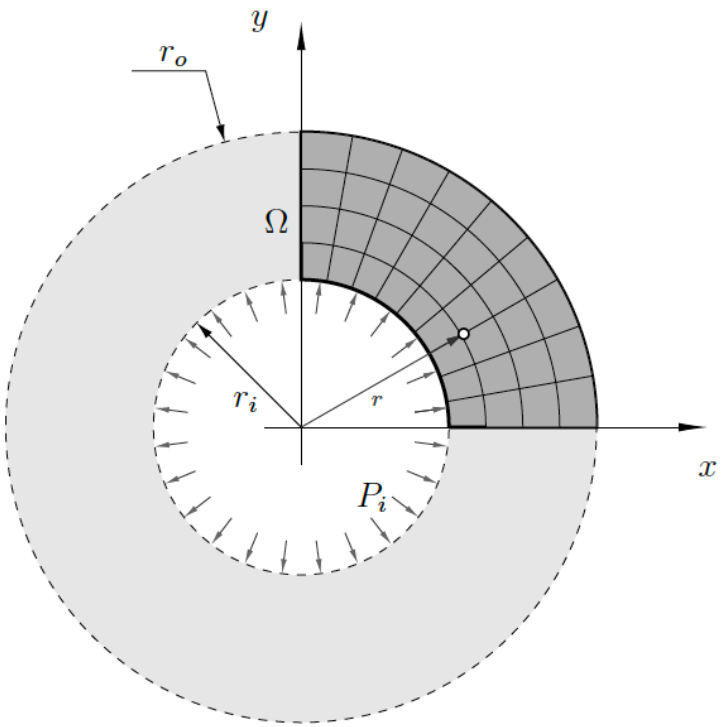
Non-linear oscillatory problem



- Derivatives help in faster evolution
- Higher-order derivatives helpful for complex equations

Application to Mechanics

- Problem: Thick-walled cylinder under internal pressure.
- Objective: predict displacement (u) given six features:
    1. Inner radius (r_i)
    2. Outer radius (r_o)
    3. Radial coordinate (r)
    4. Internal pressure (P_i)
    5. Young's modulus (E)
    6. Poisson's ratio (ν)
- HyPAD for derivatives of u up to 4th-order
- Bingo for free-form model evolution



$$u = \frac{rP_i(1+\nu)}{E}\left(\frac{r_i^2}{r_o^2 - r_i^2}\right)\left((1-2\nu) + \frac{r_o^2}{r^2}\right)$$

DITD - SR

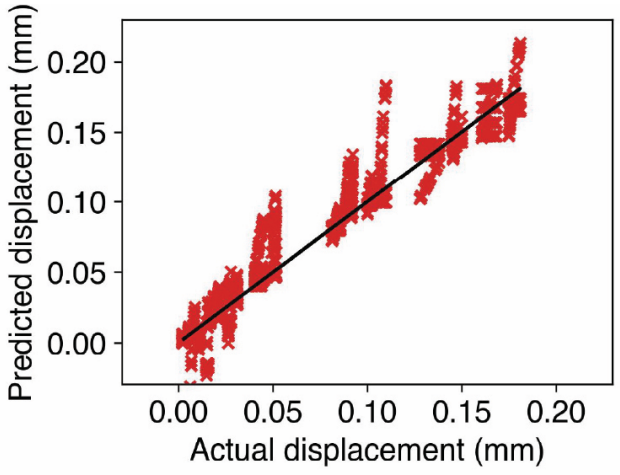
$$u = \frac{rP_i(1+\nu)}{E}\left(\frac{r_i^2}{r_o^2 - r_i^2}\right)\left((1-2\nu) + \frac{r_o^2}{r^2}\right)$$

Features	Min	Max	Unit
r_i	10	50	mm
r_o	15	60	mm
E	50	300	GPa
ν	0	0.4	-
P_i	0	100	MPa

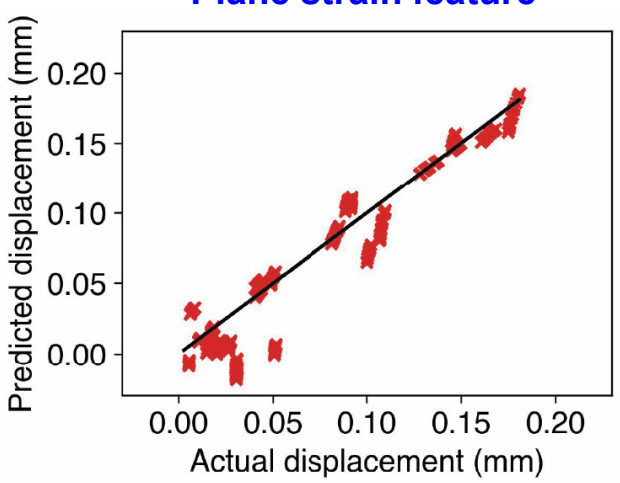
- 25 different feature combinations.
- 20,000 2D 6-noded triangular plane-strain elements.
- We chose 10 random points from the available data.
- Four different experiments:
  - No derivatives
  - 2nd order derivatives
  - 4th order derivatives
  - Plane strain feature ($A = \frac{(1+\nu)(1-2\nu)}{E}$)

Results using 10 training points

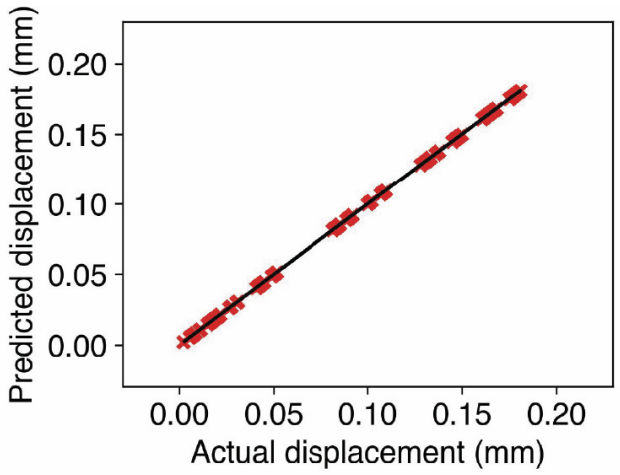
No derivatives



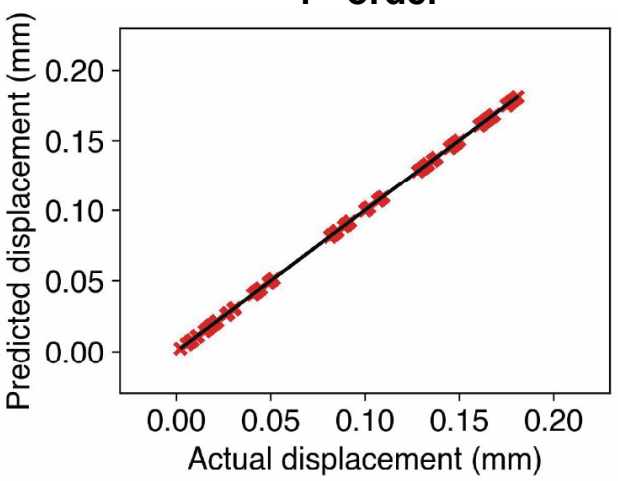
Plane strain feature



2nd order



4th order



- **No derivatives** case did not result in any accurate models
- Mechanics-guided **plane strain feature** imposes expert knowledge

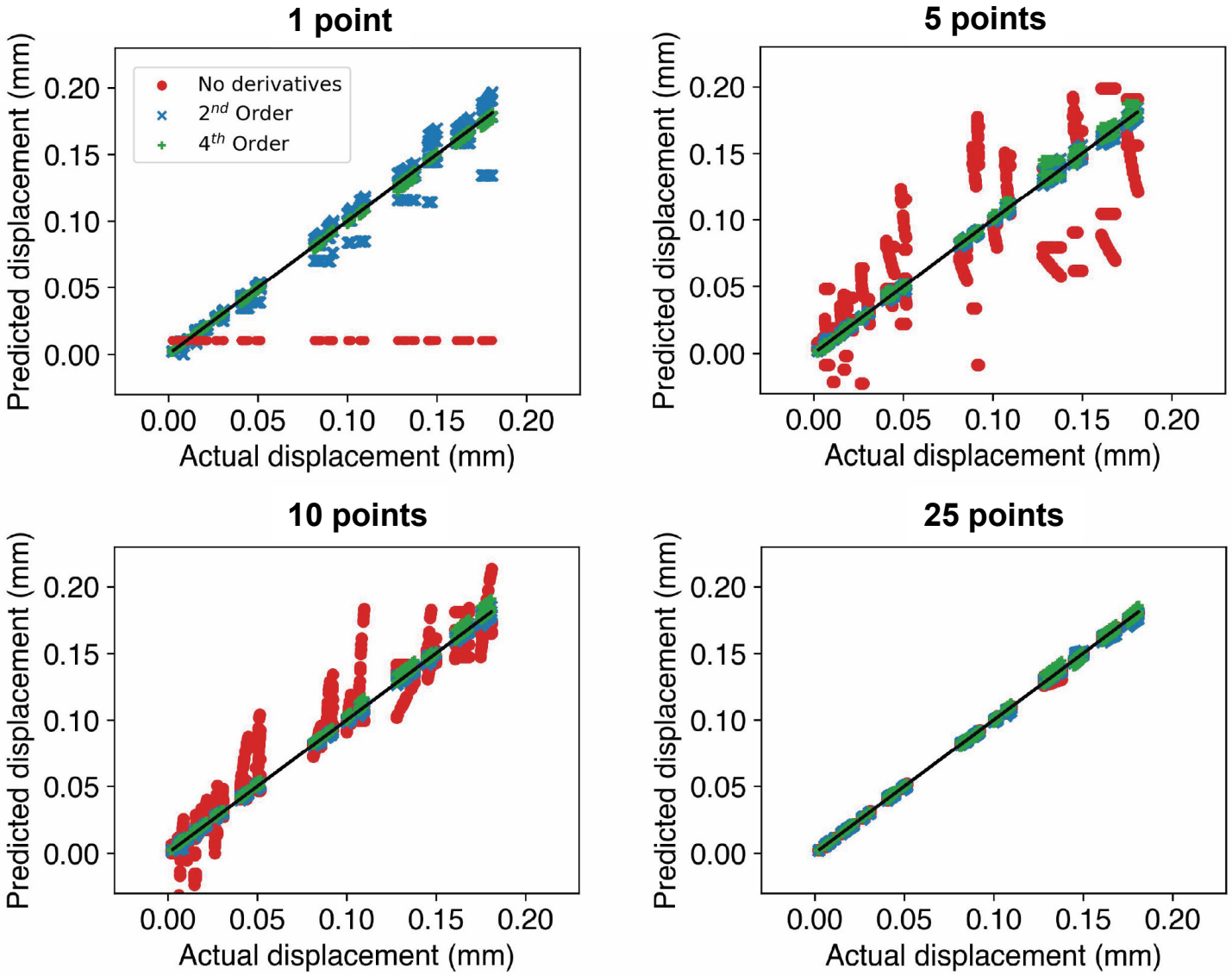
- **2nd-order derivatives**

$$\hat{u} = \frac{rP_i(1+\nu)}{E + 0.35}\left(\frac{r_i^2}{r_o^2 - r_i^2}\right)\left((1-2\nu) + \frac{r_o^2}{r^2}\right)$$

- **4th-order derivatives**

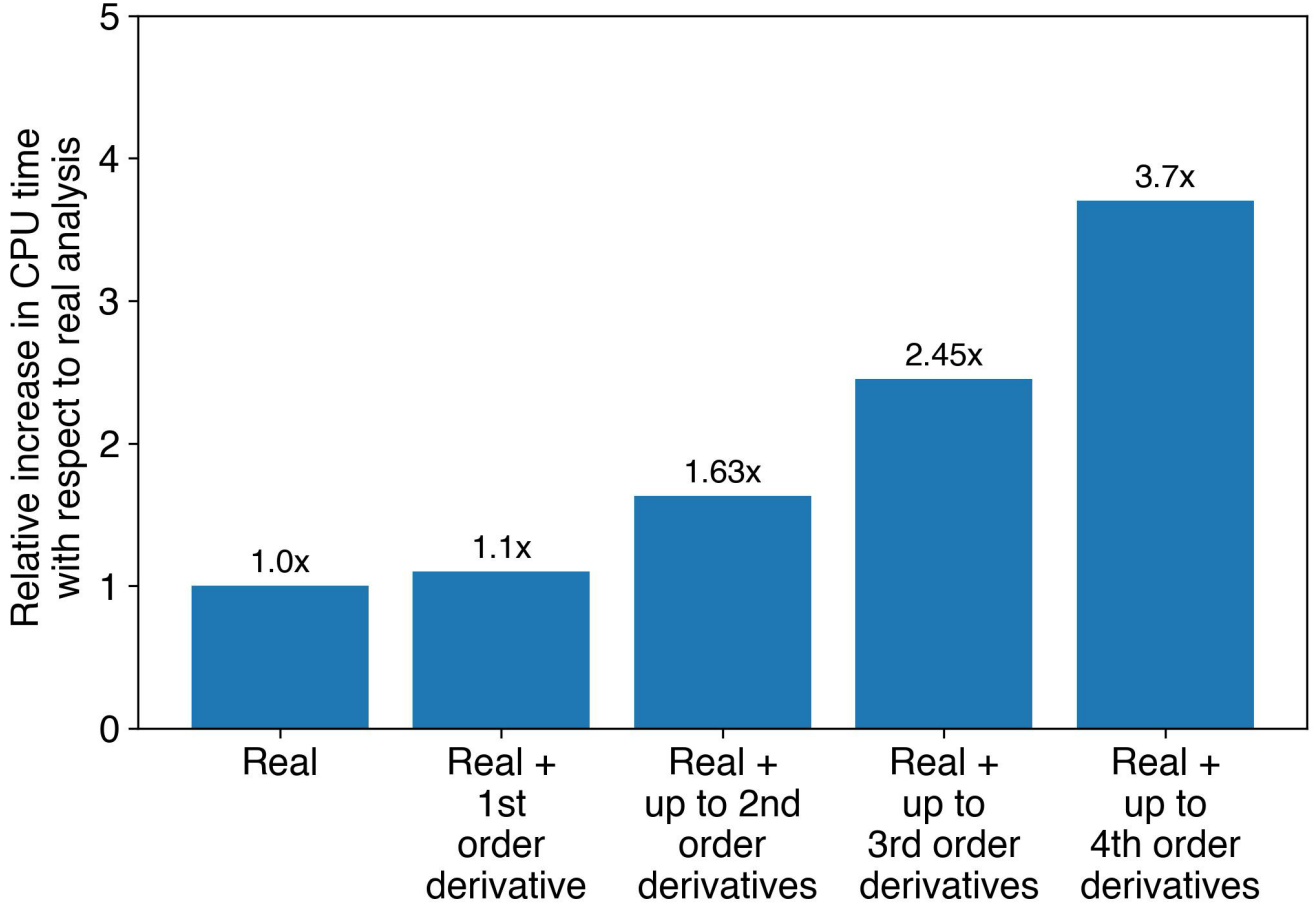
$$\hat{u} = \frac{rP_i(1+\nu)}{E}\left(\frac{0.54\left(\frac{0.41r_i}{r_o} + 1\right)}{\frac{r_o}{r_i - 0.19r_o} - 1.24}\right)\left((1-2\nu) + \frac{r_o^2}{r^2}\right)$$

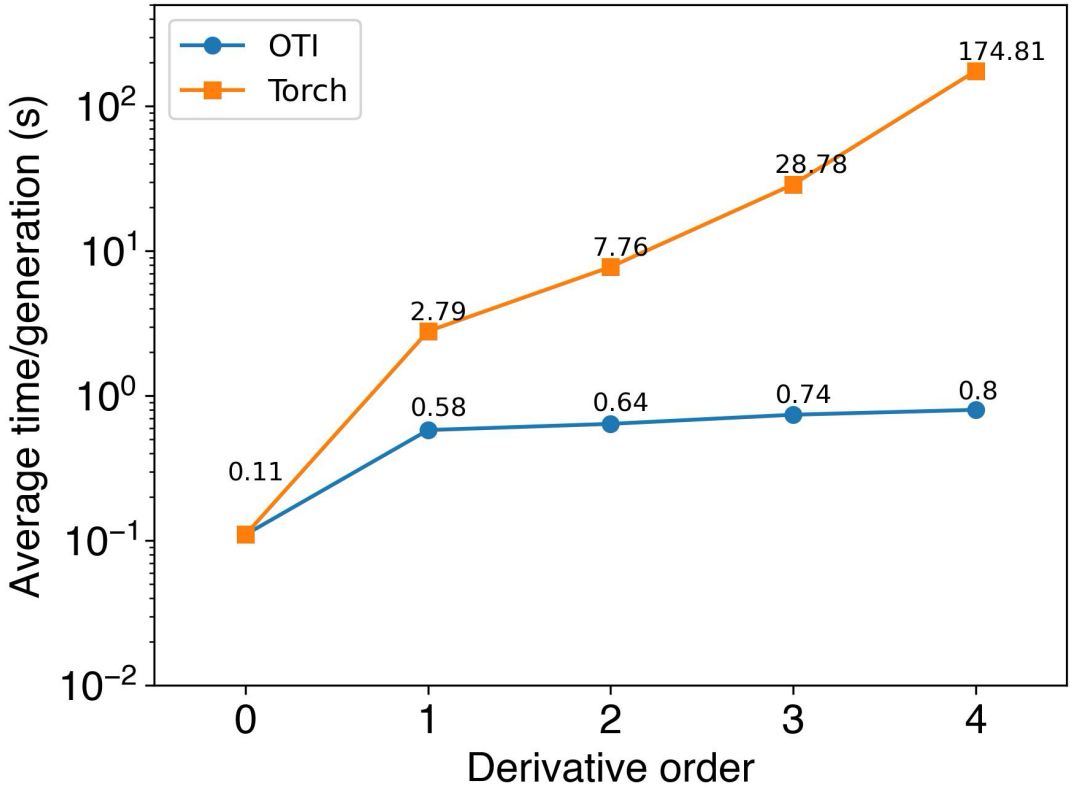
Benefits of higher-order derivatives



- Significant benefit of higher-order derivatives when the number of training points was low.
- With just one point, GPSR without derivatives would predict a constant at the solution of the dataset.
- Incorporating 2nd and 4th order derivatives improved the prediction R^2 of 0.976 and 0.999, respectively

Cost analysis





- Obtaining 4th order derivative costs 3.7x in data generation time
- With DITD approach, we need 1 point compared to 25 points without derivatives

Conclusion

- Physics- or mechanics-guides should be used *whenever possible*
- Derivative informed training data (DITD) can complement physics guides or be used independently
- Presented methods reduce training costs in four ways:
  - **~10x** fewer training data points required with DITD
  - **~100x** fewer model evolution steps required with DITD
  - **~50x** speed up using HYPAD for DITD generation
  - **~200x** speed up using OTI lib for computation ML model derivatives
- Note - using a numerical method to generate training data will inherently have error which increases with derivative order

Thank you