TU Delft

NONLINEAR DYNAMIC SYSTEM IDENTIFICATION IN VIBRATORY PILE DRIVING

An attempt in understanding pile-soil interaction from vibratory driving tests

2025

Civil Engineering

Altayeb Malik

Structural Engineering Track

Feb 2025

Nonlinear dynamic system identification in vibratory pile driving

An attempt in understanding pile-soil interaction from vibratory driving tests

MSc Thesis in Structural Engineering

by

Altayeb Malik

to obtain the degree of Master of Science at the Delft University of Technology
to be defended publicly on January XX, 2025 at 00:00

Thesis committee:

Chair:	Dr. Apostolos Tsouvalas
Supervisors:	Dr. Andrei Faragau
	Dr. Athanasios Tsetas
	Dr. Hayo Hendrikse
Place:	Faculty of Civil Engineering & Geo-sciences, Delft
Project Duration:	January, 2023 - January, 2024
Student number:	5736293

An electronic version of this thesis is available at `repository.tudelft.nl`.

QR code for repository

Deltares

Copyright © Altayeb Malik, 2025
All rights reserved.

Acknowledgment

I extend my gratitude to Andrei and Athanasios for their guidance and patience throughout this thesis.

To my family, who stood by me and supported me every step of the way, thank you for your unwavering love and encouragement.

To my family away from home at the TU Delft Global Initiative, your warmth and camaraderie made this experience unforgettable.

To lecturers at TU Delft, who provided the foundation and inspiration for my work.

A special tribute goes to my late father, whose love and sacrifices have brought me this far in life—this achievement is as much his as it is mine.

My heart is utterly unclouded that going through this program was an amazing choice, for all the experiences and interactions it has offered. I wish you all the very best and hope our paths cross again.

Summary

The study explores different system identification techniques that utilize machine learning, including the **Restoring Force Surface (RFS)** method and **Sparse Identification of Nonlinear Dynamics (PySINDy)**. These methods have potential to help uncover physical models for soil-pile interaction.

To test these methods , the research first applies them to well-known **benchmark systems**—simple mechanical models with known nonlinear behaviours. This ensures that the identification techniques work correctly before applying them to real pile-driving experiments.

The experimental data comes from **lab-scale vibratory pile-driving tests** using strain gauges and accelerometers. The study analyses how forces acting on the pile change over time, focusing on both the tip and shaft resistance. Various mathematical models are tested to see which best captures the nonlinear behaviour.

The results show that while RFS and PySINDy as well as custom RX models work well for simple nonlinear systems, they struggle to create a stable, generalizable model for pile-soil interaction. The complexity of soil behaviour, time-dependent effects, and measurement noise make accurate modelling challenging. Despite not producing a working surrogate pile driving model, the thesis gives some insights for future work on the topic.

0.1. Significance of the Study

Contributing to improved modelling to progress vibratory driving techniques, with an overall goal of adding value to the knowledge of general non-linear systems identification and choice of best approach for a given problem. Work in the context of offshore pile construction has an impact on the global energy transition as well as minimizing noise generated by driving and its impact on marine wild life. Being able to derive physical equations of motion of a dynamic system comes in far more useful than deriving simple empirical formulas, as it enables simulation for multiple conditions. The work focuses on the issue of pile driving but can be generally used for the identification of any non-linear system to an extent, thus having further applications in health and safety monitoring as well as design of systems exhibiting significant dynamic behaviour.

Acronyms

ARMAX Autoregressive moving average with exogenous inputs
BDF Backward differentiation formula
DTW Dynamic time warping
IVP Initial value problem
L-BFGS-B Limited-memory Broyden–Fletcher–Goldfarb–Shanno with box constraints
LTI Linear time-invariant
MI Mutual information
MLP Multiple layer perceptron
NARMAX Nonlinear autoregressive moving average with exogenous input
PCA Principal component analysis
PySINDy Python sparse identification of nonlinear dynamics
QQ Quantile-quantile
QQCC Quadratic quantum cross-correlation
RFS Restoring force surface
RK45 Runge-Kutta method of order 5(4)
RMSE Root mean square error
RX Autoregressive with exogenous inputs
SDOF Single degree of freedom
SVM Support vector machines

Contents

List of Figures

List of Tables

0.2. Introduction

Background

Nonlinear Dynamics in Engineering Systems

Engineering systems often exhibit nonlinear behaviour due to complex interactions between geometry, material properties, and boundary conditions. These nonlinearities—such as stiffening in structures and frictional slip in mechanical systems—complicate predictive modelling and introduce uncertainties in design processes. While linear approximations may suffice for simplified analyses, many real-world phenomena, including hysteresis, yield, and amplitude-dependent damping, require a nonlinear framework to accurately capture their dynamic response. In such contexts, system identification—the process of deriving mathematical models from observed data—becomes particularly challenging, necessitating methodologies to distinguish coupled while maintaining physical interpretability.

Nonlinear system identification

Nonlinear system identification plays a crucial role in understanding complex systems across various fields, enhancing modelling, monitoring & prediction, and optimisation processes.

Structural Vibration in Engineering: Structures such as buildings and bridges often display nonlinear behaviour under extreme conditions like large deformations or seismic activity. Identifying these nonlinear dynamics helps improve safety and design resilience, particularly against earthquakes and wind loads.

Turbo-machinery and Fluid Flow: Turbulent flow and high-velocity fluid systems are inherently nonlinear. Understanding these nonlinearities is vital for optimising efficiency and preventing failure in engines, turbines, and fluid transport systems.

Electrical Circuits: Nonlinear components such as diodes and transistors in electrical circuits can cause signal distortion. Identifying these nonlinearities is crucial for designing more efficient power converters, amplifiers, and noise reduction systems in electronic devices.

Chemical Reactions: Nonlinear behaviours in chemical reactions, such as combustion, arise from temperature and concentration dependencies. Accurate identification of these behaviours improves control in industrial processes, ensuring efficiency and reducing emissions.

Pile Driving

Offshore wind turbines worldwide are most often supported by mono-piles. Traditionally, these foundations are installed using impact hammering, which generates high levels of underwater noise, posing a major threat to marine ecosystems. The environmental risks associated with impact pile driving have spurred interest in alternative installation methods, with vibratory pile driving emerging as a promising option. Unlike impact hammers, vibratory driving uses oscillatory forces to ”fluidize” the surrounding soil by having it resonate to a degree to the waves generated by driving, reducing friction and enabling smoother pile penetration. This technique has been shown to lower underwater noise emissions compared to impact hammers, potentially aligning with stricter marine noise regulations [25].

However, the use of vibratory driving is extremely limited in offshore conditions where contractors prefer the much more common impact driving, leading to a lack of field observations and major open questions regarding noise and drivability [31]. A key challenge is the nonlinear nature of the pile-soil interaction, which introduces significant uncertainty. Existing empirical methods for estimating required vibratory driving forces are often scale-dependent, partially validated, and lack generalizability across different soil profiles and pile geometries. Given the increasing demand for offshore wind energy and the need for sustainable foundation installation methods, there is a critical need for a systematic identification framework that can leverage both existing and future datasets.

Nonlinear Soil-Pile Interaction

The fundamental challenge in vibratory pile driving lies in the nonlinear soil-pile interaction. Surrounding soil has properties that evolve dynamically during driving, Necessitating frameworks capable of capturing amplitude-dependent stiffness, hysteresis, and transient effects. Additionally, seabed and soil strata variations require case-by-case modelling, making it difficult to determine the optimal vibratory driving characteristics for achieving a desired pile depth with sufficient certainty [31].

Current physics-based models, though rigorous, rely on idealized assumptions (e.g., homogeneous soil layers) and require computationally expensive finite-element analyses. Conversely, purely data-driven approaches struggle with sparse experimental datasets, a common limitation in large-scale geotechnical testing. Bridging this gap—leveraging measured data to construct interpretable, physics-informed and reduced order surrogate models—is critical for advancing vibratory pile driving from a niche technique to a mainstream solution that supports offshore wind energy expansion and broader marine infrastructure development.

0.3. Research Problem

Identifying and mathematically formulating a surrogate pile driving model that captures non-linearities arising from pile and soil interaction. That of which would make a great alternative to widespread empirical approaches used to estimate required driving forces. The model will be built to fit measurements from scale driving tests, and will be based on mathematical and machine learning based system identification methods that perform well with relatively limited datasets, in attempt to identify key components representing the system's equation of motion.

0.4. Objectives

The study aims to assess the validity and applicability of some system identification methods in modelling the nonlinear dynamics occurring during pile driving, particularly in the context of pile-soil interaction. It aims to offer possible improvement of results by incorporating knowledge about the system as guidance to the approach to counteract lack of extensive measurements and tests. Thus, improving the understanding of the physical process by identifying a functional form of the non-linear soil behaviour. A main goal is to use the identified equation of motion to accurately simulate the pile driving tests paving the way to use such formulation of an equation of motion on larger scale experiments and then codify it into actual practice. These objectives can be summarised as:

- **Validate System Identification Methods**: Assess the ability of techniques like Restoring Force Surface (RFS) and Sparse Identification of Nonlinear Dynamics (PySINDy) to uncover nonlinear components within noisy, limited data.
- **Decipher Soil-pile interaction Behaviour**: Derive functional forms of soil resistance (e.g., cubic stiffness, hyperbolic friction) from experimental measurements.
- **Enable Predictive Simulations**: Develop reduced-order models to simulate vibratory driving for future applications, such as offshore wind turbine installations, i.e. derive reliable predictions of response to different amplitudes and frequencies of excitation.

0.5. Research Questions

1. Can a non-linear dynamic system be identified with limited measurements?
2. Which identification methods are best suited in context of the research problem?
3. How is the system identification process physics-informed? and can more prior-knowledge be included?
4. What is the criteria to qualify an identified system's fidelity to actual physics?
5. Would a time-invariant identified system produce reasonably close responses, to be later refined by incorporating time-variant components? (See Subsection 1.1)

0.6. Outline of the thesis

Chapter 1: Introduction

Background

- Environmental challenges of impact pile driving (e.g., underwater noise).
- Vibratory pile driving as an eco-friendly alternative.
- Nonlinear pile-soil dynamics and need for systematic identification.

Research Problem

- Develop a surrogate model for nonlinear pile-soil interaction with limited data.
- Bridge empirical and physics-based modelling approaches.

Objectives

- Validate system identification methods (e.g., RFS, PySINDy).
- Improve understanding of soil behaviour through functional forms.
- Enable accurate simulations for large-scale applications.

Significance

- Contributions to offshore wind energy and marine ecosystem preservation.
- Advancements in nonlinear system identification.

Research Questions

- Feasibility of identification with limited data.
- Suitability of methods for pile-soil dynamics.
- Role of physics-informed modelling and model fidelity.

Thesis Outline

- Study nonlinear systems, simulate benchmark cases, apply methods to experimental data, validate results.

Chapter 2: Mathematical Framework

System Identification

- Limitations of linear models.
- Challenges in nonlinear dynamics (e.g., frequency mixing, time variance).

Key Methods

- **RFS**: Chebyshev-polynomial-based interpolation.
- **Parametric Models**: RX/ARMAX/NARMAX, feature selection (Lasso, Elastic Net).
- **Optimization**: Physical constraints, L-BFGS-B tuning.

Chapter 3: Benchmark Nonlinear SDOF Systems

Case Studies

- Duffing oscillator, friction-slip model, 2-DOF system.

Methodology

- Benchmark system simulation, noise, state-space formulation.
- Validation & comparison.

Results
- Performance of RFS, custom RX, PySINDy in capturing nonlinearities.
- Impact of noise, feature selection, over-fitting.

Chapter 4: Experimental Analysis and Results

Experimental Setup
- Lab-scale vibratory pile-driving tests (SIMOX project).
- Instrumentation: strain gauges, accelerometers, force calculations.

Data Processing
- Signal filtering, separation of tip and shaft resisting forces.

Identification Results
- Polynomial/Chebyshev-based force models, cross-validation.
- Stability challenges and alternative formulations.

Simulation and Validation
- State-space simulations (BDF, RK45 solvers).
- Limitations in capturing higher harmonics, time-variant effects.

Chapter 5: Conclusions & Recommendations

1

Mathematical Framework

1.1. Identification of linear systems

Linear system identification, which seeks to ascertain mathematical models of linear dynamic systems from vibration measurements, is a recognized field of study. Modal testing and analysis methods are readily available commercially. In linear systems, the transfer function that connects the system's input to its output stays invariant across all excitation levels. Consequently, the mathematical model derived from identification at a specific operating point can subsequently be employed for predictions at an alternative operating point. In non-linear systems, deriving a universal mathematical model through system identification at a single excitation level is impossible. A model derived under specific operating conditions can, at most, yield the corresponding linear system at that juncture. Consequently, the identification of non-linear systems diverges from traditional linear system identification, as no simple transfer function exists to represent it nor the analytical methods for the majority of such systems.This thesis encompasses the domain of non-linear system identification and focused on it for a single degree of freedom systems, the main goal is to identify the nonlinear aspects of pile-soil interaction in pile driving.

Linear Time Invariant Linear time-invariant (LTI) Systems

[28] Linear Time Invariant (LTI) system theory focuses on the mathematical relationship between input and output signals, where the system's behaviour is both linear and time-invariant. A system is considered **linear** when it satisfies the *superposition principle*, which encompasses two key properties:

- **Additivity:** $f(x+y) = f(x) + f(y)$
- **Homogeneity:** $f(\alpha x) = \alpha f(x)$

This means that the response of the system to a combination of inputs is equal to the sum of its responses to each input individually, scaled accordingly. Furthermore, **time-invariance** implies that the system's response to a specific input remains consistent over time.

In contrast, **nonlinear systems** do not adhere to these principles, and as a result, parts of the signal may be transferred to other frequencies, causing nonlinear distortions.

Time-Invariant and Time-Variant Systems

A system is **time-invariant** if its behaviour does not depend on the passage of time. i.e. for the same input the system produces the same response regardless of what the system underwent before. Mathematically, this can be expressed as:

$$\beta(t) = N[u(t)] \implies \beta(t-\tau) = N[u(t-\tau)]$$

where τ is any time shift. Systems for which the above equation does not hold are called **time-variant**.

1.2. Key Factors

Model identification is a crucial step in understanding and predicting the behaviour of complex systems. It involves selecting the appropriate model structure and parameters to represent the system accurately.

This process considers factors, each playing a vital role in the final model's effectiveness as shown in figure 1.1.

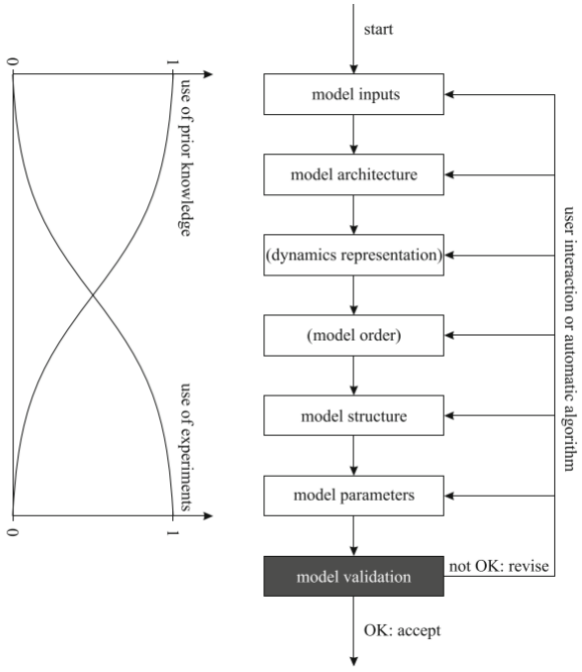


Figure 1.1: Key aspects of system identification models[24]

1.3. Model Inputs

The first hurdle lies in choosing the model inputs. These are the variables that influence the system's output. Several approaches exist. One is to include all available inputs, but this creates a high-dimensional problem requiring vast amounts of data and extensive computational resources. Another option is to try all input combinations, which becomes impractical for models with many variables. Supervised techniques, such as correlation analysis or evolutionary algorithms, can also be employed, aiming to select inputs that maximize model accuracy.

1.4. Choice of Excitation

The second critical factor is the selection of excitation signals to analyse the behaviour of the system. Signal design is influenced by prior understanding of the process and the purpose of the model. In the event that active excitation is not feasible, it is recommended that representative data sets be employed to ensure that the operating conditions are adequately represented in order to evaluate the reliability of the model and reduce extrapolation errors. Harmonic excitation should be sufficient for pile driving as for all intents and purposes it's the driving force to be applied on site.

1.5. Model Architecture

- Intended use: Will the model be used for simulation, optimization, control, fault detection, or something else?
- Problem dimensionality: What is the number of relevant inputs and outputs.
- Data availability and quality: Sparse or noisy data may favour models with global approaches, averaging out disturbances, with a risk of losing attributes of the system.
- Development time constraints: Balancing model complexity and training time is crucial. Simpler models with faster training times might be preferred if development time is limited.

Additional factors influencing architecture selection include user experience, available tools, the need for offline or online learning, and customer acceptance (interpretability of the model).

1.6. Model Order, Structure, and Complexity

Determined by the number of terms or parameters. Balancing the ability to capture system dynamics with avoiding over-fitting is crucial. Trial-and-error and prior knowledge often play a significant role in selecting the optimal model type. Machine learning based techniques can automate the process, while manual approaches offer more user control. Determining the right balance between automation and user judgement is necessary.

1.7. Overview of Relevant Identification Methods

Nonlinear structural dynamics identification has been studied over the years [27]. Some prominent methods are:

- Spectral analysis
- Volterra and Wiener series
- Nonlinear auto-regressive moving average models
- Restoring force method
- Describing function methods
- Direct parameter estimation
- Hilbert transforms
- Wavelet transforms
- Neural Networks

a few prominent time domain methods are mentioned in brief:

Restoring Forces Surface Method Restoring force surface (RFS)

Proposed in 1982[18], this approach fits a surface (using Chebyshev or Legendre polynomials) to the resisting force data based on displacement/velocity or displacement/acceleration. It finds a suitable polynomial to represent the force variations. Further work has been done to treat bias and over-fitting as well as implement it in the frequency domain.

Sparse Regression (e.g. using Python sparse identification of nonlinear dynamics (PySINDy) library [4])

This method aims to identify the most significant factors influencing the system's behaviour by ignoring negligible ones. It assumes a linear combination of terms and gradually eliminates terms that don't contribute significantly. This approach utilizes derivative system variables, offering additional insights.

Neural Networks [19](e.g. Multi-Layer Perceptron)

While powerful machine learning algorithms, neural networks require a large dataset for effective training. This method often creates a ”black box” model, making it difficult to understand the internal workings of the system as the only physics involved are the ones used to formulate the problem.

NARMAX Models[1]

Proposed by Billings in 1981, this model (Nonlinear Auto-regressive Moving Average with eXogenous input) can be thought of as an enhanced linear regression that incorporates past and future states of the system. Its parameters are solved using linear recursive least squares or other iterative methods. While it may not reveal specific details about the system, it allows for simulations. The correlation of state variables can be investigated and offer much insight into the underlying mechanisms.

Other Methods

While powerful modal-based methods like nonlinear normal modes hold promise, they require prior knowledge of the system, which isn't available in this case. Similarly, frequency domain methods wouldn't be suitable since the tests weren't designed to capture a broad range of frequencies for a comprehensive Frequency Response Function (FRF). Another machine learning approach is Support vector machines (SVM) [6] . It works well with smaller datasets and offers some level of interpretability. However, its ability to capture complex relationships may be limited.

1.8. Restoring Force Surface Method

The restoring force surface method[18] is used to identify the nonlinear characteristics of the system. The method is rooted in the equation of motion expressed as:

$$m\ddot{y} + f(y, \dot{y}) = x(t) \tag{1.1}$$

Where m is the mass of the system, y is the displacement, $\dot{y}$ is the velocity, and $f(y, \dot{y})$ represents the restoring force, which can be a general function of $y(t)$ and $\dot{y}(t)$.

If the mass m is known and both the excitation $x(t)$ and the acceleration $\ddot{y}(t)$ are measured, then $f(y, \dot{y})$ can be computed as:

$$f(y(t), \dot{y}(t)) = x(t) - m\ddot{y}(t) \tag{1.2}$$

This method allows the identification of the restoring force at each sampling point, enabling the construction of a force surface in the phase plane via an interpolation method, e.g. 2D interpolation,k-Nearest Neighbour or other machine learning approaches.

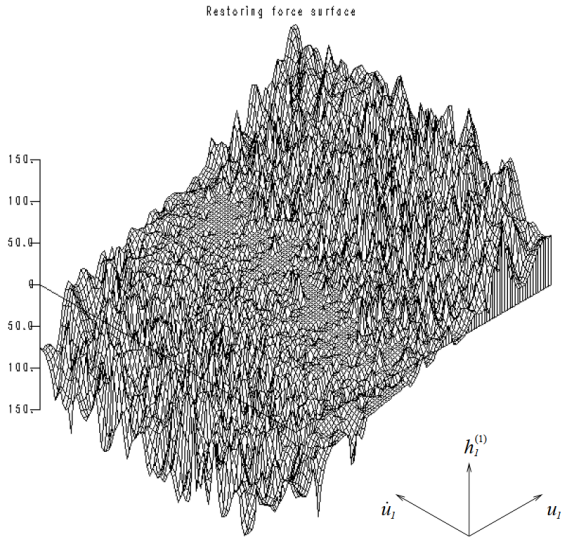


Figure 1.2: Interpolated restoring force for a 3-DOF piecewise linear oscillator [32]

The interpolated surface is then expanded into the general form:

$$f(y, \dot{y}) = \sum_{i=0}^{m} \sum_{j=0}^{n} C_{ij} T_i(y) T_j(\dot{y}) \tag{1.3}$$

Where $T_i(y)$ and $T_j(\dot{y})$ are Chebyshev polynomials, and C_{ij} are the coefficients of fit to be determined.

Chebyshev polynomials are a class of orthogonal polynomials that arise naturally in approximation theory and numerical methods. Due to their excellent convergence properties, they are often used in interpolation, function approximation, and spectral methods for solving differential equations. see Appendix D. The most important property is that they can form an orthogonal basis allowing for separation of equations for displacement and velocity allowing direct calculation of the fit coefficients C_{ij}

These polynomials are named after the Russian mathematician Pafnuty Chebyshev. In this document, we focus on the Chebyshev polynomials of the first kind, denoted by $T_n(x)$, which play a significant role in minimizing errors during polynomial approximation.

The orthogonality of the Chebyshev polynomials leads to the integral equation: The coefficients C_{ij} are determined using a double integral over the phase space:

$$C_{ij} = X_i X_j \int_{-1}^{1} \int_{-1}^{1} w(y) w(\dot{y}) T_i(\zeta(y)) T_j(\zeta(\dot{y})) f(y, \dot{y})\, dy\, d\dot{y}, \tag{1.4}$$

where:

- $w(y) = \frac{1}{\sqrt{1-y^2}}$ is the weight function for Chebyshev polynomials,
- $X_i = \frac{1}{\pi}(1 + \delta_{i0})$ is a normalization factor,
- δ_{ij} is the Kronecker delta.

In discrete form, the integral is approximated using numerical quadrature:

$$C_{ij} \approx X_i X_j \sum_{k=1}^{n} \sum_{l=1}^{n} w_k w_l T_i(\zeta(y_k)) T_j(\zeta(\dot{y}_l)) f(y_k, \dot{y}_l) \Delta y \Delta \dot{y}, \tag{1.5}$$

where w_k and w_l are weights associated with the quadrature points.

To facilitate the calculation, the coordinates can be normalized as:

$$\zeta(y) = \frac{y - \frac{1}{2}(y_{\text{max}} + y_{\text{min}})}{\frac{1}{2}(y_{\text{max}} - y_{\text{min}})} \tag{1.6}$$

$$\dot{\zeta}(\dot{y}) = \frac{\dot{y} - \frac{1}{2}(\dot{y}_{\text{max}} + \dot{y}_{\text{min}})}{\frac{1}{2}(\dot{y}_{\text{max}} - \dot{y}_{\text{min}})} \tag{1.7}$$

The restoring force surface is then approximated as:

$$f(y, \dot{y}) = \sum_{i=0}^{m} \sum_{j=0}^{n} C_{ij}^{\zeta} T_i(\zeta(y)) T_j(\zeta(\dot{y})) \tag{1.8}$$

The transformation of coordinates $\zeta(y)$ and $\dot{\zeta}(\dot{y})$ allows the data to be mapped onto a standardized interval, ensuring that the Chebyshev polynomials are properly applied for further calculations.

Interpolation and Approximation

To address the interpolation of the force surface on a regular grid, a bilinear interpolation procedure is applied. This procedure involves identifying the grid square containing the point $(y, \dot{y})$ and interpolating the force value $f(y, \dot{y})$ using surrounding grid points.

The interpolation process it the core decision of this method, the one with the seemingly best result using a Neural Network (Multiple layer perceptron (MLP)) when judging simply the RMSE of a known restoring force surface, e.g. for a Duffing oscillator.

Chebyshev Polynomial Approximation

The restoring force $f(y, \dot{y})$ is approximated using Chebyshev polynomials:

$$f(y, \dot{y}) = \sum_{i=0}^{m} \sum_{j=0}^{n} C_{ij} T_i(\zeta(y)) T_j(\zeta(\dot{y})), \tag{1.9}$$

where $T_i(x)$ are the Chebyshev polynomials of the first kind, and C_{ij} are the coefficients to be computed.

Reconstruction of Restoring Force

Using the computed coefficients C_{ij}, the restoring force $f(y, \dot{y})$ can be reconstructed:

$$f(y, \dot{y}) = \sum_{i=0}^{m} \sum_{j=0}^{n} C_{ij} T_i(\zeta(y)) T_j(\zeta(\dot{y})). \tag{1.10}$$

one can derive a mathematical expansion of the Chebyshev into polynomials however it doesn't have an inherent physical meaning, i.e. you won't see the linear stiffness component separately from the rest.

1.9. Identification using Parametric Time Domain Models

In system identification, several models are commonly used to represent dynamical systems in the time domain. These include the Autoregressive with exogenous inputs (RX), Autoregressive moving average with exogenous inputs (ARMAX), and Nonlinear autoregressive moving average with exogenous input (NARMAX) models, each differing in how they incorporate the input-output relationship and how they handle nonlinearity.

RX (Auto-regressive with eXogenous inputs) Model

The RX model is a simpler linear time-invariant (LTI) model that relates the current output to past outputs (auto-regressive part) and past inputs (exogenous part). It is often used when the relationship between input and output is assumed to be linear and straightforward. The mathematical form of the RX model is given by:

$$y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) = b_1 u(t-1) + \cdots + b_m u(t-m) + e(t), \tag{1.11}$$

where:

- $y(t)$ is the output at time t,
- $u(t)$ is the input at time t,
- $e(t)$ is the white noise (error term),
- a_i and b_j are model parameters to be estimated.

The RX model assumes a purely linear relationship between inputs and outputs, making it computationally efficient and easier to implement. It is particularly well-suited for applications where the dynamics are relatively simple and noise does not play a significant role. Parameter estimation for RX models is typically performed using least squares methods [16].

ARMAX (Auto-regressive Moving Average with eXogenous inputs) Model

The ARMAX model extends the RX model by including a moving average (MA) term that models the noise as a linear combination of past errors. The ARMAX model is expressed as:

$$y(t)+a_1 y(t-1)+\cdots+a_n y(t-n) = b_1 u(t-1)+\cdots+b_m u(t-m)+c_1 e(t-1)+\cdots+c_l e(t-l)+e(t), \tag{1.12}$$

where c_i are the coefficients for the moving average part.

The inclusion of the MA term allows the ARMAX model to account for dynamic noise behaviour, making it more robust for systems where noise dynamics influence the output. However, this added complexity requires iterative parameter estimation methods [16].

NARMAX (Nonlinear Auto-regressive Moving Average with eXogenous inputs) Model

The NARMAX model builds on the ARMAX model by incorporating nonlinear relationships between inputs, outputs, and noise. This allows the model to capture complex input output interactions as well as memory like behaviour, providing greater flexibility for real-world applications where system behaviour often deviates from linear assumptions.

Mathematically, the NARMAX model is expressed as:

$$y_k = \mathcal{F}\left[y_{k-1}, \ldots, y_{k-n_y}, x_k, x_{k-1}, \ldots, x_{k-n_x}, e_{k-1}, \ldots, e_{k-n_e}\right] + e_k, \tag{1.13}$$

where $\mathcal{F}$ is a nonlinear function that governs the relationship among inputs, outputs, and noise. The NARMAX model is ideal for applications involving strong nonlinearities and significant noise effects [1].

Comparison of RX, ARMAX, and NARMAX Models

The key differences among these models lie in their complexity and ability to handle nonlinearity and noise:

- **RX Model:** Best suited for systems with simple, linear dynamics and negligible noise. It is computationally efficient but lacks the ability to model complex behaviours.
- **ARMAX Model:** A linear model that extends RX by modelling noise dynamics. Useful for systems where noise plays a significant role but where the dynamics remain linear.

- **NARMAX Model:** The most flexible and robust among the three, capable of modelling nonlinear dynamics and accounting for noise. This comes at the cost of increased computational complexity and the need for advanced parameter estimation techniques.

Model Structure and Order

The structure of a NARMAX model is designed to capture both the deterministic and stochastic components of a system. The **model order**, plays a crucial role in defining the complexity of the model [1]. Higher model orders allow the model to capture more complex dynamics by considering longer memory effects, but they also increase the risk of over-fitting, particularly when the available data is limited. Therefore, the selection of the appropriate model order is a critical step in the modelling process and requires careful consideration of both the system dynamics and the quality of the data [5]. Additionally very high order models run into the issue of overflow when working with ODE solvers, and normalization isn't always possible, refer to section 2.7.

Advantages of NARMAX Models

The NARMAX model offers several advantages that make it a preferred choice for modelling nonlinear systems. One of the key benefits is its ability to capture complex time and history relationships between the inputs, outputs, and noise components [12]. This is particularly valuable in real-world systems where time-invariant models fail to adequately describe the system dynamics. Additionally, the NARMAX model is highly flexible, allowing for the use of various types of functions for the nonlinear mapping $\mathcal{F}(\cdot)$.

Another significant advantage of the NARMAX model is its ability to model uncertainty. By including noise terms explicitly in the model structure, the NARMAX model can account for uncertainties, unaccounted for dynamics, and measurement noise, providing a more accurate and robust representation of the system [5].

Commonly Used Functions in NARMAX Models

The NARMAX model's performance is largely dependent on the features selected. Other functions including neural networks, wavelets, and radial basis functions [1] are also utilized in more complex situations, even though polynomial functions are the most widely used because of their simplicity and ease of implementation [5]. The aforementioned functions are appropriate for modelling highly nonlinear systems because they offer more flexibility and are able to identify more complex patterns in the data, but I'd find it more appropriate and interpretable to use more recognizable forms of features instead of machine learning type of global optimizers.

Model Formulation

For a 1 DOF system, the internal force $F_{\text{int}}(t)$ is reconstructed as a weighted sum of features:

$$F_{\text{int}}(t) = \sum_i c_i f_i(y(t), \dot{y}(t)),$$

where:

- c_i are the coefficients learned through regression,
- $f_i(y(t), \dot{y}(t))$ are the features i.e. functions e.g. polynomials, trigonometric functions etc..

Equation of Motion

The equation of motion combines the internal force and external forcing term, $F_{\text{ext}}(t)$, to model system dynamics:

$$m\ddot{y}(t) = -F_{\text{int}}(t) + F_{\text{ext}}(t). \tag{1.14}$$

Substituting the feature-based representation of $F_{\text{int}}(t)$:

$$m\ddot{y}(t) = \sum_i c_i f_i(y(t), \dot{y}(t)) + F_{\text{ext}}(t).$$

For the simplest parametric model this formulation can be expanded to :

RX Model

$$\begin{bmatrix} F_{\text{int},t} \\ F_{\text{int},t+1} \\ F_{\text{int},t+2} \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & y_t & \dot{y}_t & f_1(y_t, \dot{y}_t) & \cdots \\ 1 & y_{t+1} & \dot{y}_{t+1} & f_1(y_{t+1}, \dot{y}_{t+1}) & \cdots \\ 1 & y_{t+2} & \dot{y}_{t+2} & f_1(y_{t+2}, \dot{y}_{t+2}) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \end{bmatrix}$$

The next step after getting some idea of main time invariant features of the system, is adding history terms, choice of history terms is discussed further one, and in addition to that is the inclusion of noise into the model (by judging the internal force residuals or using any known properties of the instrumentation system) the formulation of the problem becomes the well known **NARMAX** Model

$$\begin{bmatrix} F_{\text{int},t} \\ F_{\text{int},t+1} \\ F_{\text{int},t+2} \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & y_t & y_{t-1} & \dot{y}_t & e_t & f_1(y_t, \dot{y}_t) & f_1(y_{t-1}, \dot{y}_{t-1}) & f_1(e_t) & \cdots \\ 1 & y_{t+1} & y_t & \dot{y}_{t+1} & e_{t+1} & f_1(y_{t+1}, \dot{y}_{t+1}) & f_1(y_t, \dot{y}_t) & f_1(e_{t+1}) & \cdots \\ 1 & y_{t+2} & y_{t+1} & \dot{y}_{t+2} & e_{t+2} & f_1(y_{t+2}, \dot{y}_{t+2}) & f_1(y_{t+1}, \dot{y}_{t+1}) & f_1(e_{t+2}) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \end{bmatrix}$$

The above formulations include only 2 inputs, displacement and velocity, however even the output internal force history can be included, but as we have some expectations for the equation of motion to be mostly displacement and velocity dependent they make the most rational sense to use. The formulated models are a classic optimization problem, no unique solution. plenty of mathematical as well as machine learning approaches exist to find resolve it.

Regression techniques

Regression techniques are fundamental for parameter estimation and model identification, particularly in nonlinear ordinary differential equations (ODEs) and structural dynamics. These methods connect observed data with governing equations, providing predictive insights into dynamic systems. Widely used approaches include Least Squares, Lasso, and Elastic Net regression, each suited to specific optimization scenarios (refer to Appendix E).

In these methods, the columns represent features or functions, and irrelevant functions are eliminated through the optimization process. Lasso regression is particularly effective due to its penalty term, which minimizes the coefficient matrix and identifies relevant features.

This optimization problem can be solved using a custom Lasso regressor, referred to from now on as ”Custom Lasso”, or with PySINDy’s sparse regression implementation of the same RX model referred to as ”PySINDy”.

Constraints

It’s always best to try to involve as much physics as possible to guide the optimization problem into a suitable solutions, those can be added in as constraints such constraints to the coefficient matrix c_i can be

1. inequality constraints: Coefficients corresponding to specific features can be constrained to be non-positive:
$$c_i \geq 0, \quad \forall i \in \text{ indices of positive terms.} \tag{1.15}$$
This is useful in domains where certain coefficients must adhere to physical or practical constraints such as having positive linear stiffness and positive linear damping.
2. Fixed Values: Specific coefficients can be fixed to predefined values:
$$c_i = v, \quad \forall i \in \text{fixed values.} \tag{1.16}$$
this however is not useful, if we have prior knowledge of a particular component of the system, e.g. linear stiffness it would be best to reformulate the model with it on the left hand side , i.e. fit for $F_{int} - k * y$ instead of F_{int}
3. bounds for parameters, if there is any knowledge what the expected range for a certain parameters, it would be best to limit them

Implicit Parameter Optimization in the Features Model

The default parameters are multipliers of features, however having a function e.g. $c_1 * cos(A.y)$ as a feature is problematic as linear regression does not accommodate multiple parameters, thus an overarching optimization is done with the linear regression within it, i.e. perturbing the initial value of A to get multiple fits of c_1 and see if a parameters like A are optimized to minimize the objective function with the quasi-newton method of Limited-memory Broyden–Fletcher–Goldfarb–Shanno with box constraints (L-BFGS-B) [33] see appendix I for an overview of such optimization algorithms.

Feature Engineering

In machine learning, feature engineering is the process of turning raw data into useful input variables (features) that improve the performance of a model. It includes adding, changing, or picking features that pick up on important patterns in the data so that the model can make good predictions.

One important part is feature creation, which means making new variables from old ones. For example, you could calculate ratios or get time-based traits (like month, day). Using statistical tests or importance scores from models, feature selection gets rid of features that aren’t needed or are repeated, making the data simpler to understand. Getting rid of missing values and reducing dimensionality also helps simplify datasets , which can improve the performance of models.

Using feature engineering correctly can make models more accurate, prevent over-fitting, and reduce computational time. A lot of literature highlights that feature selection is more critical than algorithm choice, as effective features enable even basic models to perform adequately.

Many techniques exist for feature engineering in terms of linear system, e.g. Principal component analysis and K-means Nonlinear feature engineering is only feasible when a dataset (or the physics generating it) is well understood.

2

Identification of Benchmark Non-Linear SDOF Systems

Due to the relative lack of data available for pile-driving measurements, simpler benchmark cases of nonlinear 1 DOF systems are explored. These benchmarks aim to investigate different approaches, examine the effects of hyper-parameters, and assess the impact of noise on the identification process.

Main Points of Investigation

1. Can the response be extrapolated to other types of excitation?
2. Can the different types of nonlinearities and their combinations be clearly identified?
3. How much information about the system can be extracted? Is the technique effectively a black-box approach?
4. Can a technique properly accommodate history or time variant physics?

Benchmark Case Studies

The case studies include:

1. A simple Duffing oscillator.
2. A Duffing oscillator with higher-order stabilizing terms.
3. A friction-slip model represented by a hyperbolic tangent function.
4. A 2-degree-of-freedom system with a single cubic stiffness term

Process for Benchmark Cases

System Generation

Systems are generated with arbitrary coefficients for the various components, ensuring that terms are effective compared to the excitation. Terms of negligible contribution (stabilizing or destabilizing) to the internal force are excluded as they neither significantly influence the response.

Incorporating Noise

Noise is added to the applied force to mimic imperfections in harmonic loading, set at 5% of the force peak. This level is based on the harmonic force used in the tests. Using state-space formulation and a numerical solver (Runge-Kutta 45), the system is simulated to generate displacement (y) and velocity ($\dot{y}$) series. These measurements are further corrupted with 5% noise. From velocity, acceleration ($\ddot{y}$) is derived, though noise amplification during signal differentiation makes it most evident in the acceleration (see L). the state space formulation and initial value problem solver was verified by using it on the default SDOF system ($ky + c\dot{y}$) against Duhamel's integral [11]

Restoring Force Derivation
The restoring force is computed using the relationship:

$$f(y, \dot{y}) = F_{\text{ext}} - m \cdot \ddot{y}$$

Comparing Representations
After fitting the force, each of the three identification approaches generates a representation of the restoring force. These representations are compared statistically against the ground truth (both with and without induced noise) to evaluate how well the methods capture the system's behaviour.

Validation Through New Simulations
A new simulation is performed using a forcing function not included in the training series. The three representations are tested by observing their accuracy in predicting the restoring force. Additionally, a new state-space formulation is created:

$$m \cdot \ddot{y} = F_{\text{ext}} - f_{\text{fitted}}(y, \dot{y})$$

Solving this equation provides predictions for y and $\dot{y}$, which are compared against the validation case. Acceleration validation ($\ddot{y}$), while possible, adds little value as it is effectively another way of comparing forces ($F_{\text{ext}} - f_{\text{fitted}}$ vs. $F_{\text{ext}} - f_{\text{actual}}$).

2.1. Benchmark Cases Problem Formulation
1- A generic formulation of the simple benchmark systems is:

$$m\ddot{y} + f(y, \dot{y}) = F(t), \tag{2.1}$$

where:

- m is the mass,
- $\ddot{y}$ is the acceleration,
- $\dot{y}$ is the velocity,
- $f(y, \dot{y})$ represents the nonlinear internal forces dependent on displacement y and velocity $\dot{y}$
- $F(t)$ is the external forcing function,

the state-space formulation converts this second-order ODE into a system of first-order differential equations.

Oscillator Type	**Equation of Motion**
Default SDOF System	$m\ddot{y} + \alpha y + \delta\dot{y} = F_{\text{ext}}$
Default Duffing Oscillator	$m\ddot{y} + \alpha y + \beta y^3 + \delta\dot{y} = F_{\text{ext}}$
Destabilising Duffing Oscillator	$m\ddot{y} + \alpha y - \beta y^3 + \delta\dot{y} = F_{\text{ext}}$
Modified Duffing	$m\ddot{y} + \alpha y + \beta y^3 + \delta\dot{y} + \epsilon_1 y^5 + \epsilon_2 \cdot 10^3 y^3 \dot{y}^3 = F_{\text{ext}}$
SDOF with Yield	$m\ddot{y} + \delta\dot{y} + \epsilon_1 \tanh(\epsilon_2 y) = F_{\text{ext}}$
SDOF with a Harsh Yield	$m\ddot{y} + \alpha \cdot \text{yield}(y) + \delta\dot{y} = F_{\text{ext}}$, where: $\text{yield}(y) = \begin{cases} \frac{\epsilon_1}{\alpha} y, & \text{if } y > 0.001, \\ y, & \text{otherwise.} \end{cases}$
Van der Pol Oscillator	$m\ddot{y} + \alpha y + \delta(1 - y^2)\dot{y} = F_{\text{ext}}$

Table 2.1: Equation of motion for various dynamic systems.

All parameters α, β, δ, ϵ_1, and ϵ_2 are positive numbers chosen arbitrarily in order to highlight some different aspects. and all of which are constants that are time invariant. although quite a number of systems and range of parameters was investigated the thesis only focuses and highlights some key concepts and observations to help guide the system identification of the pile driving test data.

The same RX formulation intended for the pile test is to be used:

$$\begin{bmatrix} F_{\text{int},t} \\ F_{\text{int},t+1} \\ F_{\text{int},t+2} \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & y_t & \dot{y}_t & f_1(y_t, \dot{y}_t) & \cdots \\ 1 & y_{t+1} & \dot{y}_{t+1} & f_1(y_{t+1}, \dot{y}_{t+1}) & \cdots \\ 1 & y_{t+2} & \dot{y}_{t+2} & f_1(y_{t+2}, \dot{y}_{t+2}) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \end{bmatrix}$$

2- Training Data to be used for fit, is the internal force, which was calculated by solving the systems to the excitation of 2 simple harmonic forces, and direct substitution of the 2 state variables into the equations of motion gives the internal force, i.e. the target of fit.

The Force used for the training data is a simple single frequency harmonic

$$F_1(t) = \gamma_1 \cos(\omega_1 t) + \text{noise}, F_2(t) = \gamma_2 \cos(\omega_2 t) + \text{noise}, \tag{2.2}$$

where:

- $\gamma_1 = 2000$, $\omega_1 = 20 \times 2\pi$ rad/s,
- $\gamma_2 = 2000$, $\omega_2 = 30 \times 2\pi$ rad/s,

the force amplitude and frequency were chosen to be similar to the pile test, but that's an arbitrary choice here.

State-Space Formulation

for most non-linear ODE's getting an analytical expression as a solution is not possible (although different approximations and linearizations exist based on how strong the nonlinearity is), but numerical solution can be used to solve nonlinear ODE's via reducing the second order equation into a set of 2 first order equations that can be easily handled by numerical solvers. refer to appendix H. the main numerical solvers used were Runge-Kutta method of order 5(4) (RK45) for the benchmark cases and test data and Backwards Differentiation formula Backward differentiation formula (BDF) for the test results.

The second-order differential equation can be rewritten in state-space form:

$$y = y, \tag{2.3}$$
$$\dot{y} = \dot{y}. \tag{2.4}$$

The state-space representation becomes:

$$\dot{y} = \dot{y}, \tag{2.5}$$
$$\ddot{y} = -\frac{1}{m}\big(f(y, \dot{y}) - F_{\text{ext}}(t)\big). \tag{2.6}$$

Here, y represents the displacement, and $\dot{y}$ the velocity. The system is fully described by the first-order equations for y and $\dot{y}$.

Initial Conditions and Time Integration

The initial conditions for the system are specified as:

$$y(0) = 0, \quad \dot{y}(0) = 0. \tag{2.7}$$

The time parameters are defined as:

- $t_0 = 0$, the initial time,
- $t_f = 5$,
- $\Delta t = 0.00125$, the time step chosen to be similar to the pile test.

the final time, start and end time were arbitrary choices, but it is noticeable that the length of test is not as important as the sampling rate, when talking about time invariant systems. it's worth noting that when identifying a system, the sampling rate matters greatly and a rule of thumb is having it cover a frequency 10-20 times the loading frequency, e.g. for a frequency of 30 Hz, that equates to 0.003 - 0.0017 seconds These parameters are used to simulate the response of the system under the combined effects of the harmonic force

Application of Noise

very simple random additive noise was added as a percent of the forcing amplitude

$$\text{noise} = 0.05 \cdot \gamma \cdot U(-1, 1), \tag{2.8}$$

where:

- γ is the amplitude of the forcing function,
- $U(-1, 1)$ is a random variable uniformly distributed in the range $[-1, 1]$, representing the noise scaling factor.

a critique of choice of noise and its importance can be found in 2.7

2. Noise in Measurements: Noise is also applied to the measured displacement and velocity to simulate real-world imperfections in observations:

$$y_{\text{measured}} = y + 0.05 \cdot \mathcal{N}(0, \sigma_y), \tag{2.9}$$
$$\dot{y}_{\text{measured}} = \dot{y} + 0.05 \cdot \mathcal{N}(0, \sigma_{\dot{y}}), \tag{2.10}$$

where:

- 5% is an arbitrary choice of noise magnitude, simple chosen
- $\mathcal{N}(0, \sigma)$ is a Gaussian random variable with mean 0 and standard deviation ,
- $\sigma_y = \text{std}(y)$ is the standard deviation of the displacement series,
- $\sigma_{\dot{y}} = \text{std}(\dot{y})$ is the standard deviation of the velocity series.

2. Validation Force: The validation force is defined as a piecewise function:

$$F_{\text{ext}}(t) = \begin{cases} 0, & t \leq 2.2, \\ \gamma_3 \big[\sin(\omega t) + \sin(2\omega t) + \sin(3\omega t) \big], & t > 2.2, \end{cases} \tag{2.11}$$

where:

- $\gamma_3 = 75$,
- $\omega = 30 \times 2\pi$ rad/s.

Figure 2.1: Benchmark cases: chosen validation force

No noise was applied to the validation data to ensure that the comparison is clear, and seeing that noise was not modelled it will not be a clear comparison starting from the Custom RX or PySINDy results without noise, the residual can be used to model noise as well.

Both the actual equation of motion and the identified systems are simulated and comparisons are done there for various aspects. note: simulations with the identified systems were done using the start of the trimmed actual response as initial conditions, thus ensuring that the transient state at the start e.g. jumping from (0,0) does not cloud our comparison by a phase delay or anything similar.

Referring to 1 for the literature review the workflow of the 3 methods of system identification can be summarised as follows:

2.2. Workflow of the RFS Approach

1. **Normalize the coordinates.** Transform y and $\dot{y}$ to normalized coordinates $\zeta(y)$ and $\zeta(\dot{y})$ using scaling transformations.
2. **Fit a surface.** Apply interpolation or regression techniques (MLP was the method of choice for the benchmark cases) to interpolate points along the restoring force surface.
3. **Approximate the Surface with Chebyshev polynomials.** Use Chebyshev polynomials to represent the restoring force $f(y, \dot{y})$, and calculate the coefficients for each polynomial C_{ij} through numerical integration or regression techniques.
4. **Reconstruct the force surface.** Combine the coefficients C_{ij} and Chebyshev polynomials to reconstruct $f(y, \dot{y})$.
5. **Validation.** Validate the internal force by using the reconstructed internal force in a new state space formulation with the validation force.

Workflow of the Parametric Model Approach (RX Model)

1. **Data Collection.** Collect time-series data of the system's state variables.
2. **Library Construction.** Generate a library of potential functions that describe the system's behavior.
3. **Regression.** Apply regression techniques to select the most relevant terms that describe the dynamics while ignoring unnecessary ones.
4. **Model Validation.** Compare the identified equations with the observed data to ensure accuracy and consistency.

All of this uses the PySINDy Python library, where choices can be made regarding the regressor, method, and combination of potential functions.

Judging Quality of the Identified System

- When the main interest is the magnitude of forcing, e.g. I believe matching the magnitude and decay if any would be a key point for pile driving, thus the Root mean square error (RMSE) would be the preferred statistic
- however for other nonlinear systems, e.g. offshore wind turbine, where fatigue and number of cycles i critical R^2 would be preferable, an oversimplified explanation is that RMSE reflects the error in amplitude, while R^2 reflects the error in shape, including the location and magnitude of peaks, but realistically they both have the same.

2.3. Case 1

A standard duffing oscillator **Ground Truth:**

$$\boxed{f(t) - m\ddot{y} = 1.0 \cdot 10^6 y^3 + 2.0 \cdot 10^4 y + 10.0\dot{y}}$$

RFS:

$$\begin{aligned} f(t) - m\ddot{y} = & 88.0y^4\dot{y}^4 + 11.99y^4\dot{y}^3 - 48.06y^4\dot{y}^2 - 1.426y^4\dot{y} + 382.9y^4 \\ & + 85.17y^3\dot{y}^4 + 4.28y^3\dot{y}^3 - 42.14y^3\dot{y}^2 - 172.0y^3\dot{y} - 5578.0y^3 \\ & - 135.4y^2\dot{y}^4 - 35.17y^2\dot{y}^3 + 111.8y^2\dot{y}^2 + 114.1y^2\dot{y} - 240.9y^2 \\ & - 188.9y\dot{y}^4 - 4.608y\dot{y}^3 + 214.6y\dot{y}^2 + 63.0y\dot{y} - 6107.0y \\ & + 147.9\dot{y}^4 - 47.4\dot{y}^3 - 212.5\dot{y}^2 - 221.7\dot{y} + 534.1 \end{aligned}$$

Custom RX:

$$\begin{aligned} f(t) - m\ddot{y} = & -3.891 \cdot 10^6 y^5 - 1.981 \cdot 10^4 y^4\dot{y} - 262.7y^3\dot{y}^2 \\ & + 1.039 \cdot 10^6 y^3 + 12.45y\dot{y}^2 + 2.059 \cdot 10^4 y + 9.123\dot{y} \end{aligned}$$

pySINDy:

$$f(t) - m\ddot{y} = -2.409 \cdot 10^4 y^4 \dot{y} - 260.4 y^3 \dot{y}^2 + 8.949 \cdot 10^5 y^3 + 448.8 y^2 \dot{y} + 15.13 y \dot{y}^2 + 2.0 \cdot 10^4 y + 7.844 \dot{y}$$

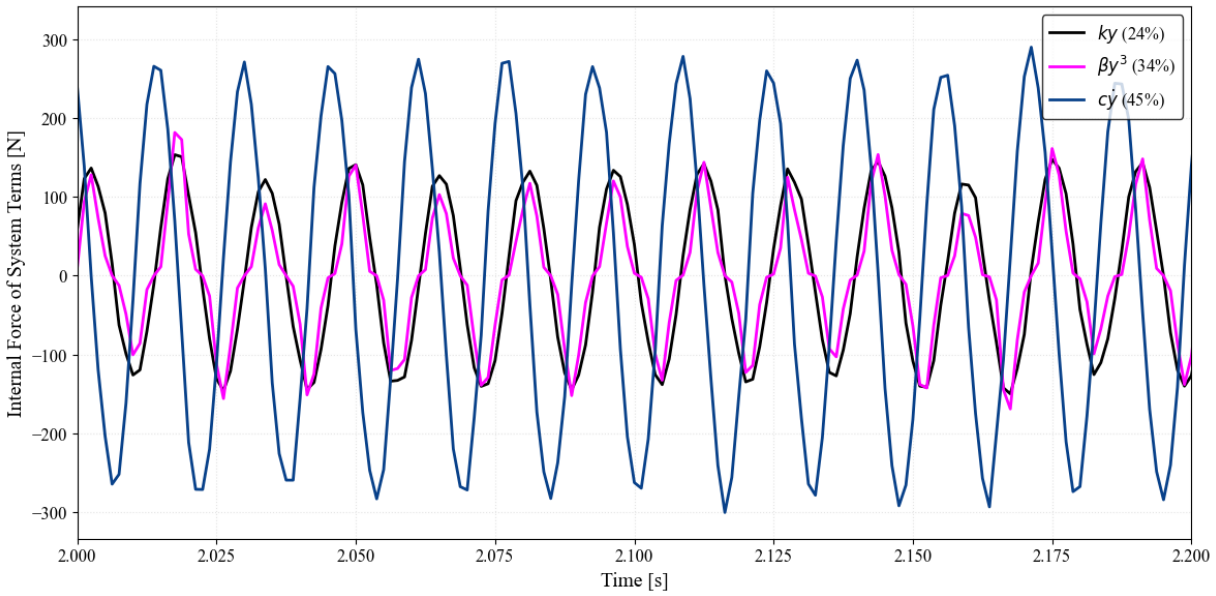


Figure 2.2: Case 1: Contribution of system terms to overall internal force as percentage of external forcing

The system parameters were selected so that both linear and cubic stiffness terms represent a significant proportion of response to the harmonic force applied, if the testing used does not excite a response from all sources of force in the system it will not likely show up in the identified system, thus test design is crucial in identifying the underlying physics of any nonlinear system this is discussed more in 2.7

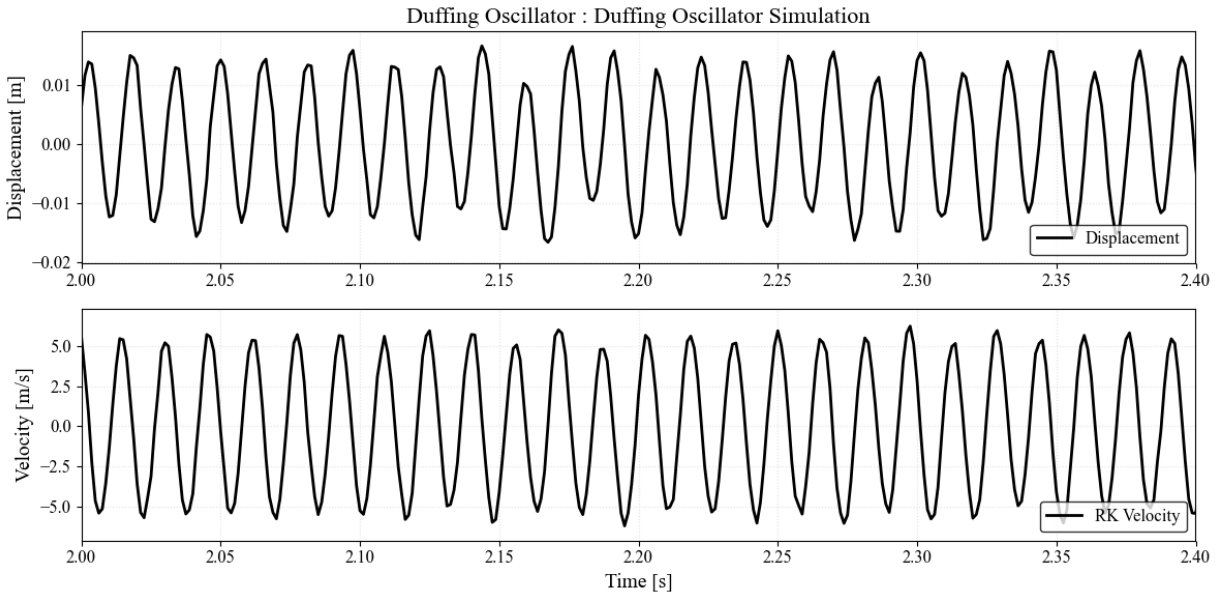


Figure 2.3: Case 1: Simulated displacement and velocity

Despite the noise, the harmonics of the response are quite clear. Determining whether the chosen

noise level is sufficient may be subjective. However, attempting to identify a system that is excessively noisy, after filtering out undesired frequencies with a low- or high-pass filter, is overly ambitious.

Additionally, the simulated displacement and velocity data were trimmed to ensure that transient responses did not affect system identification. Trials including the transient start proved highly unreliable. For this specific system, the response was trimmed from 2 seconds onward. The trimming time can vary greatly depending on the parameters and nature of the non-linear system, but selecting the point where a steady state is achieved is essential.

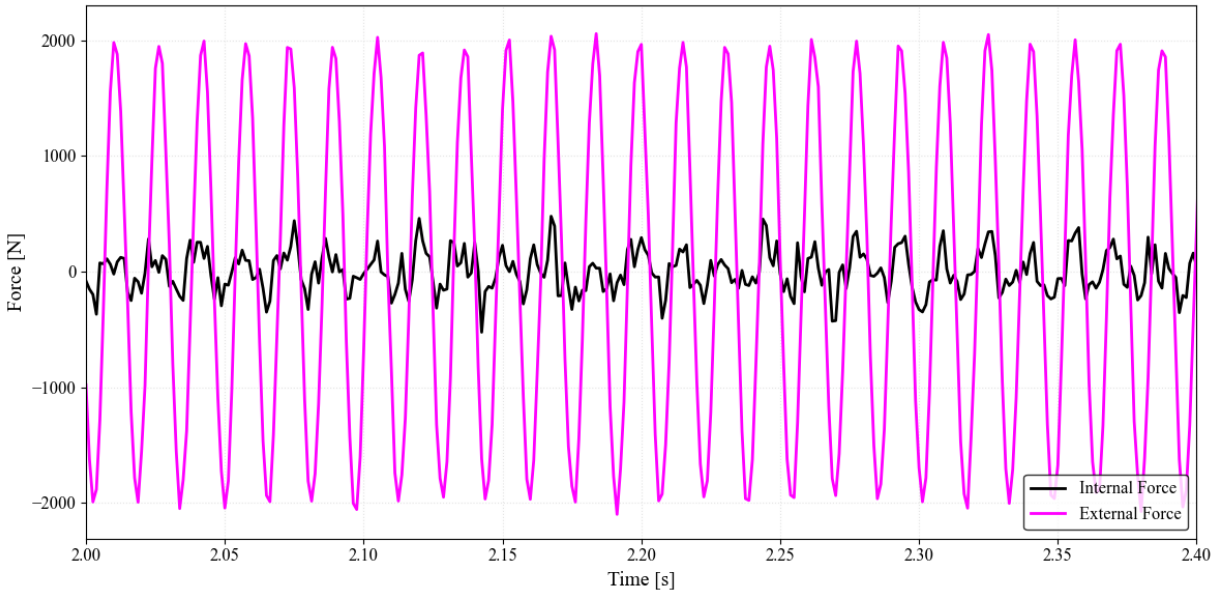


Figure 2.4: Case 1: External force and calculated internal force in a simulation

Acceleration was derived from velocity, and the mass inertia was subtracted from the external force to compute the restoring force as described in 1.8.

Restoring Force Surface (RFS)

For the RFS, the simulation data (y, $\dot{y}$) was normalized to the range $[-1, 1]$, and this normalization is taken into account when getting the coefficients for the polynomials. multiple machine learning approaches were used to fit a response surface that represents the correlation between the restoring force and the velocity/displacement data points, visualized as 3D points. This process follows the method discussed in mathematical framework (1).

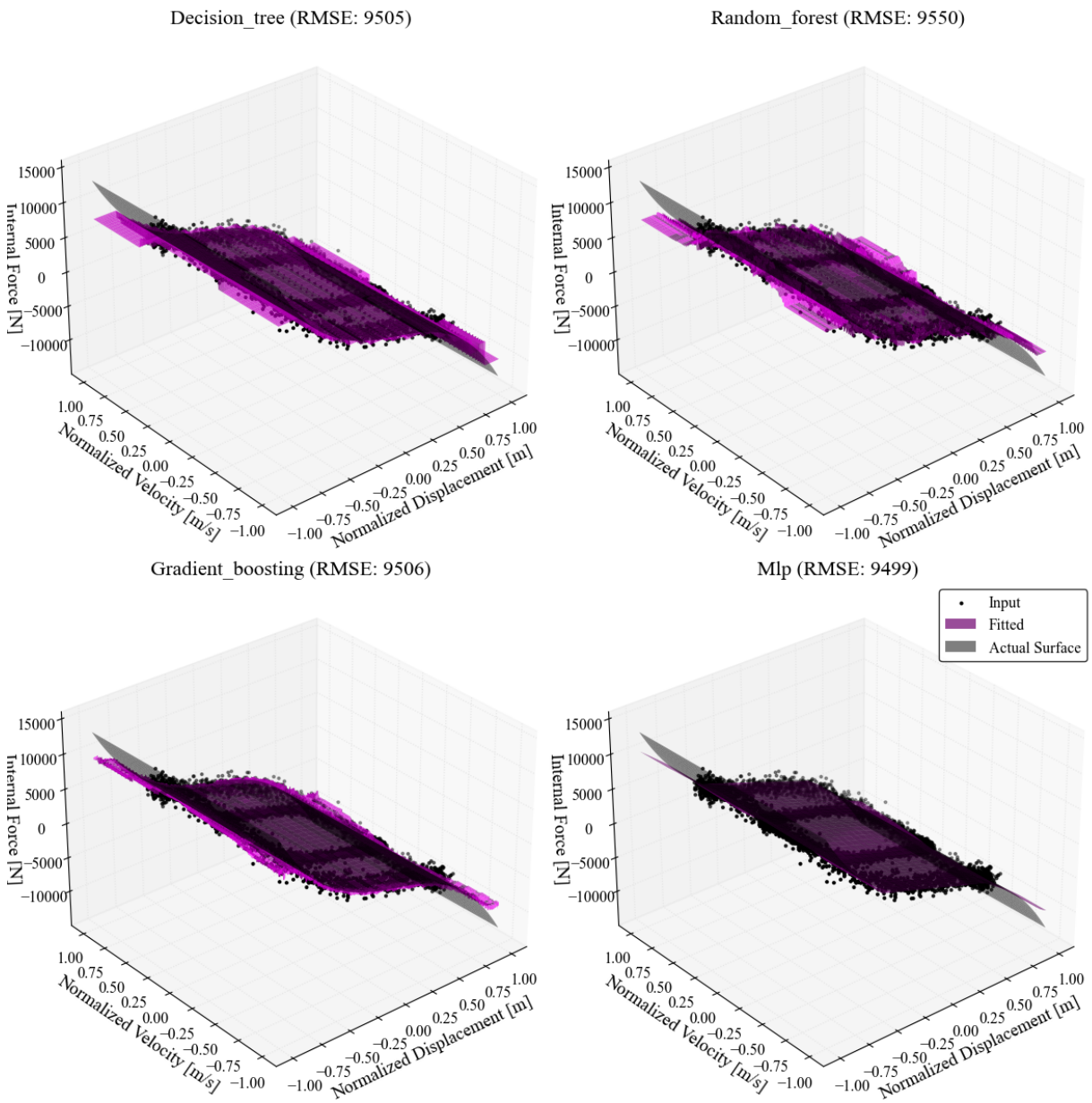


Figure 2.5: Case 1: Restoring Force Surfaces via multiple machine learning pipelines

Prior knowledge greatly aids in determining which approach yields the best fit. However, relying solely on the mean squared error (MSE) for judgment is appropriate. Any abnormalities in the surface, such as sudden jumps, should be questioned unless supported by intermediary data. Fortunately, this does not appear to be the case here. Other criteria for comparison, such as using a validation set beyond the peak velocity or displacement, are possible but are instead directly investigated in the verification case.

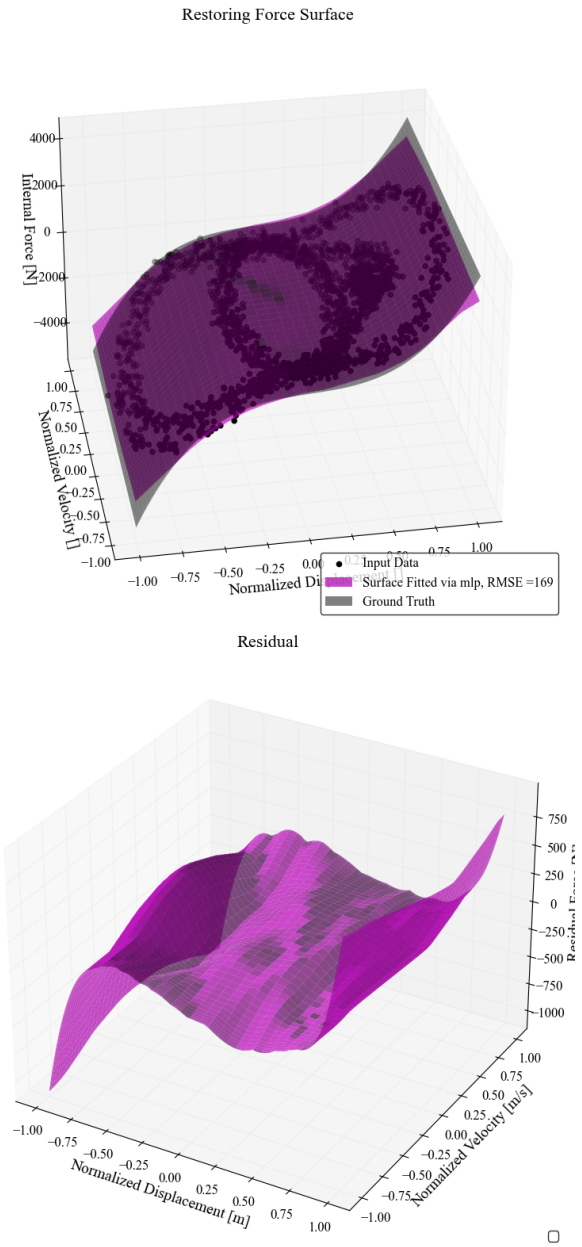


Figure 2.6: Case 1: Selected restoring force surface (RFS)

The neural network MLP appears to provide the best results overall. However, when examining the residuals, it performs poorly at the edges of the surface due to insufficient data in those regions. Additionally, extrapolation beyond the surface is unlikely to yield accurate results, similar to how overfitting an interpolated 1D function fails beyond its training range.

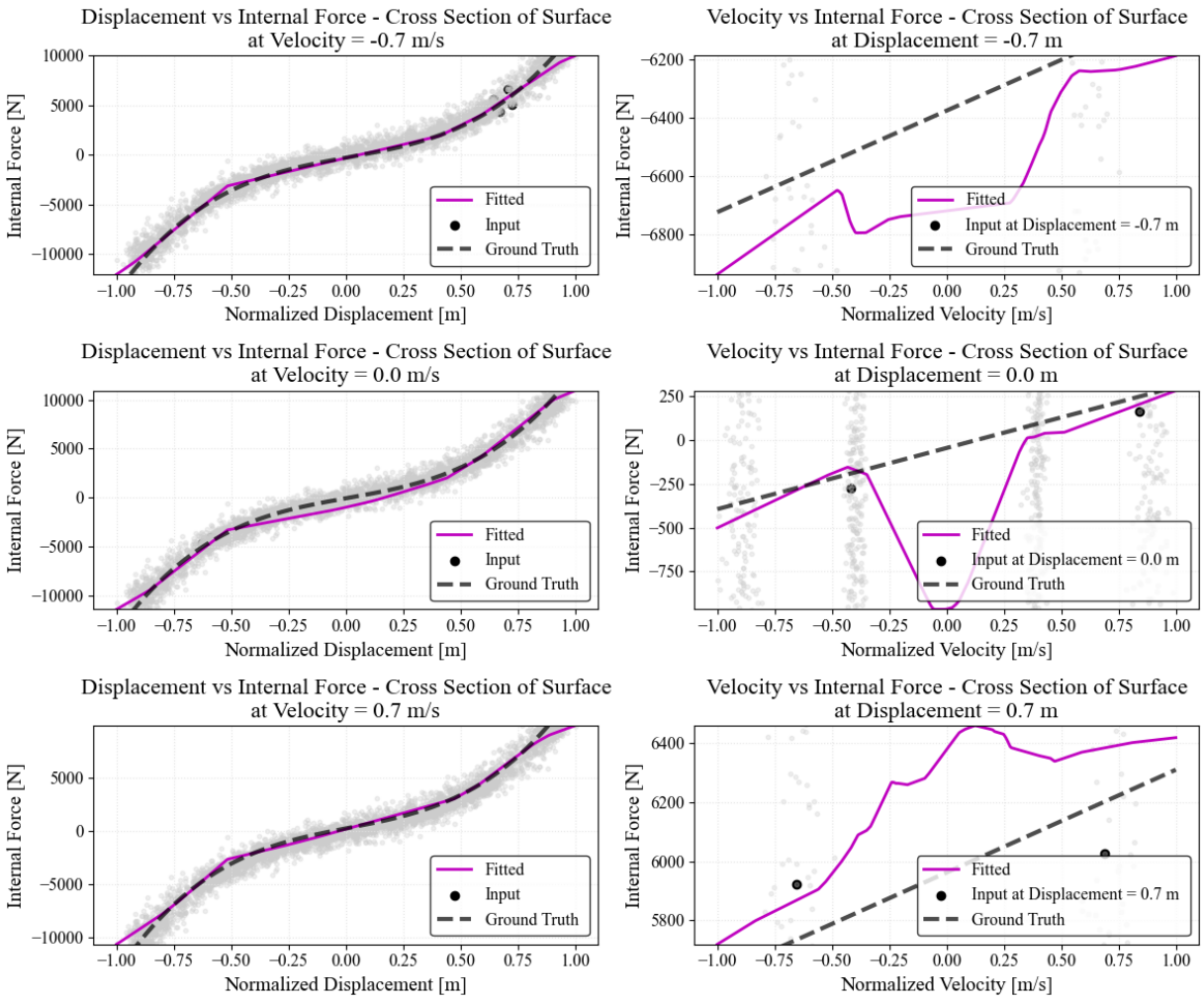


Figure 2.7: Case 1 : Cross Sections of the surface including side views of the simulated 3D dataset (internal force vs displacement and velocity)

Taking cross-sections at the fitted surface shows how the surface provides insight into the system's properties. A cross-section along displacements (showing only the input points at a given velocity) clearly indicates the cubic relationship of the system. This can also be seen from the input data itself. However, for more noisy and complex systems, it may provide a hint about what happens at specific ranges of displacements and velocities. The darker marked input points represent those that coincide with the respective section. However, depending on the training data, they may not accurately reflect the system's behaviour, as certain sections might have no data points. In such cases, the behaviour would have to be inferred from the remaining points.

A cross-section along velocity is expected to show a linear trend (representing linear damping), but this was not the case. For the given system, more points along different velocities are needed to achieve a good representation of velocity-dependent terms. The effect of this error is noticeable in the validation case. Plotting different functions of $(y, \dot{y})$ helps to evaluate how well they fit, which is beneficial for the other two approaches. Zooming out on the velocity-internal force plane (see figure 2.8), the actual linear trend is more apparent, however the error is due to the velocity term contribution being small in comparison to the applied noise (3% compared to 5% of loading f such that minute differences in it are harder to notice as they have less of an effect of the system and thus harder to detect.

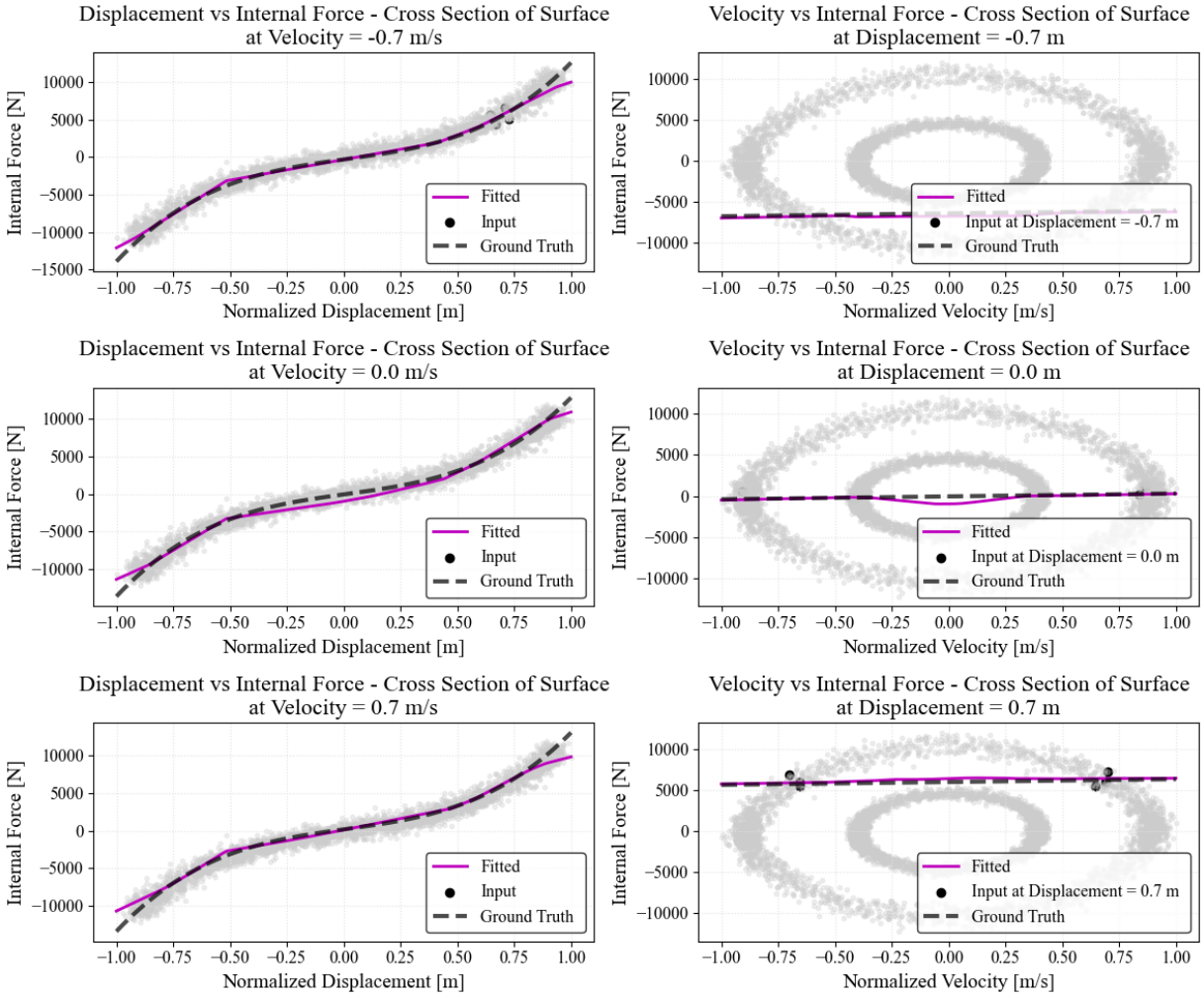


Figure 2.8: Case 1 : Cross sections of surface - Zoomed out velocity-internal force plane

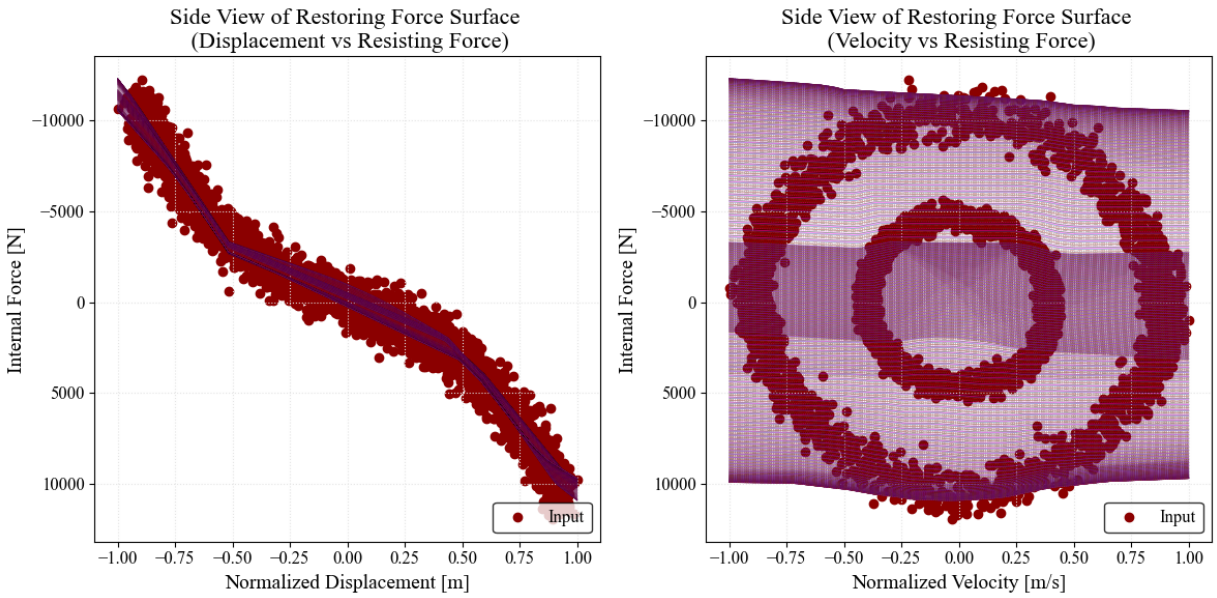


Figure 2.9: Case 1: Side view of surface and data sets

A side view of the surface shows that the cubic stiffness is quite evident in the data itself, but velocity-dependent trends are less clear as you don't have enough data points at edges of the system. it should be noted that for the selected system as the governing formula does not change when calculating the restoring force, thus extrapolating the surface is not problematic, the issue of extrapolating will occur only if the training data does not represent the entire cubic curve, e.g. for low excitation both the cubic and linear stiffness appear very similar and the cubic component will not be identified, but that would be a fault of the test but not the identification procedure.

pySINDy and Custom RX Fit

pySINDy and Custom RX is straightforward. The main control is the function library to look for. For the Duffing oscillator, simply sweeping all polynomial terms up to the fifth power seemed like a reasonable choice based on how smooth the RFS was, and polynomials are a good starting point for many systems seeing that having a linear stiffness (if only at certain ranges) is a reoccurring feature in the structural engineering field.

The functions used are:

$$y,\ y\dot{y},\ y\dot{y}^2,\ yy^2,\ yy^3,\ yy^4,\ y^2,\ y^2\dot{y},\ y^2y^2,\ y^3, y^3\dot{y},\ y^3\dot{y}^2,\ y^4,\ y^4\dot{y},\ y^5,\ \dot{y},\ \dot{y}^2,\ \dot{y}^3,\ \dot{y}^4,\ \dot{y}^5$$

If the restoring force surface derived earlier shows similarity to a particular function, it can be included, and whether it is a good choice or not will become evident. It is important to note that the choice of the function library is the most critical step in identifying the system.

From trials, going up to higher terms indefinitely proved problematic—not due to computational load but due to numerical issues. Overflow became common depending on the magnitudes used when simulating, making the system unreliable

Regarding overflow, normalization can be applied in a simple manner by normalizing the external force and coefficients using a constant value. However, minimizing individual problematic terms, such as y^7, is not feasible. The nonlinearity of the system makes reversing a change in displacement or velocity very tricky and the identified parameters would be hard to interpret.

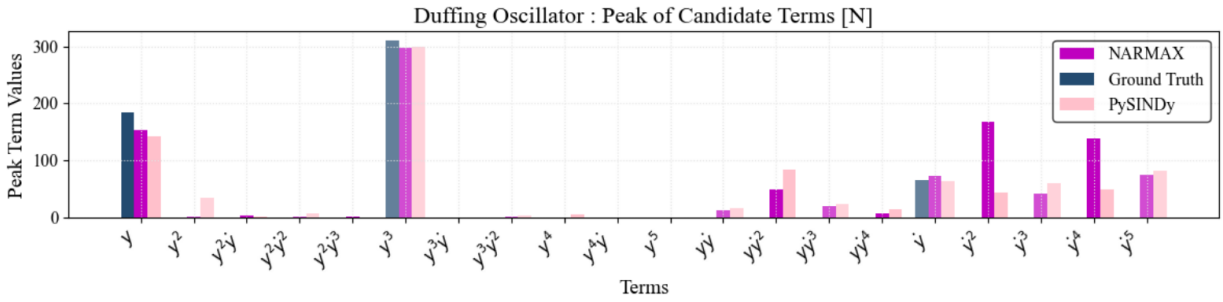


Figure 2.10: Case 1: Terms fit

It is clear that both approaches successfully identified the main linear and cubic stiffness terms with reasonable accuracy. The plot shows the absolute peak values of each feature (term) in the system in time. Since both approaches use the exact training data, these peaks correspond 1:1 to the coefficients of the system.

However, there appear to be some unnecessary terms. An initial hypothesis is that these terms either represent the added noise or counteract one another (or errors in the main terms) to minimize the objective function (i.e., the difference between the fitted and actual values), removing the noise from the system does not eliminate these terms. Setting the lookup libraries to only the system terms $y, \dot{y}, y^3$ would result in exact identification but such prior knowledge is commonly unavailable.

The statistical approach of "Mutual Information" (Mutual information (MI)) F was used to evaluate whether specific features correlate with the restoring force, i.e., whether they exhibit similarities in terms of

peaks and sign. For this case the 2 highest values of the mutual information statistic corresponds to terms actually within the system.

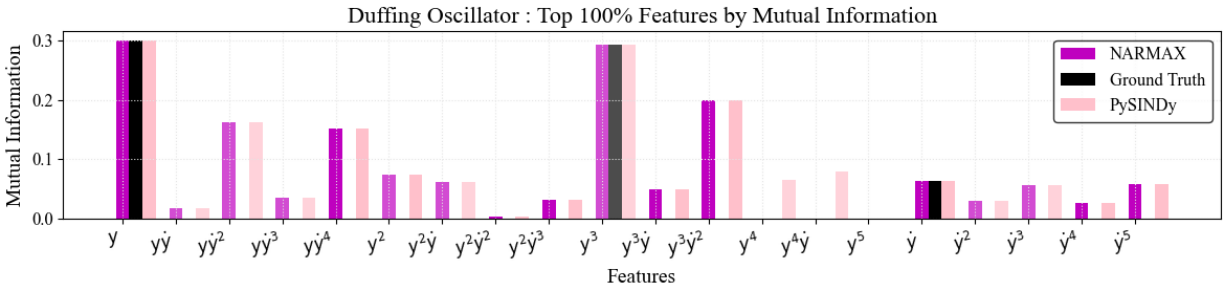


Figure 2.11: Case 1: Mutual Information statistic of the different terms

The mutual information test did not provide a clear indication of whether a feature belongs to the system or not, despite the system being relatively small. For instance, the two highest values correspond to system terms, but the third-highest does not. A slightly different approach was used by multiplying the MI values with the feature peaks to see if this helps eliminate unnecessary terms, as shown in figure 2.12 below.

Proposed Significance Index = $MI_i * max(c_i * f_i(y, \dot{y}))$

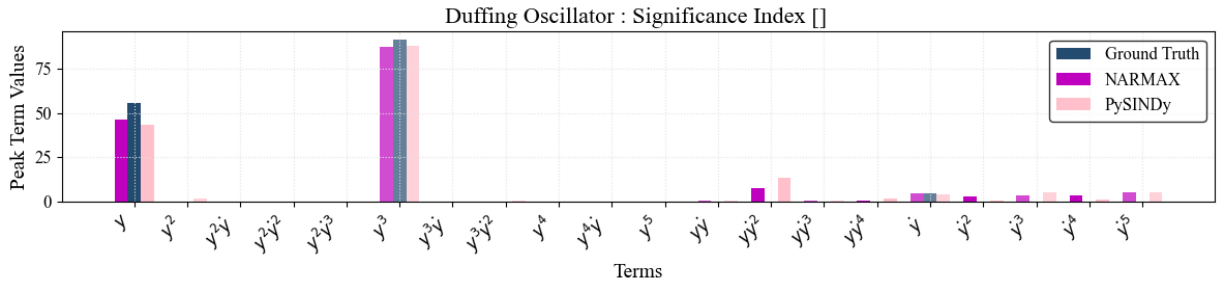


Figure 2.12: Case 1: Proposed Significance index

Unfortunately, neither the MI value nor the proposed significance index reliably identified features of the nonlinear system. One of the main terms, linear damping, did not stand out. Without prior knowledge, the term $y\dot{y}^2$ would have appeared to be a promising candidate but would ultimately be misleading.

Further more, elimination of non-relevant features was done with a combination of changing the regulation coefficients (affects the weight of the penalty term, a higher value eliminates more terms) and by inspecting how the fit changes when dropping of terms based on their magnitude, this is illustrated in figure 2.13 below

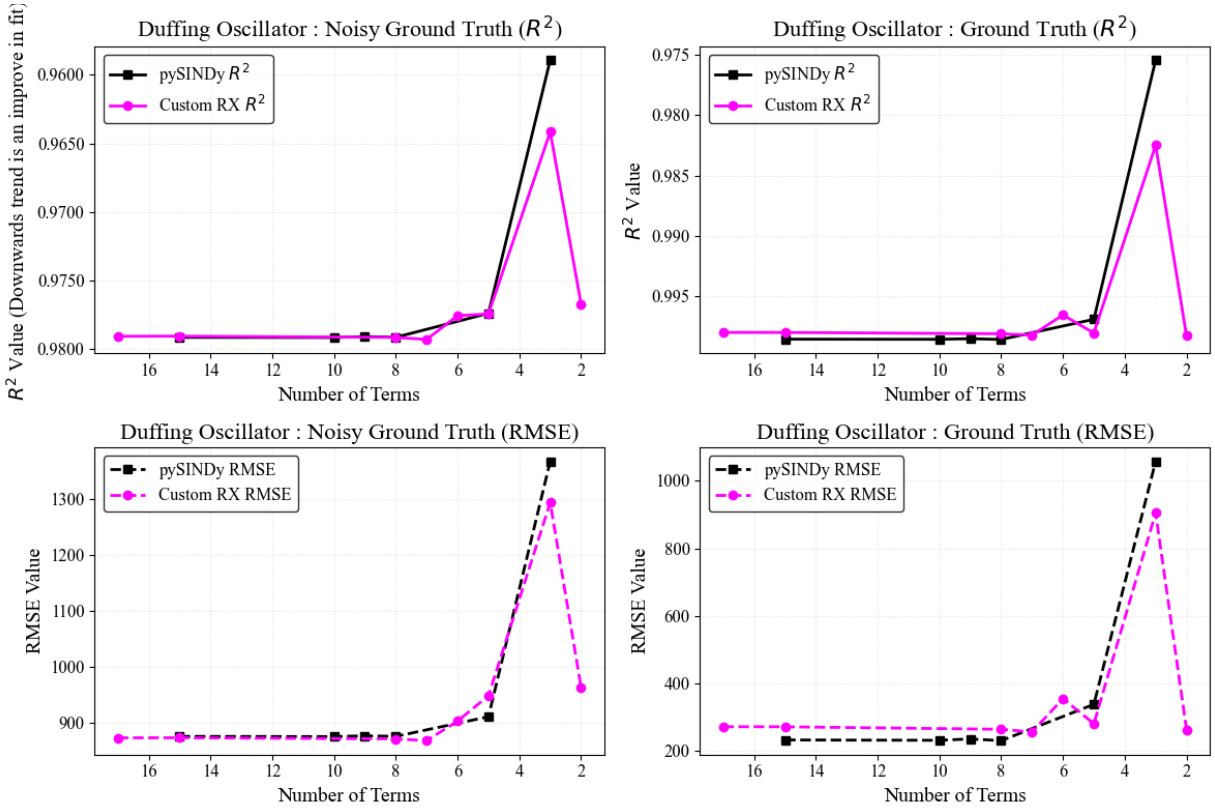


Figure 2.13: Case 1 : Removing Unnecessary Terms

Figures on the left are the statistics representing the score of the fit, the x-axis is the number of terms used (as in how many high contribution terms remain not the total number of terms used in the library) there is convergence of fit against number of terms.

The expressions above were adjusted by eliminating any terms that don't significantly affect both scores of the system (in lack of a clear trend in 2.13 above a 5% cut-off would be the next logical choice) it's also worth noting that adjusting the regularization parameter does something to a similar effect but trimming the excess terms this way can give insight, what is the most appropriate functional form for modelling damping behaviour—linear, cubic, or another formulation?

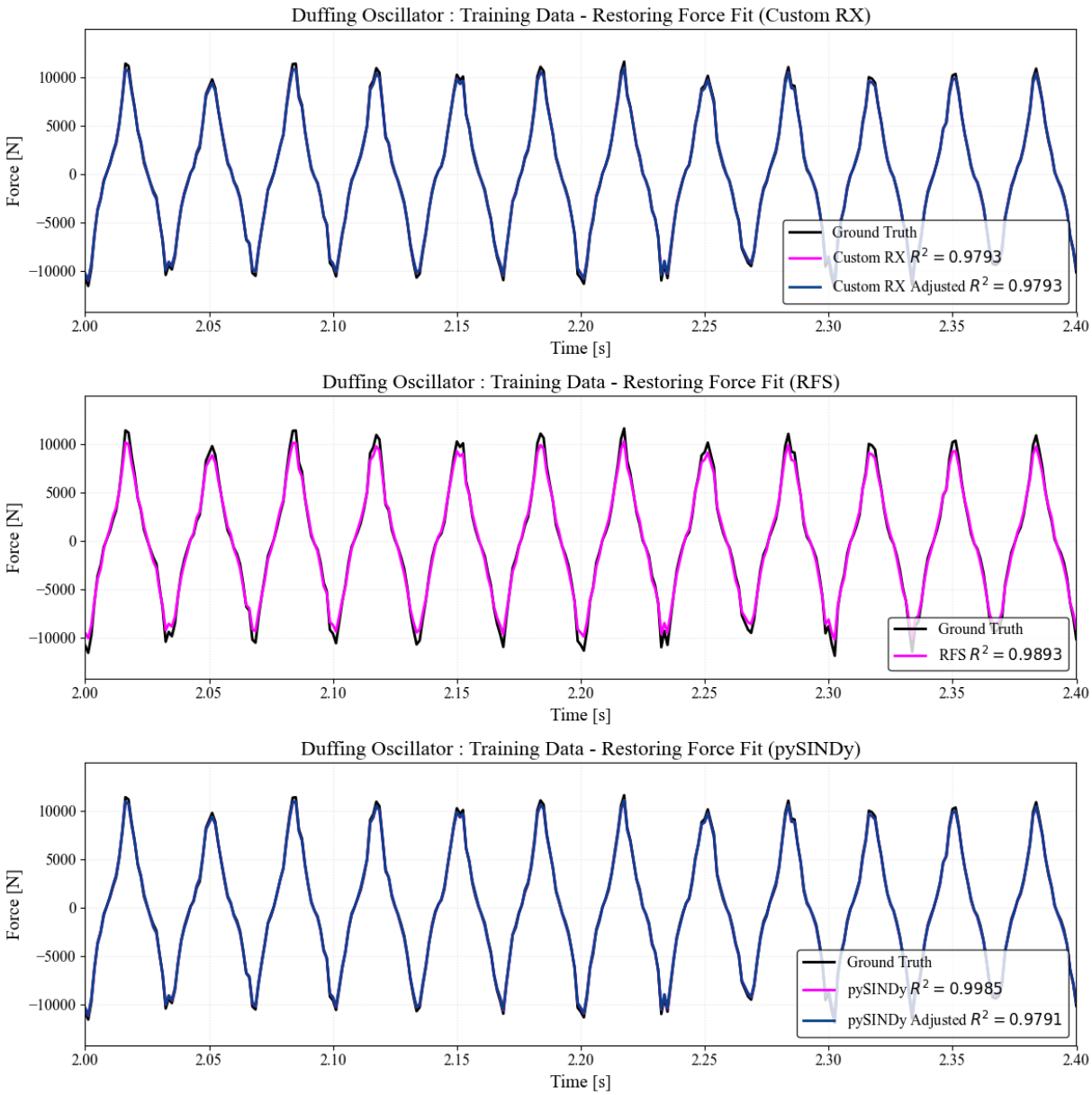


Figure 2.14: Case 1: Force fit

The plot above demonstrates that the system is able to discern the artificial noise added. All three methods show a good fit based on the R^2 value, with RFS performing the worst at 0.85. It is noticeable that trimming some terms from the ARX model actually improved the fit.

To investigate how well the approaches perform beyond the noise, residuals were plotted, and a Quantile-quantile (QQ) plot was created. As the noise is randomly distributed, the Quadratic quantum cross-correlation (QQCC) value (see Appendix' F for specifics of those statistics) provides an indication of how effectively the methods account for the noise. pySINDy appears to perform the best in this regard, as shown in Figure 2.15.

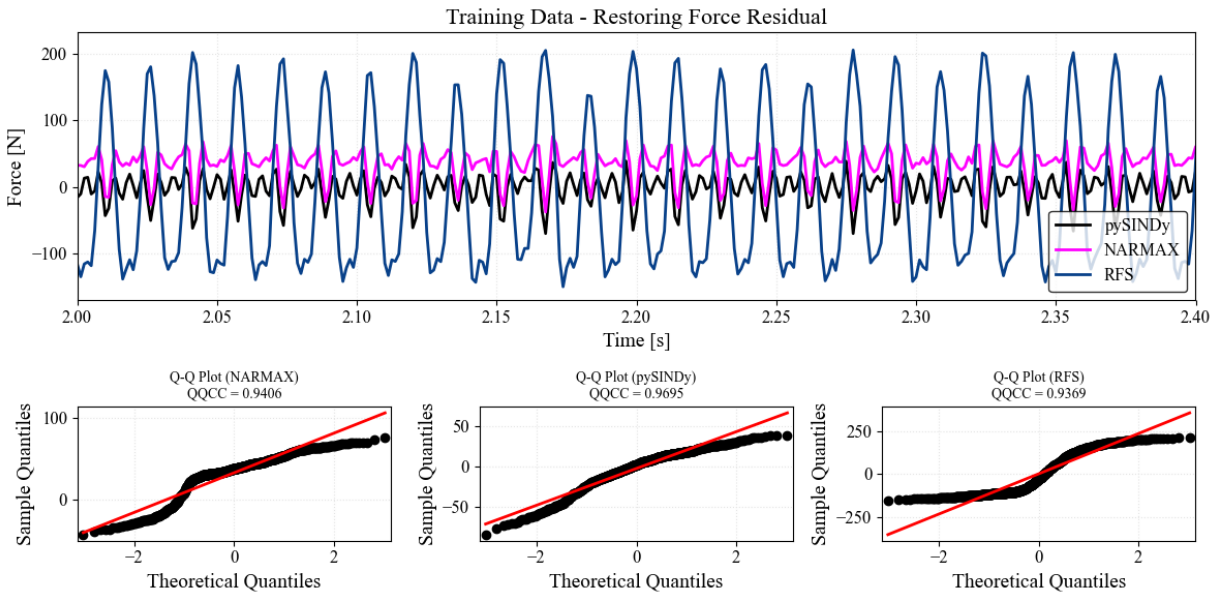
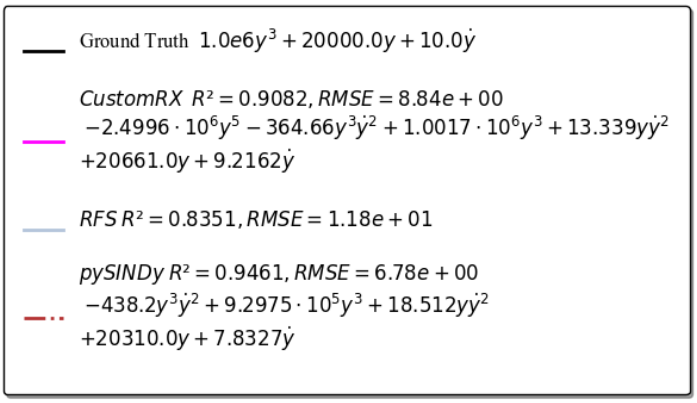


Figure 2.15: Case 1: Residual plot

Not all of the noise is accounted for. Thus, the unnecessary terms in pySINDy and Custom RX are likely a combination of representing noise and counteracting each other to minimize the difference between the fitted expression and the training data. Re-fitting the system with clean signals and noise does not eliminate these terms but results in a different combination. There appears to be no straightforward way to eliminate these terms entirely, which reinforces the importance of selecting an appropriate function library. In actual practice, we'd have no prior knowledge that the error is just additive normally distributed but would be a good first guess.

Finally, validation was performed by doing a blind simulation of the fitted systems excited by the validation force and the response was compared as seen below.



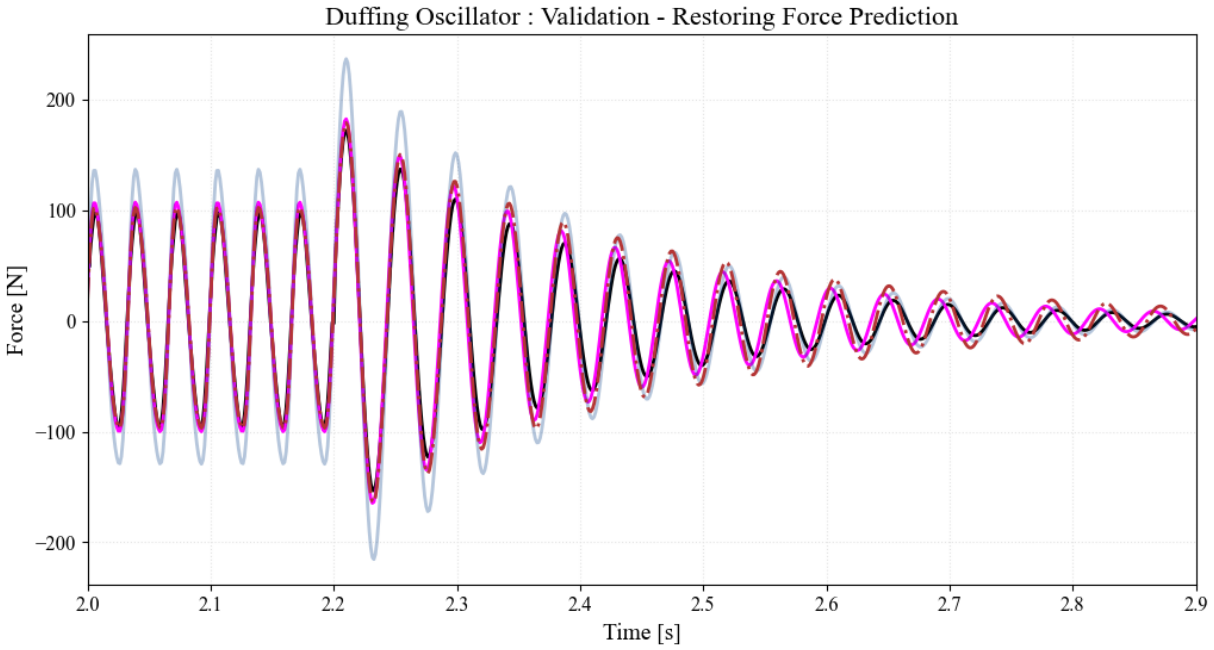


Figure 2.16: Case 1 : Validation - Restoring Force Prediction

Both Custom RX, pySINDy approaches perform well in terms of shape (R^2) and magnitude ($RMSE$) as they got the 3 main parameters correct within 20% but with additional terms especially the custom RX model. The RFS approach however does poorly in terms of the $RMSE$ but a good fit nonetheless.

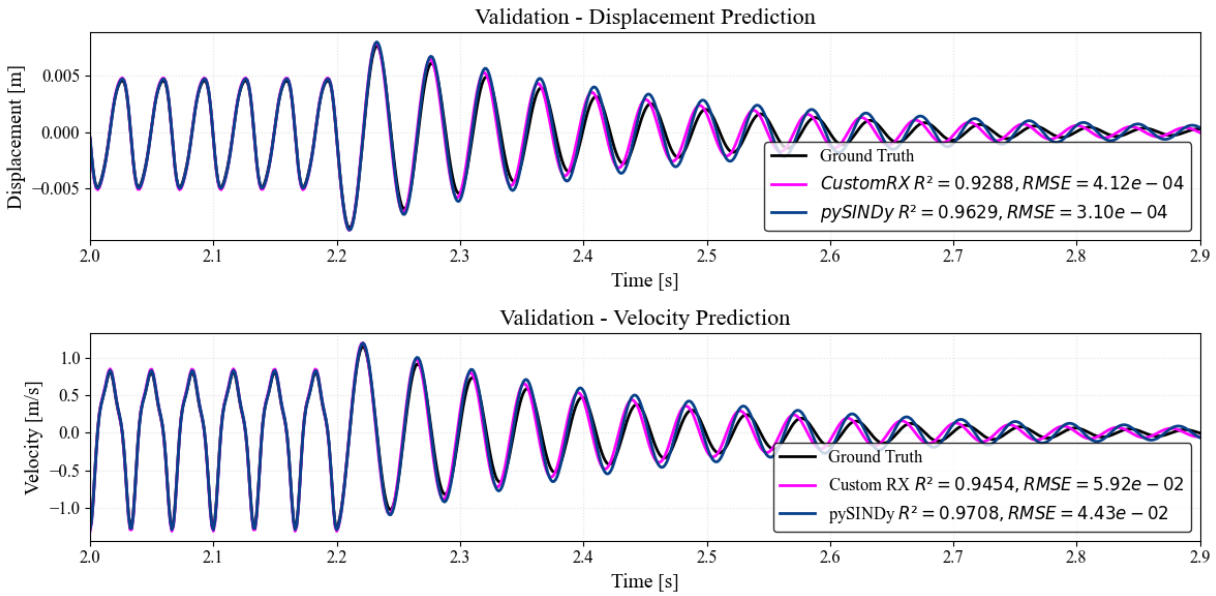


Figure 2.17: Case 1 : Validation - Displacement & Velocity Prediction

The displacement fit for the validation case demonstrates that the system's response aligns well with the applied validation force, even with the abrupt changes. This behaviour highlights the robustness of the fitted model in capturing the essential dynamics of the system under varying excitation conditions.

Finally to investigate whether the "unnecessary" terms added to the system affect simulations at different frequencies an FRF was constructed by applying harmonic forces at multiple frequencies and solving for peak amplitude (after trimming transient response) and the normalized by the force amplitude γ , this was also done for various magnitudes of loading forces. for a duffing oscillator a linear approximation to get an expression of the FRF is available and is used for comparison. see appendix B. * the approach used can not get the unstable branch of the FRF between 20-40Hz, that can be derived by adjusting the initial conditions but is not the focus of this study

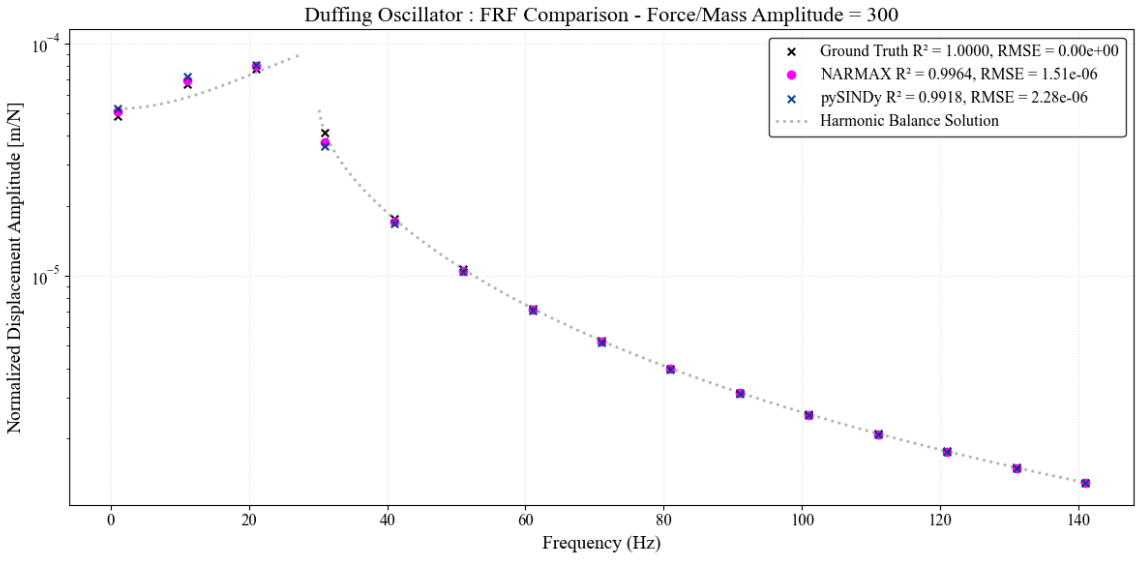


Figure 2.18: Case 1 - FRF $\gamma = 300$ from 0 - 150 Hz

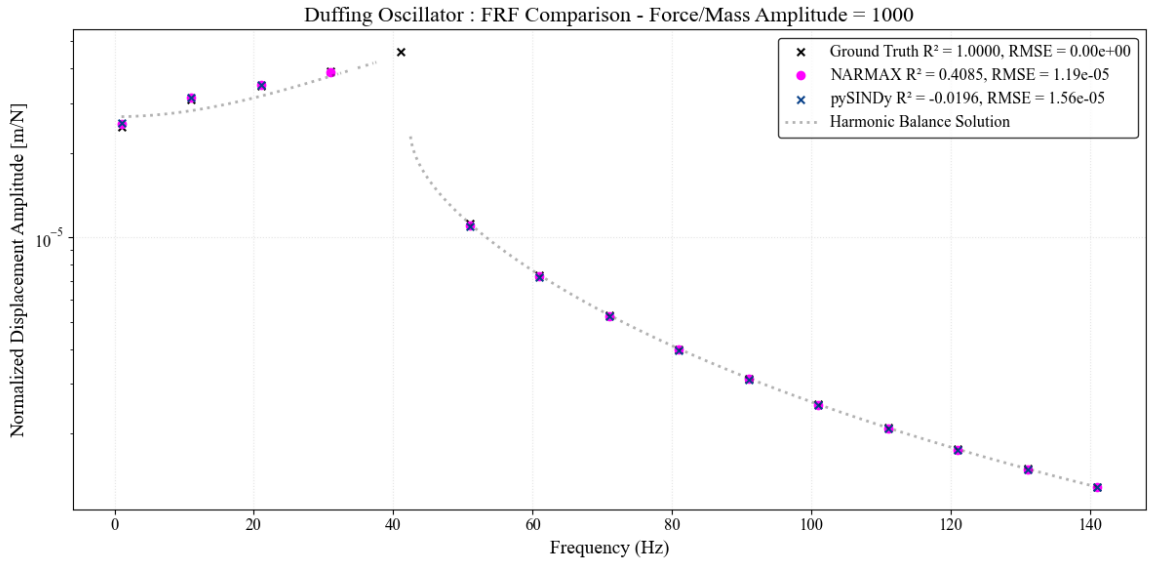


Figure 2.19: Case 1 - FRF $\gamma = 1000$ from 0 - 150 Hz

Varying the magnitude of force has no effect, and the identified systems are not sensitive to force amplitude

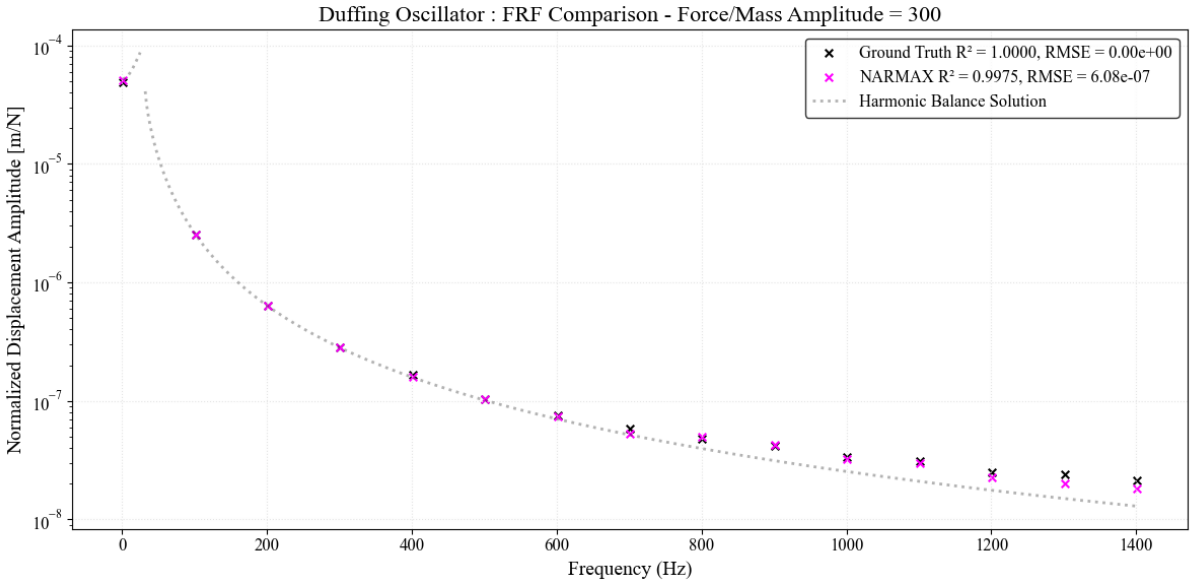


Figure 2.20: Case 1 - FRF $\gamma = 300$ from 0 - 1400 Hz

Inspecting the FRF plots, it's clear that the fit is adequate and not sensitive to excitation force nor frequency of loading. investigating numerical stability as well as ensuring we're actually capturing physics instead of over-fitting for a certain specific region. for benchmark case 2.0 this did not remain as the fit in itself was not perfect and got worse at higher frequencies. this is relevant for pile-soil interaction at hand as to correlate what frequency of testing or range of frequencies do I need in order to capture the behaviour correctly, with the lack of an actual FRF to compute nor a ground truth to compare against.

2.4. Case 2.0

Case 2 is a Duffing-like-oscillator with additional terms, as described in table 2.1

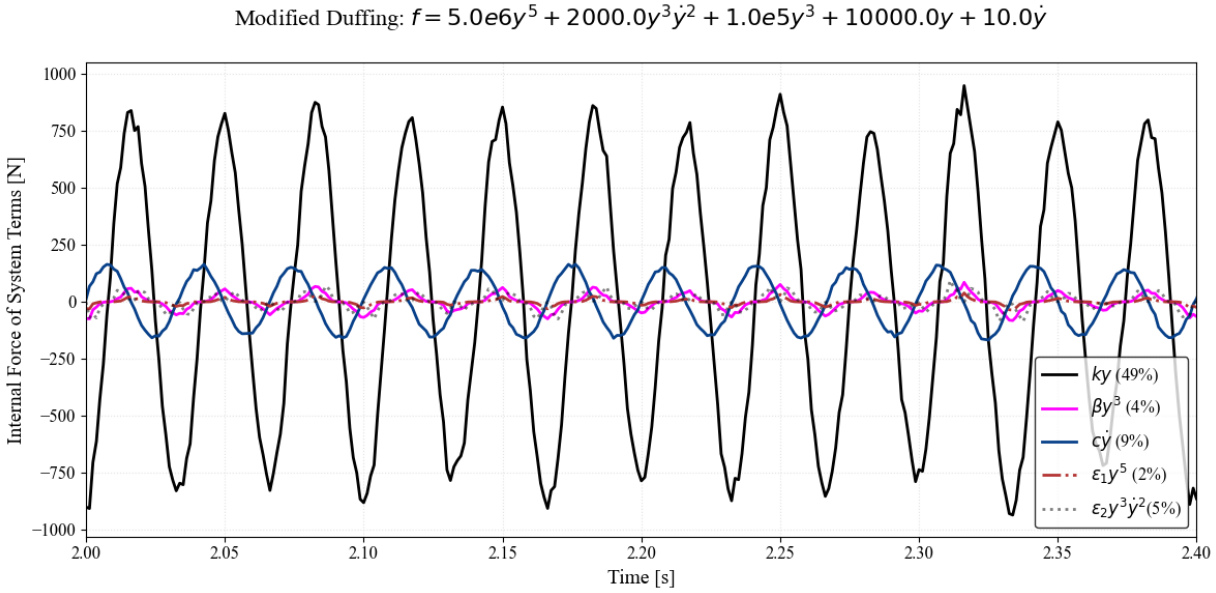


Figure 2.21: Case 2.0 Contribution of system terms to overall internal force as percentage of external forcing

The parameters of the system where chosen so that the linear stiffness contributes the most, and the higher order terms are in the order of magnitude of the applied noise to force. Afterwards the exact same procedure of Case 1 identification was followed

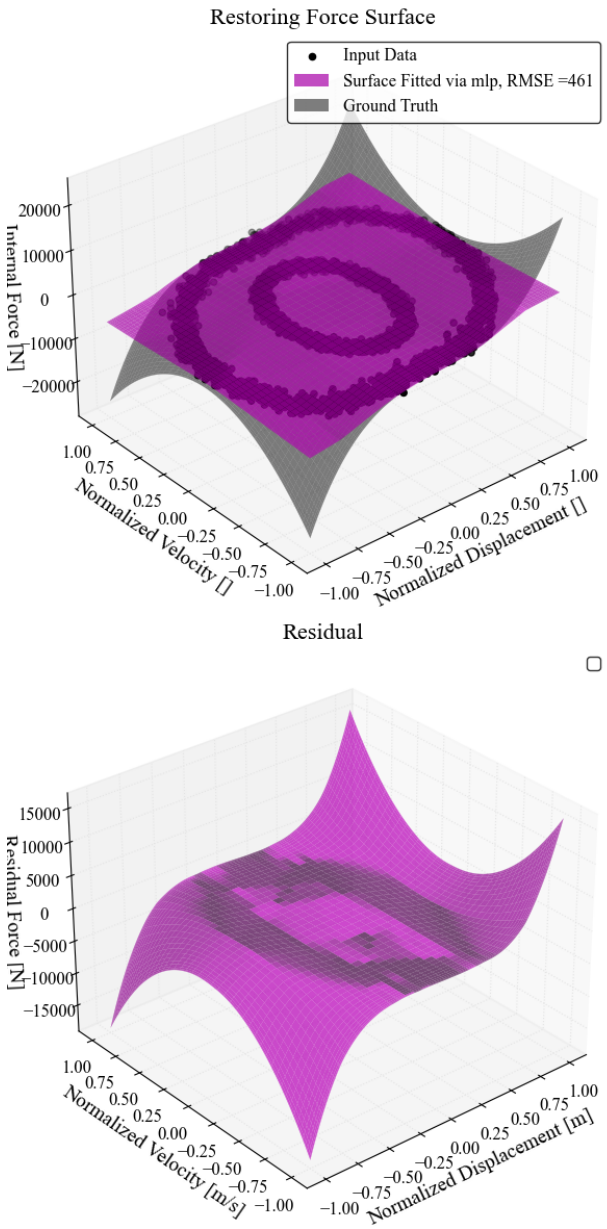


Figure 2.22: Case 2.0: Selected Restoring Force Surface

The restoring force surface shows good fit of the known region of the tests data but great residual at the edges due to lack of data points there, the excitation can be adjusted in order to capture more points there but that's a luxury not available in actual practice.

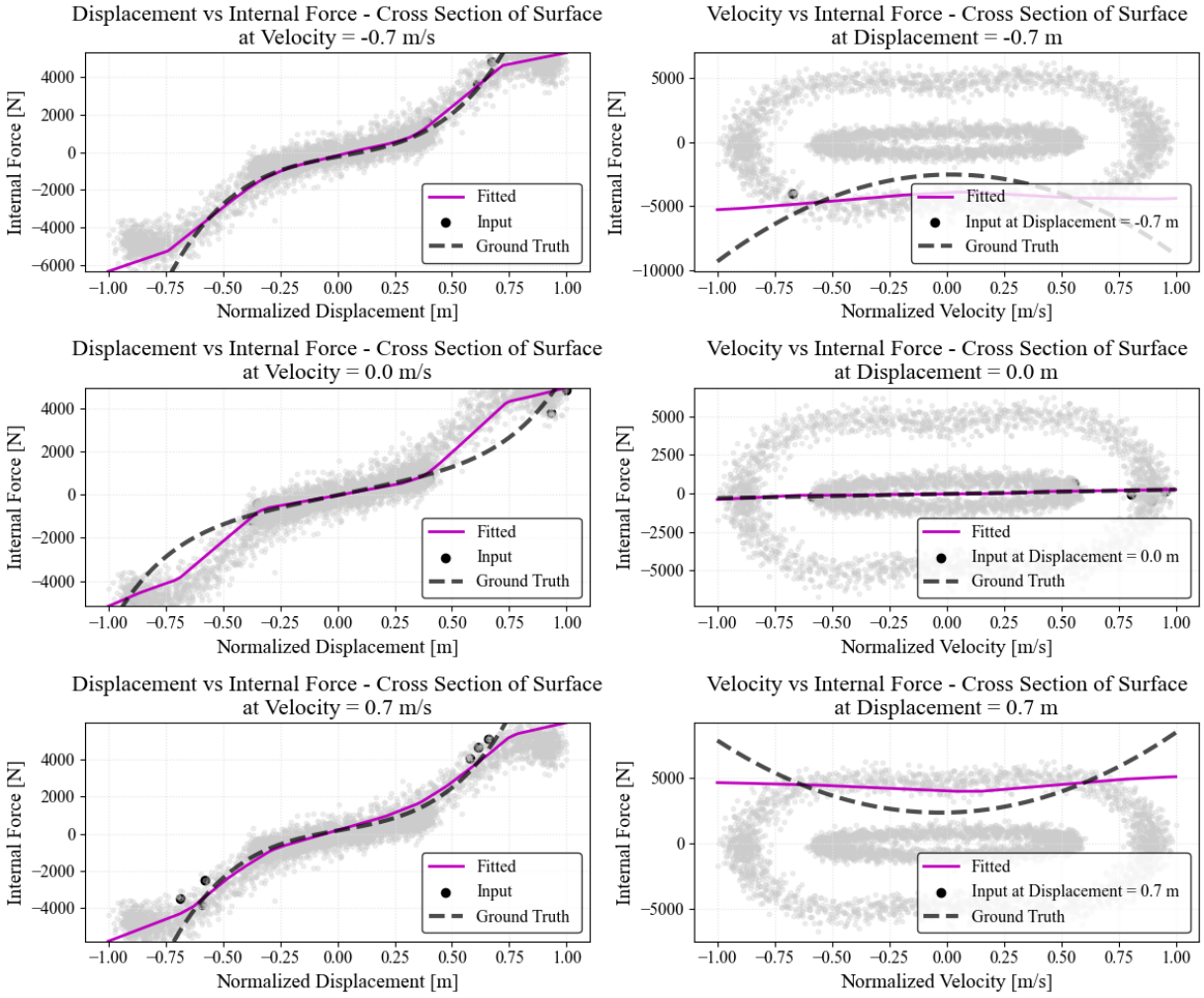


Figure 2.23: Case 2.0 :Cross Sections of the surface including side views of the simulated 3D dataset (internal force vs displacement and velocity)

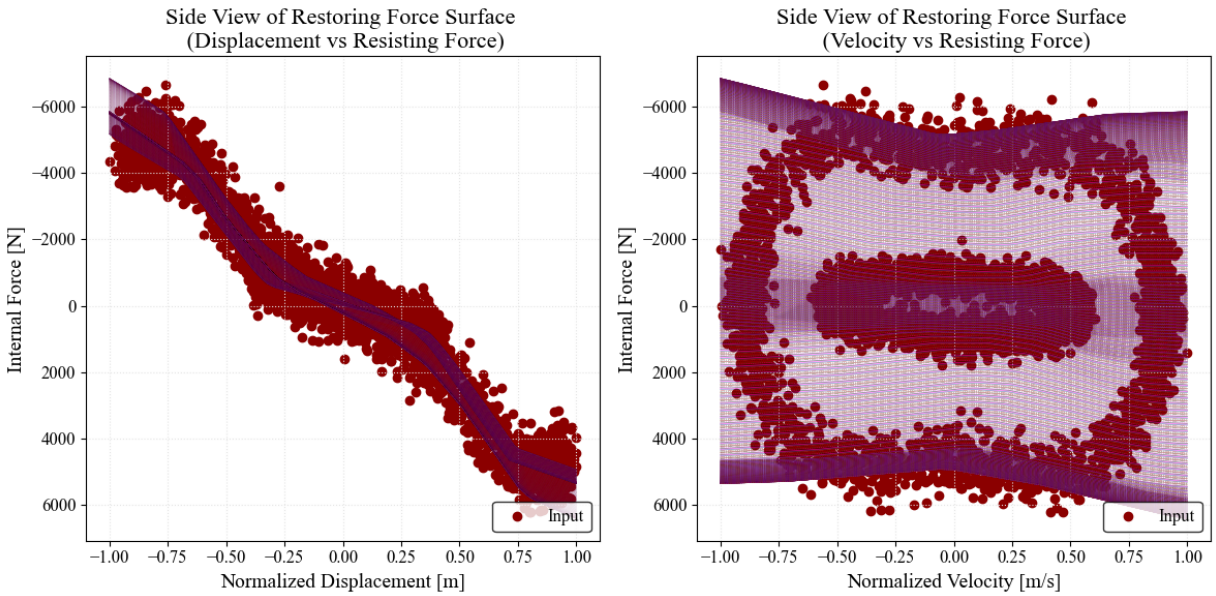


Figure 2.24: Case 2.0 : Side Views of Surface

The surface does not clearly indicate any specific functions like case 1, however seeing that it's pretty smooth (no sudden changes or jumps) a polynomial function library was used as polynomials don't have issues with such shapes.

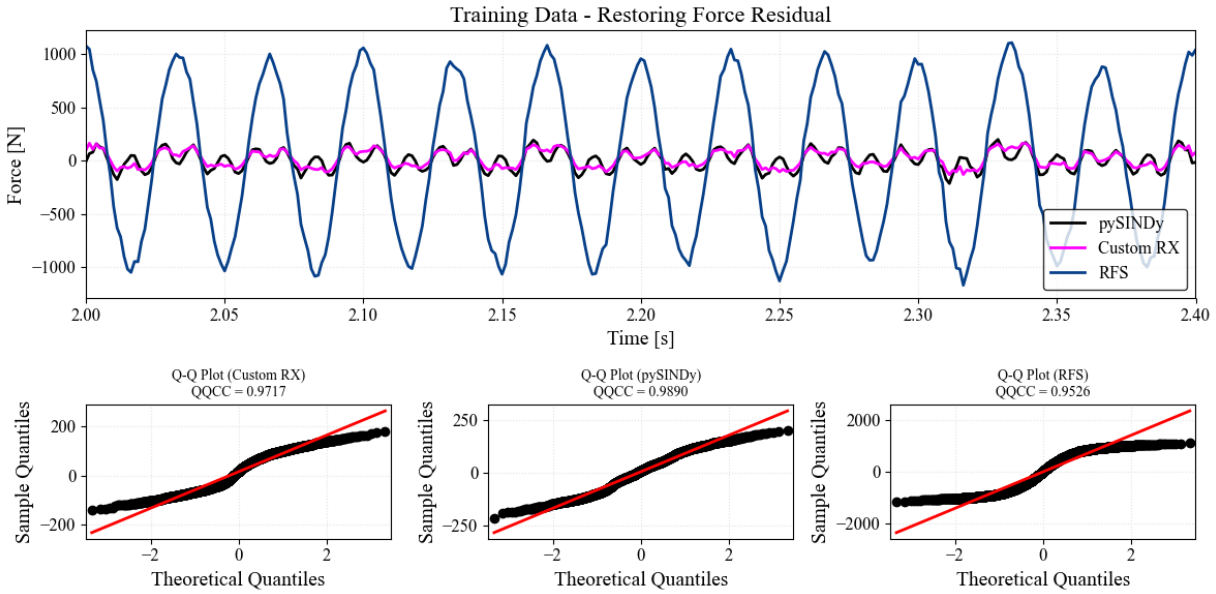
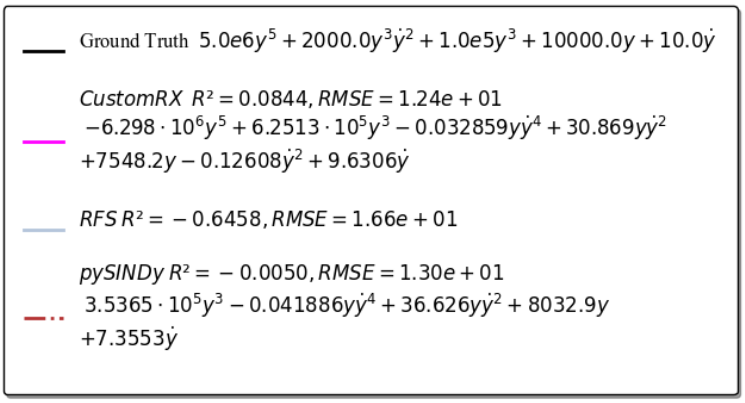


Figure 2.25: Case 2.0 : Training Data - Restoring Force Residual

The QQCC value tends to be logarithmic in scale , shape of the QQ-plot is a better indicator, all 3 fits seems to have not properly isolated the normally distributed noise.



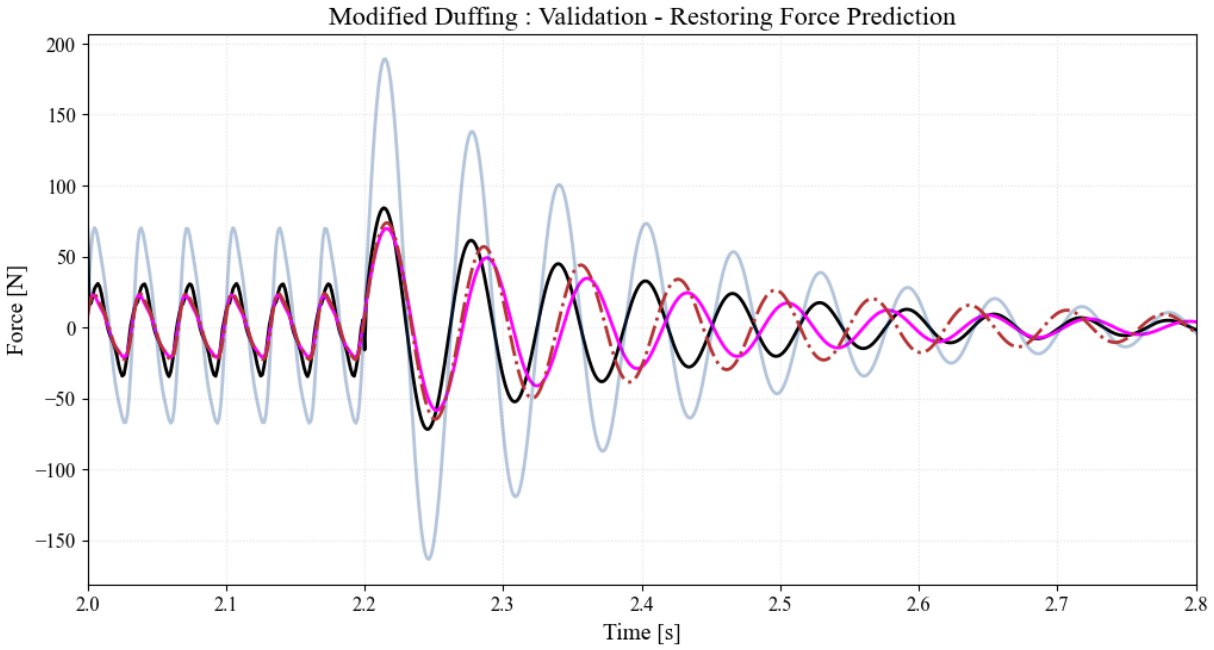


Figure 2.26: Case 2.0 : Validation - Restoring Force Prediction

When doing a blind validation, neither of the systems does a perfect fit however seeing that the magnitudes of the RMSE and rate of decay are the best for Custom RX, that would be the choice model to rely one. when observing the terms, some terms like y^5 was completely trimmed off seeing that its contribution was 2% of the total applied force and thus did not register when fitting, the rest also seem like a combination of different terms to simply mimic the response, and some destabilizing terms are there, it is possible to constrain the parameters to all be positive and stable, however that dramatically drops the accuracy of fit, a good example of why negative terms are needed are Taylor expansions, negative terms do always indicate an instability or negative response however when counteracted.

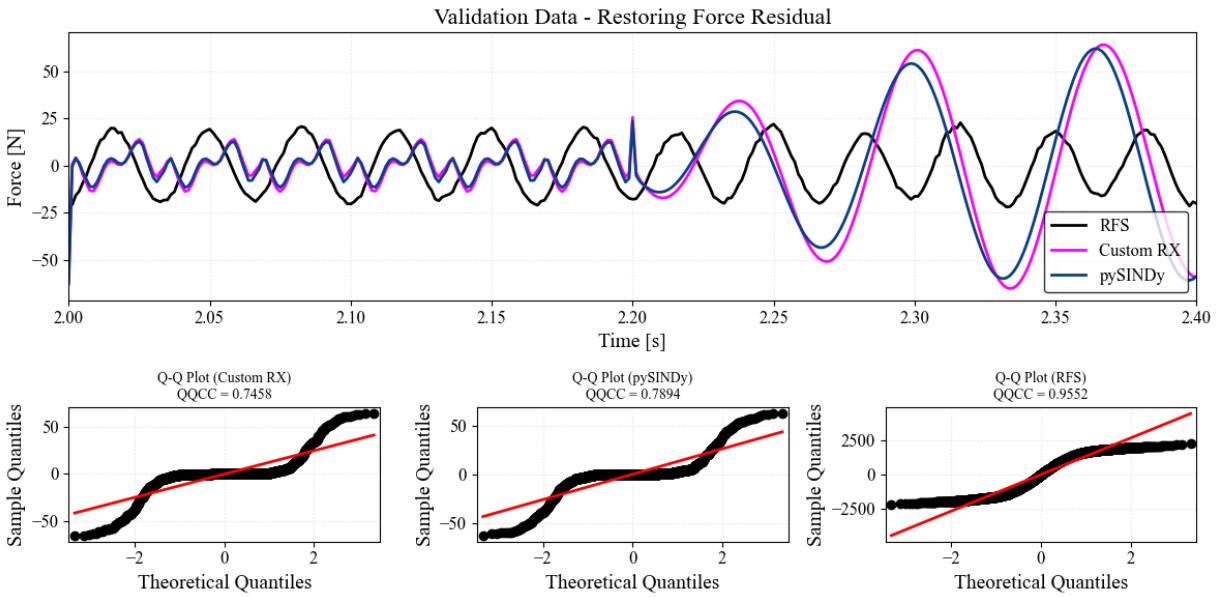


Figure 2.27: Case 2.0 : Training Data - Validation Force Residual

the residuals of the validation case clearly show that a portion of the higher order contributions were disregarded when fitting as residuals should be random not repeating harmonics.

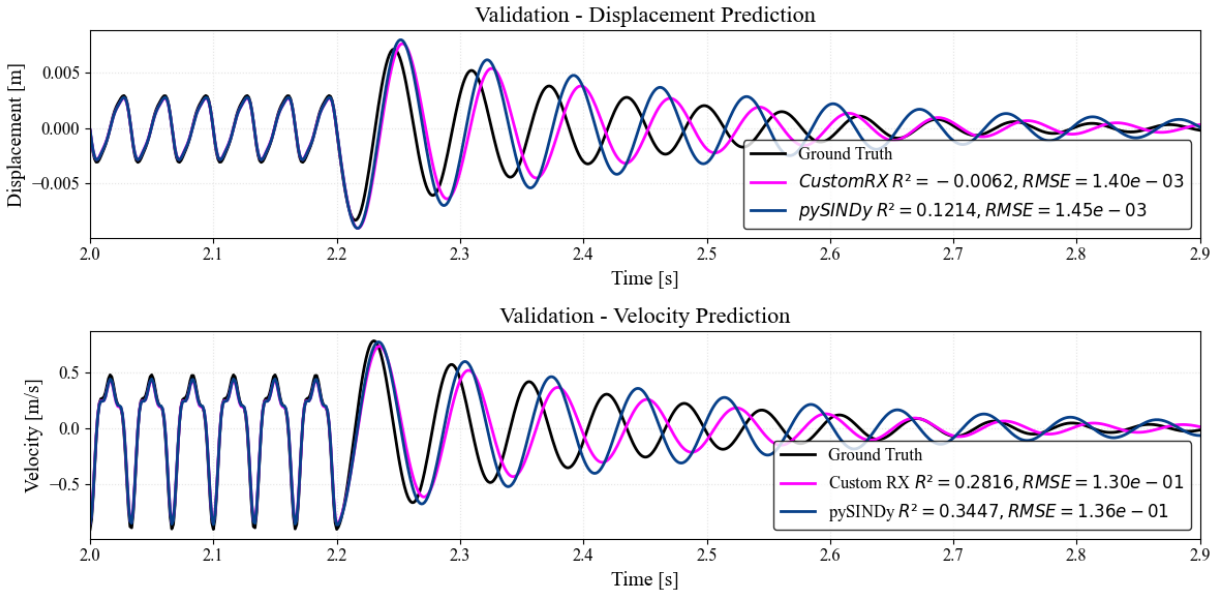


Figure 2.28: Case 2.0 : Validation - Displacement & Velocity Prediction

Despite the negative terms, the ODE solver had no issue and a harmonic response was observed, not an exponential growth or decay, linearizing the system around a point and getting the eigen values can give more insight into the stability of the system [6]. But all of that is beyond the scope and focus of this research.

Ground Truth:

$$f(t) - m\ddot{y} = 5.0 \cdot 10^6 y^5 + 2000.0 y^3 \dot{y}^2 + 1.0 \cdot 10^5 y^3 + 1.0 \cdot 10^4 y + 10.0 \dot{y}$$

RFS:

$$\begin{aligned} f(t) - m\ddot{y} = & 2.961y^4\dot{y}^4 + 7.325y^4\dot{y}^3 - 22.31y^4\dot{y}^2 - 42.58y^4\dot{y} + 606.8y^4 \\ & - 311.6y^3\dot{y}^4 - 10.97y^3\dot{y}^3 + 454.4y^3\dot{y}^2 + 58.46y^3\dot{y} - 739.4y^3 \\ & - 50.79y^2\dot{y}^4 + 1.12y^2\dot{y}^3 + 73.68y^2\dot{y}^2 + 26.46y^2\dot{y} - 477.9y^2 \\ & + 568.8y\dot{y}^4 + 26.62y\dot{y}^3 - 929.2y\dot{y}^2 - 43.04y\dot{y} - 5088.0y \\ & + 56.34\dot{y}^4 - 127.9\dot{y}^3 - 32.28\dot{y}^2 - 210.4\dot{y} - 58.26 \end{aligned}$$

RX:

$$f(t) - m\ddot{y} = -6.635 \cdot 10^6 y^5 + 6.346 \cdot 10^5 y^3 + 32.01y\dot{y}^2 + 7557.0y + 8.502\dot{y}$$

pySINDy:

$$f(t) - m\ddot{y} = 3.501 \cdot 10^5 y^3 + 38.29y\dot{y}^2 + 8019.0y + 4.86\dot{y}$$

Again the RFS expression is not as interpretable as the rest, capturing the force to an extent but not the specifics. The other two identified the linear stiffness component (within 75%) but not for the higher order stiffnesses, they produce a combination of terms (stabilising and destabilizing) to fit the rest, thus can not be distinguished but their effect is maintained to a degree.

2.5. Case 3.0

Case 3 is an SDOF but with yield or slip represented by a hyperbolic tangent, that is linear at small values of velocity but at a certain point goes to a constant value, this might be a good candidate in pile driving as seeing the pile sliding and facing resistance as certain point is intuitive.

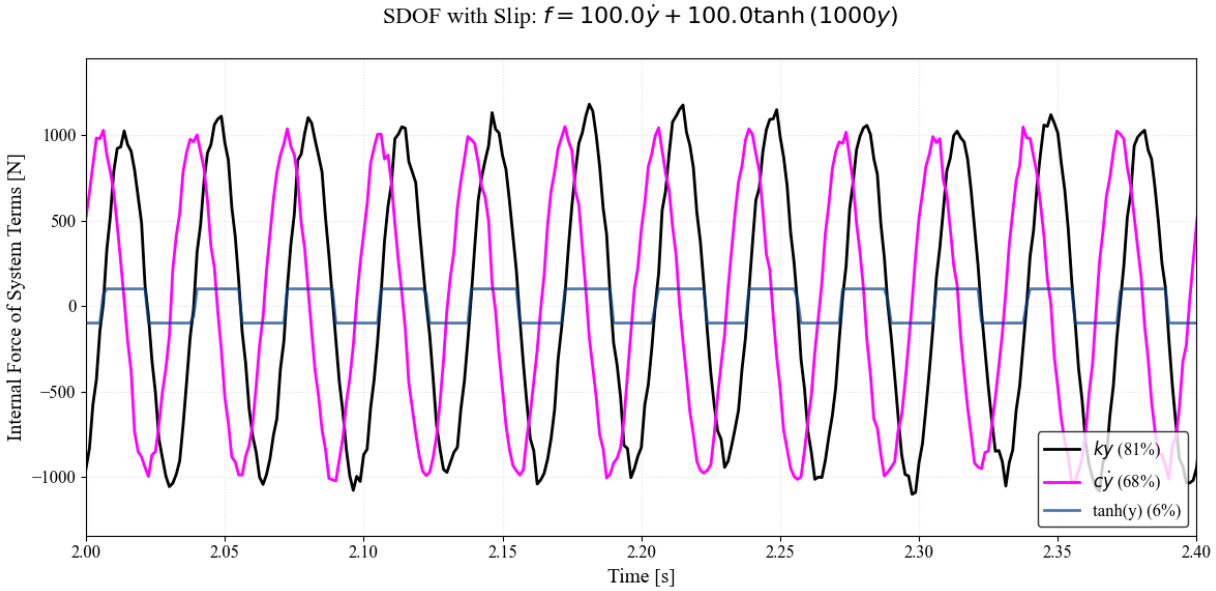


Figure 2.29: Case 3.0 Contribution of system terms to overall internal force as percentage of external forcing

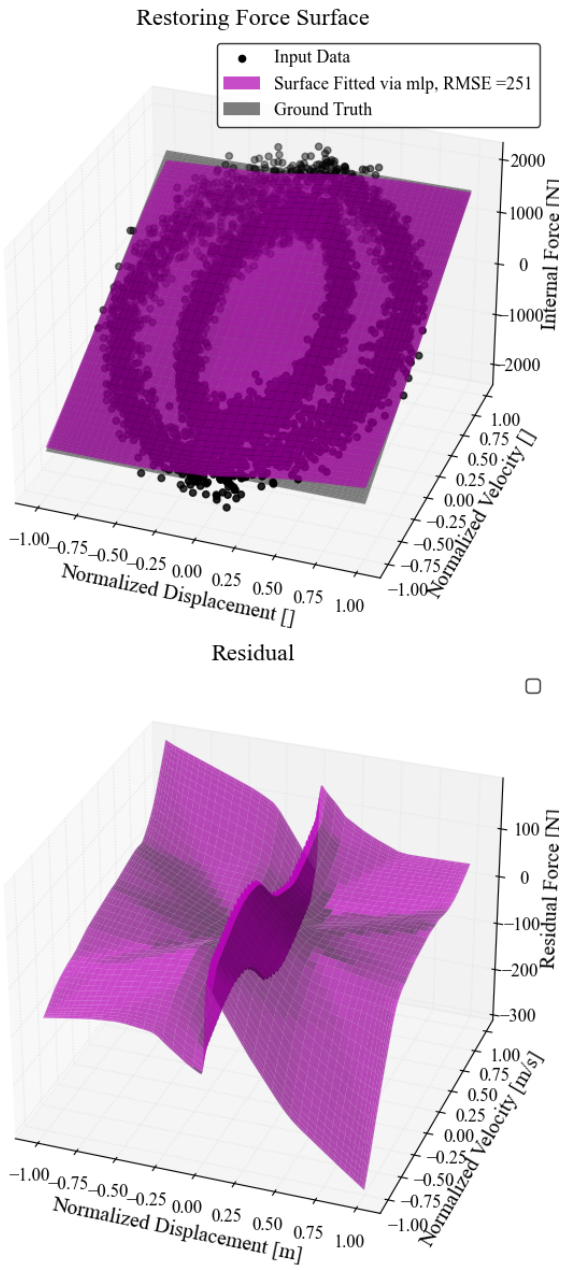


Figure 2.30: Case 3.0: Selected Restoring Force Surface

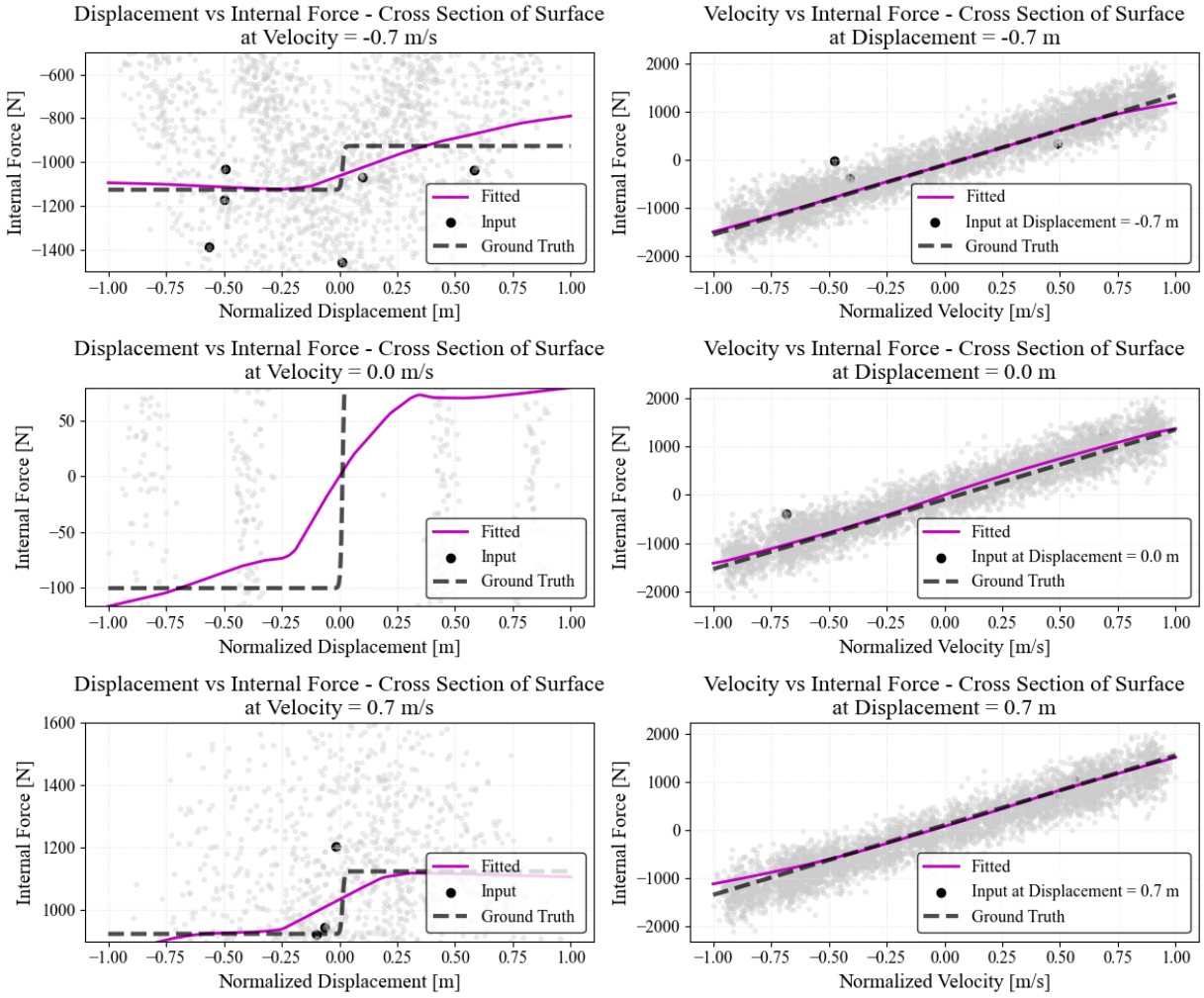


Figure 2.31: Case 3.0 : Cross Sections of the surface including side views of the simulated 3D dataset (internal force vs displacement and velocity)

The fitted RFS clearly shows the slip, or jump in terms of a plateau, however for the intermediary area between having a constant resisting force is shown as a linear-like trend due to lack of data points around the middle, the fact that the rough shape of a plastic yield is clear then we can accordingly update our candidate function library with something that would produce such a shape, as polynomials will not be sufficient to approximate this.

As per the previous cases, the initial library is just a simple list of polynomials (up to power 5)

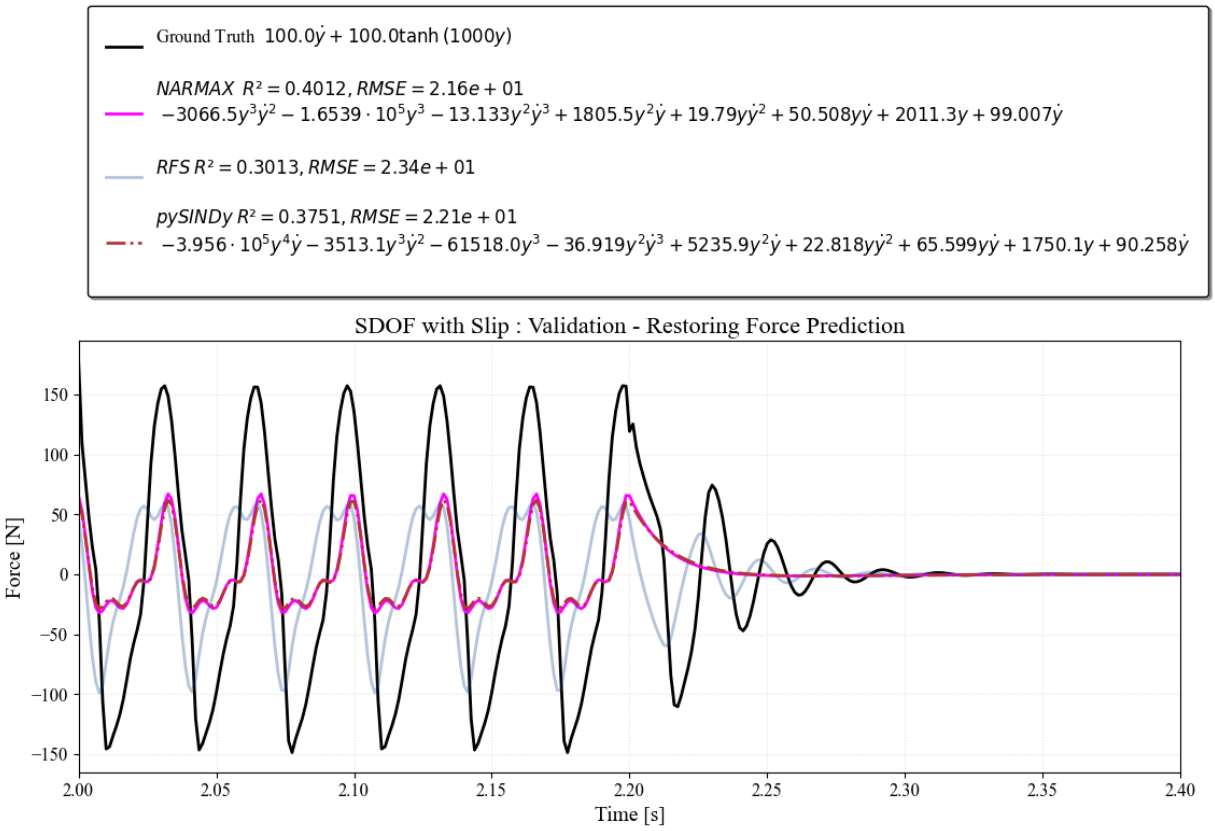


Figure 2.32: Case 3.0 : Validation - Restoring Force Prediction

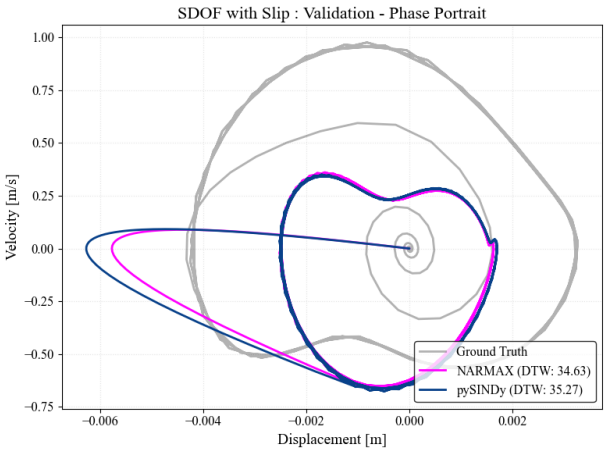


Figure 2.33: Case 3.0 : Validation - Phase Portrait

Note: The phase portrait also display a good fit, comparison here was used via the Dynamic time warping (DTW) value, see appendix F which is basically the absolute distance between the phase plots, as both x and y axes are variables and statistics such as R^2 on each on individually doesn't say much about the portrait. Its value is hard to interpret, but Custom RX outperforms pySINDy by having a lower DTW value.

The 3 plots for the validation case confirms our doubts and there is zero confidence in the fit of all 3 systems in terms of predicting response to a new excitation, or anything considerable different from the original force used for getting the training data.

2.6. Case 3.2

Another trial of case 3.0 but now, judging based on the RFS surface cross section in direction of displacement, the shape of a hyperbolic tangent was introduced, any function that starts with a linear trend then converges to a straight line can work. however fine tuning the location of convergence as well as slope maybe difficult additionally noise was also introduced to complicated tuning the function further

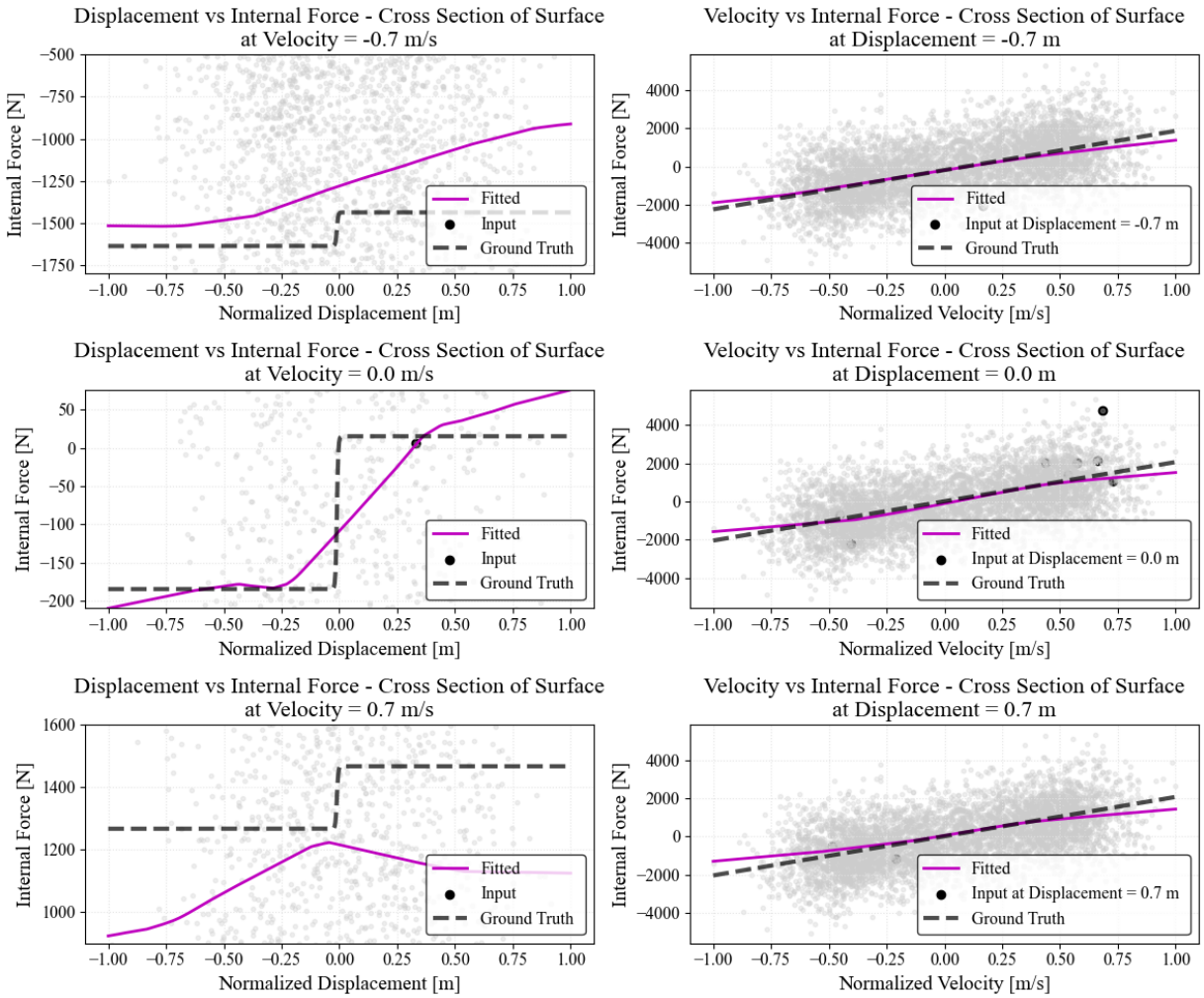


Figure 2.34: Case 3.2 : Cross Sections of the surface including side views of the simulated 3D dataset (internal force vs displacement and velocity)

Cross sections above show that the added noise was of greater magnitude than the effect of the hyperbolic tangent stiffness component, it's barely noticeable around $\dot{y} = 0$ as yield on the negative displacement going to become a positive linear trend, thus guess the shape of a hyperbolic tangent will not show up if its contribution is small enough to be muddled up with noise during instrumentation. However for the sake of investigation a hyperbolic tangent was added into the candidate function library

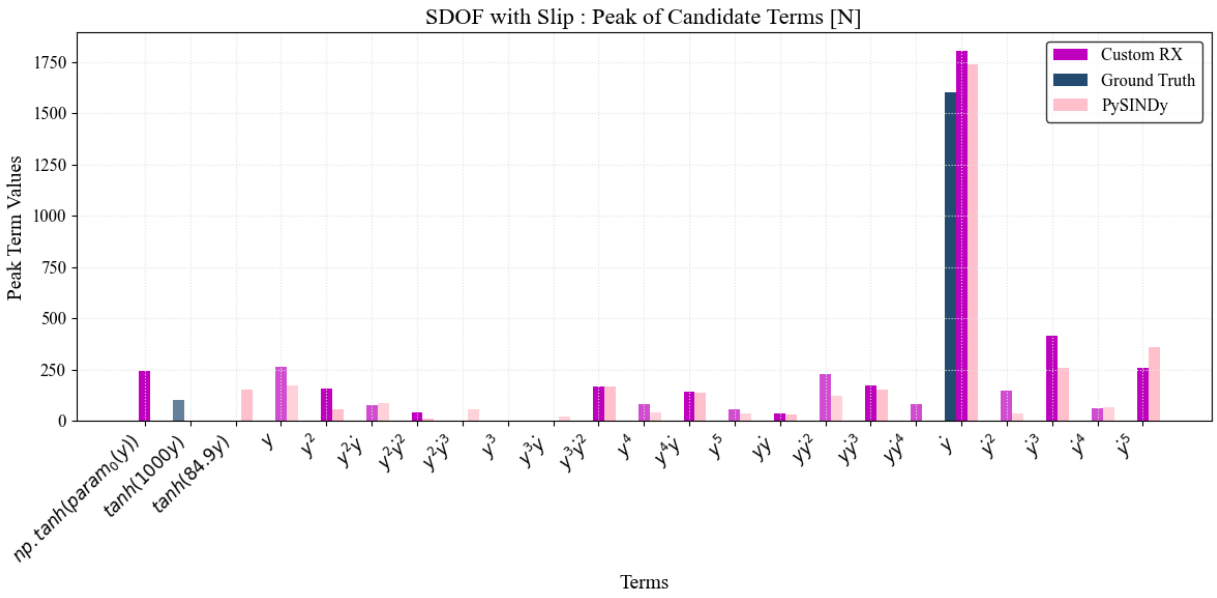


Figure 2.35: Case 3.2 : Force Features comparison

where $param_0$ = 84.9

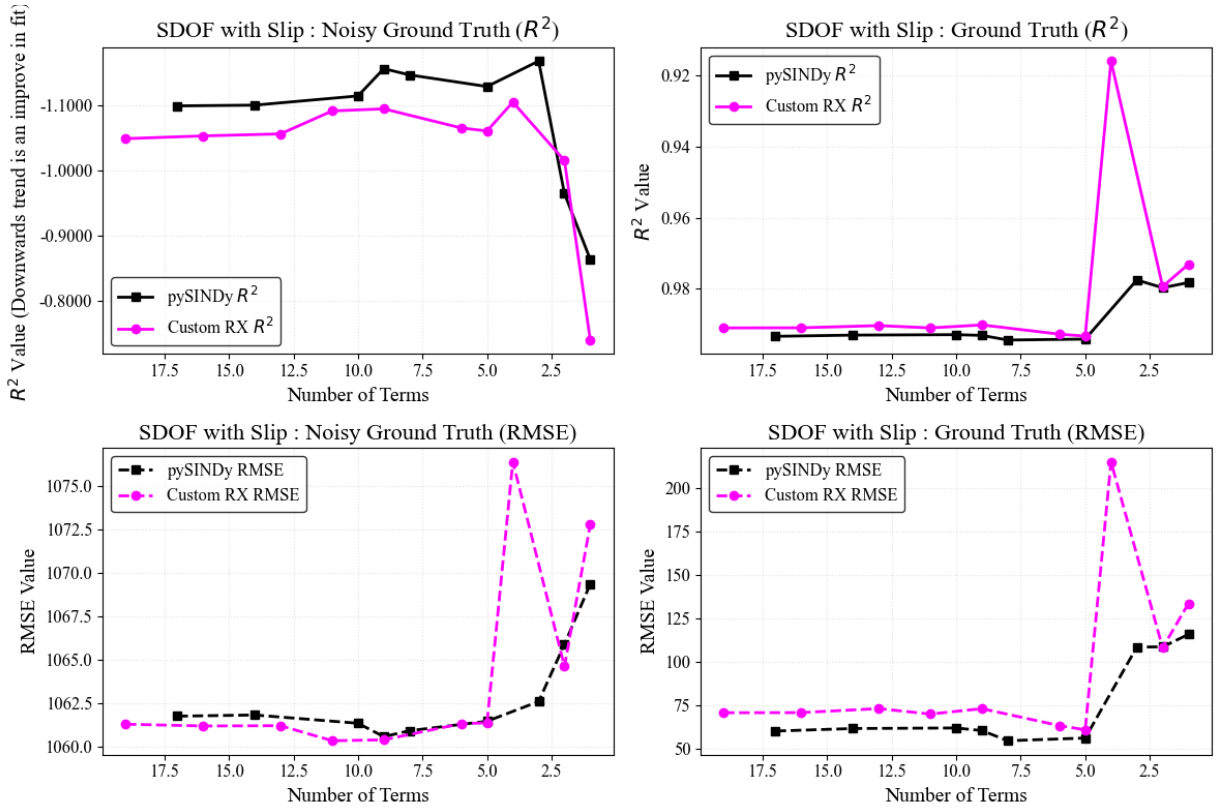
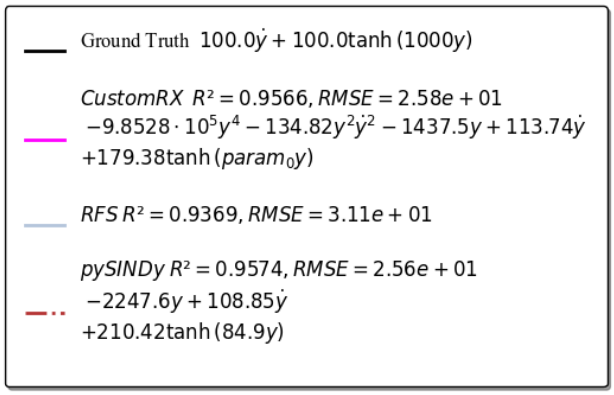


Figure 2.36: Case 3.2 : Removing Unnecessary Terms

As seen above in figure , the noise completely loses significance of the R^2 value as it does not see beyond it and the RMSE value is superior in cutting off terms, and suggests a cut-off of 9 terms for both the noisy and pristine signals.



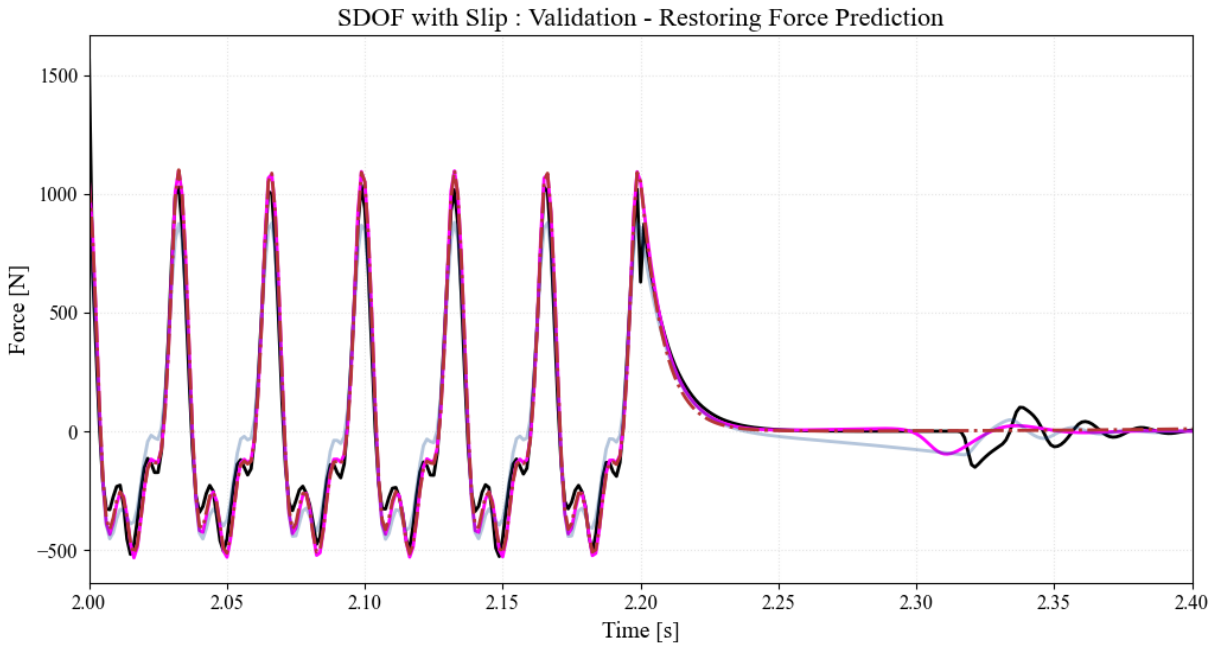


Figure 2.37: Case 3.2 : Validation - Restoring Force Prediction

A very important observation is that the custom RX model got a negative hyperbolic tangent and combined it with polynomials so that the shape is maintained.

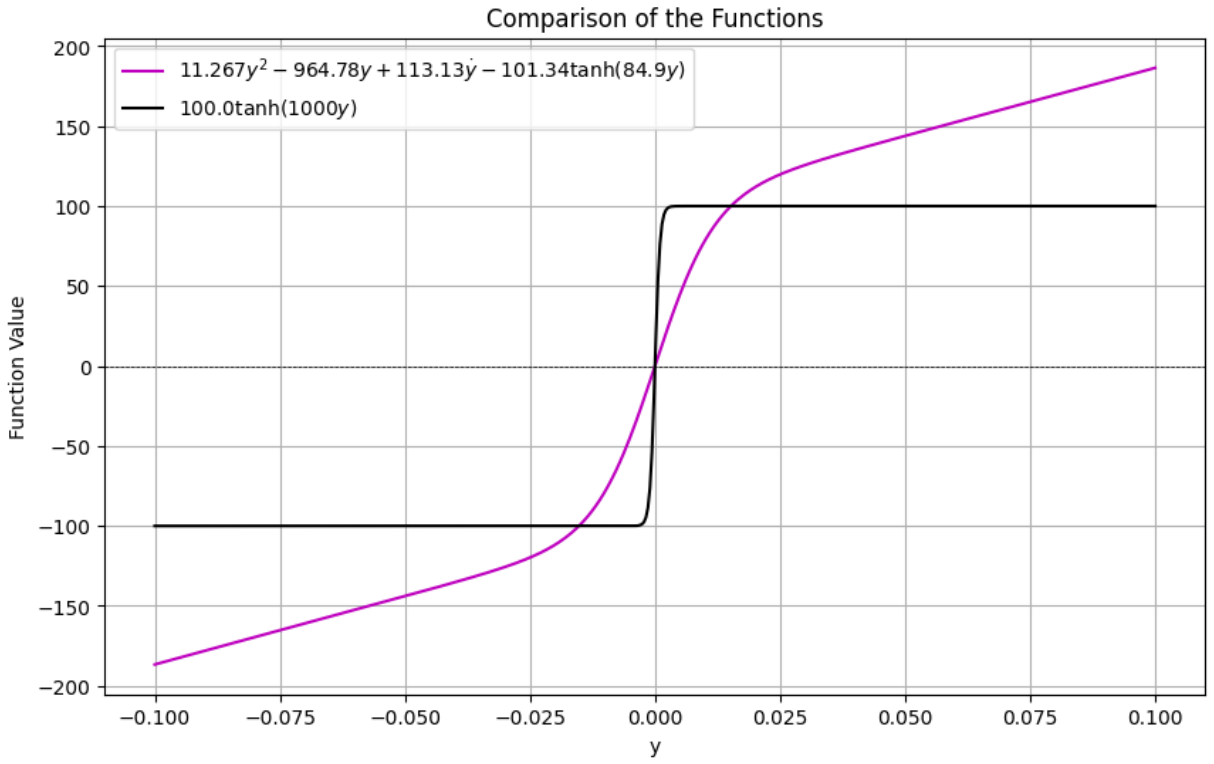


Figure 2.38: Case 3.2 Custom RX Hyperbolic tangent Fit

the fit is poor at higher values of displacement however due to the nature of the system chosen the respond revolves around low values of y (this can be adjusted by adding stiffness components that would reduce the internal force going into acceleration when slip occurs and there is no resistance from the hyperbolic tangent stiffness)

2.7. Final Notes on Benchmark Cases

- All three approaches (RFS, Custom RX, pySINDy) aim to fit a 3D surface in similar ways by arranging matrices, defining an objective function, and minimizing it, the methods vary but the idea is the same.
- Unlike physics-informed methods, RFS relies solely on parameter and hyper-parameter tuning, without incorporating any physical insights. thus use of a ”candidate library” to try informing physics beyond problem formulation is not possible, as is the case for many machine learning pipeline approaches.
- Identification results assume that experimental data filtering is accurate. Propagating measurement and signal processing uncertainties is essential for reliable system identification.
- The benchmarks cases clearly show that multi-excitation frequency reduces the quality of fit, which makes sense as each term e.g. y^3 behaves differently per frequency, in the context of pile driving having a main harmonic force in my training data is expected and using a single frequency force is not an over-simplification.
- The identification process is computationally efficient compared to solving state-space equations. This makes it feasible to integrate machine learning modules, such as neural networks, for inferring function shapes. While basic combinations of candidate functions (as tested with `pySINDy`) failed to provide proper identification, neural networks could automate shape inference and identify suitable candidates. Currently, this process has been done manually but could be automated in the future.
- Adding memory terms (previous states) significantly complicates identification. Simplified RX models are a practical starting point, but the process is not additive. Unlike the Hammerstein-Wiener model, where a known block can remain constant, RX models require constant refitting with every adjustment, such as adding a term, history term, or dataset.
- Normalization is crucial to ensure equal feature contributions, especially for regularization techniques like Lasso regression, which are sensitive to scale. Traditional normalization methods are vulnerable to outliers, whereas median and interquartile range-based methods are more robust. However, normalizing nonlinear features (e.g., y^2, y^{10}) introduces complexities and was not considered here. Chebyshev expansions were used instead of Taylor series, as they avoid exponential growth and are better suited for simulations. Feature engineering literature addresses these issues, but techniques for nonlinear systems remain limited.
- Random additive noise effectively simulates Gaussian noise but oversimplifies real-world conditions. Advanced noise models, such as coloured, multiplicative, or Poisson noise, better represent real-world systems. Adding noise without careful consideration risks hiding critical nonlinear system features, and a more robust noise model would have been preferable.
- The use of fractional polynomials (e.g., $y^{3.5}$) instead of integer powers (e.g., y^3) is a valid consideration. However, such refinements fall under fine-tuning and are more suitable for later stages of model optimization. Early inclusion of fractional powers raises regression dimensionality and degrades performance. The focus should first be on identifying the major terms of the system before optimizing implicit parameters.
- Reformulating to include velocity- or displacement-dependent mass terms (e.g., $A\dot{y}\ddot{y}$) is doable although would be more computationally expensive, e.g : for this SDOF with an acceleration dependent term:

  $$m\ddot{y} + ky + c\dot{y} + A\dot{y}\ddot{y} = f(t).$$

  state variables:

  $$x_1 = y, \quad x_2 = \dot{y}.$$

  State-space formulation:

  $$\dot{x}_1 = x_2,$$
  $$\dot{x}_2 = \frac{f(t) - kx_1 - cx_2}{m + Ax_2}.$$
  $$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{f(t) - kx_1 - cx_2}{m + Ax_2} \end{bmatrix}.$$

  the denominator may prove challenging numerically but nothing unmanageable.

Vibratory Pile Driving Frequency Range

Vibratory pile driving typically operates within the range of 20-40 Hz. In Case 1, an almost perfect fit slightly diverged from the original system at higher frequencies (contribution of ”extra” terms). However, as no high-frequency test forces were available for reference, care must be taken with higher harmonics (multiples of frequency due to nonlinearities) during driving and it’s clear that at higher harmonics the fit may not be adequate.

3

Analysis and Results

3.1. Pile Driving Experiment and Data Processing

Experimental Setup

In this study, a pile was suspended using a crane and loaded using a hydraulic vibratory device as it was lowered. Unlike conventional methods where a pile is driven purely by the vibratory device, this setup involved lowering the pile at a controlled rate while applying the hammer's force. The experimental data obtained from this process were carefully processed and filtered to for subsequent analysis.

The experimental setup, part of a lab-scale testing program, is depicted in Fig. 3.1. The primary objective of this current investigation is to derive a symbolic formulation of the soil reaction force. To achieve this, various data-driven methods were considered as potential candidates for identifying the underlying dynamics of the system.

Figure 3.1: Experimental setup, showing the pile held by the crane and the pneumatic hammer.

The pile used in this study weighs 49 kg and is sufficiently short and stiff to be reasonably modelled as a single degree of freedom (Single degree of freedom (SDOF)) system. This assumption allows us to treat the pile as a rigid body. The equilibrium of forces acting on the pile is described by the following equation:

$$m_p \ddot{u}_p + R = F_h, \tag{3.1}$$

where:

- m_p is the mass of the pile,
- u_p is the pile displacement, with $\ddot{u}_p$ representing the pile acceleration (obtained using accelerometers mounted at the pile head),
- R is the total pile driving resistance,
- F_h is the vibratory force applied to the pile head.

The vibratory force applied at the pile head, F_h, is computed as:

$$F_h = EA\epsilon_h, \tag{3.2}$$

where:

- E is the Young's modulus of the pile material,
- A is the cross-sectional area of the pile,
- ϵ_h is the axial strain measured at the pile head.

To measure forces acting on the pile, strain gauges were installed at the top and bottom of the pile. Using these measurements, the total pile resistance R can be divided into two components:

1. Pile tip resistance, R_t,
2. Pile shaft resistance, R_s.

The pile tip resistance is determined from strain gauges placed at the bottom of the pile and is given by:

$$R_t = EA\epsilon_t, \tag{3.3}$$

where ϵ_t is the axial strain measured at the bottom of the pile.

The shaft friction, R_s, is then determined from the equilibrium condition as:

$$R_s = F_h - R_t - m_p\ddot{u}_p. \tag{3.4}$$

formulation of the regression problem is identical to the benchmark cases

$$m_p\ddot{y} + R_s + R_t = F(t) \tag{3.5}$$

$$R_s + R_t = F(t) - m_p\ddot{y} \tag{3.6}$$

It was assumed that both shaft and tip forces can be identified separately thus setting one to zero to get a symbolic expression for the other. this assumption is discussed further in chapter 2.7.

The strain gauges provide critical measurements at the top and bottom of the pile, allowing us to compute the tip resistance R_t directly. The pile shaft resistance R_s is then calculated using the equilibrium equation, ensuring that all forces acting on the pile are accounted for.

This method effectively separates the contributions of the pile shaft friction and tip resistance to the total pile driving resistance. The approach leverages the short length and high stiffness of the pile, simplifying the dynamics to an SDOF model.

How would developing a SDOF surrogate model further the field in terms of vibratory pile driving?

If a equation of motion is developed, this can then be extended into MDOF systems (see formulation is appendix J), applying the same key candidates of the equation onto multiple elements for shaft or tip. Thus enabling modelling of full-sized piles not rigid enough to be considered 1 DOF (unlike the test pile). Being able to simulate an accurate response would help in minimizing the total energy required to drive the pile, i.e. by minimizing the internal force or resistance to driving and maximizing acceleration, this can be done by simulating different frequencies & amplitudes of force and then calibrating it be redoing testing with the optimized combinations of force and frequency and see how well that works. Additional calibration of the model would be possible, gaps of data in the loading cycles can be filled by having harmonic forces that would lead to a more spread out response.

Test Data

The utilized data is 4 controlled rate pile driving experiments with a sampling rate of $0.001250s$, with total time varying from $10 - 110s$ best results are as expected from the slow tests and focus will be on them,

Adopted Name in Thesis	Test Number	Frequency	Lowering Rate
Test 1	54	High Frequency	Slow
Test 2	55	High Frequency	Fast
Test 3	56	Normal Frequency	Slow
Test 4	57	Normal Frequency	Fast

Table 3.1: Summary of tests with adopted names, frequency, and lowering rates.

however different combinations of tests/ segments were considered.

Figure 3.2 illustrate the filtered measurement of Test 3

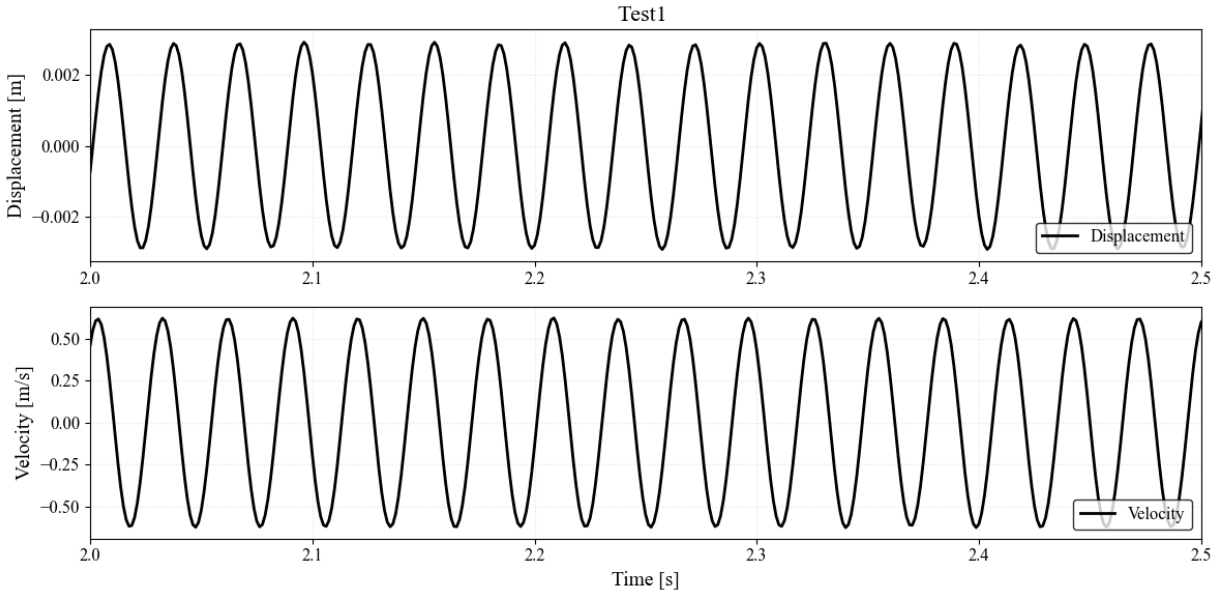


Figure 3.2: Filtered displacement and velocity measurements of Test 1

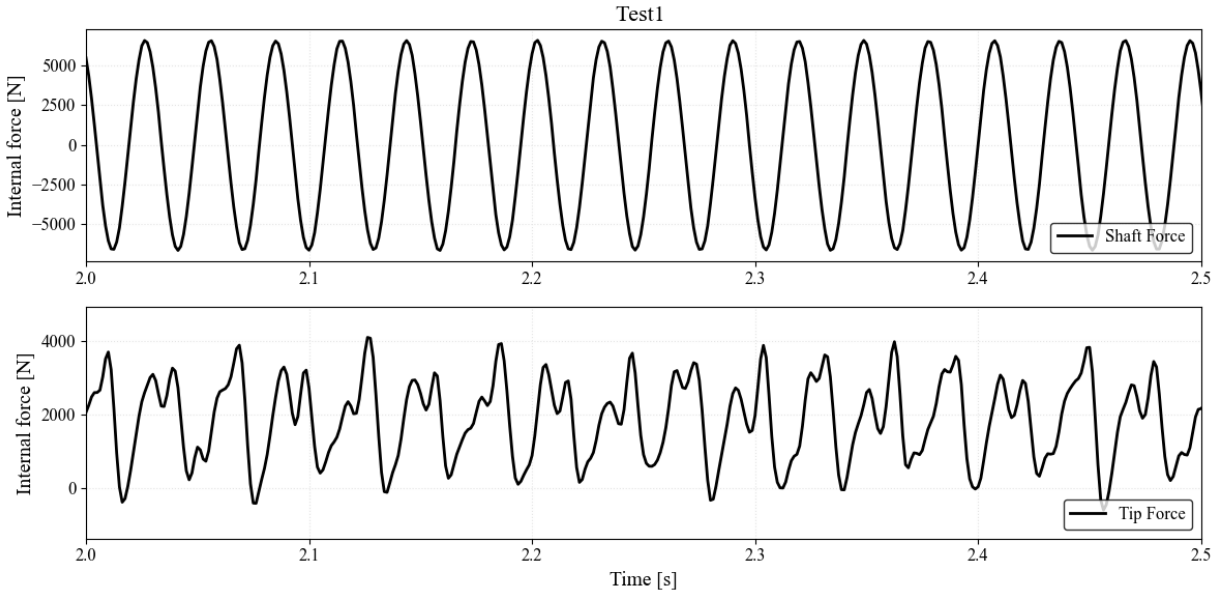


Figure 3.3: Calculated tip and shaft resisting forces of Test 1

Figure 3.3 shows the target forces of the identification process, deduced from measurements of stress, strain, displacements as well as velocities and accelerations of the pile as described in the beginning of chapter 3.1 .

Candidate Nonlinear Force Functions

Literature provides an abundance of empirically and statistically formulated functions that represent prominent types of nonlinearity. These were utilized during the search process. Drawing from select previous works on nonlinearities (Menq and Griffin [21], Worden and Tomlinson[32], and especially Gondhalekar [8]), several formulations of nonlinear resisting forces were considered as candidates. These were chosen based on guesses of possible phenomena to be observed .

It is worth noting that the naming and classification of these phenomena vary significantly across the literature plots are provided for clarity. **Cubic Stiffness**

$$g^{cub} = \beta y^3$$

where β is the coefficient of cubic stiffness non-linearity.

Clearance

$$g^{cle} = \frac{K_z \cdot y}{2\pi}\left[\pi - 2\theta_c + \sin(2\theta_c) - \frac{4y_c}{y}\cos(\theta_c)\right]$$

where:

$$\theta_c = \sin^{-1}\left(\frac{y_c}{y}\right)$$

K_z is the additional stiffness after the clearance gap is closed, and y_c is the gap distance.

Friction

Stick Region:

$$g^{fri}_{\text{stick}} = \frac{K_d y}{\pi}\left[\theta_l - \sin(\theta_l)\cos(\theta_l)\right]$$

where:

$$\theta_l = \cos^{-1}\left(1 - \frac{2\mu N}{K_d y}\right)$$

Slip Region:

$$g^{fri}_{\text{slip}} = -\frac{4\mu N}{\pi}\left[1 - \frac{\mu N}{K_d y}\right]$$

In the above equations:

- K_d is the tangential stiffness in the stick region.
- μ is the coefficient of friction.
- N is the normal reaction force.

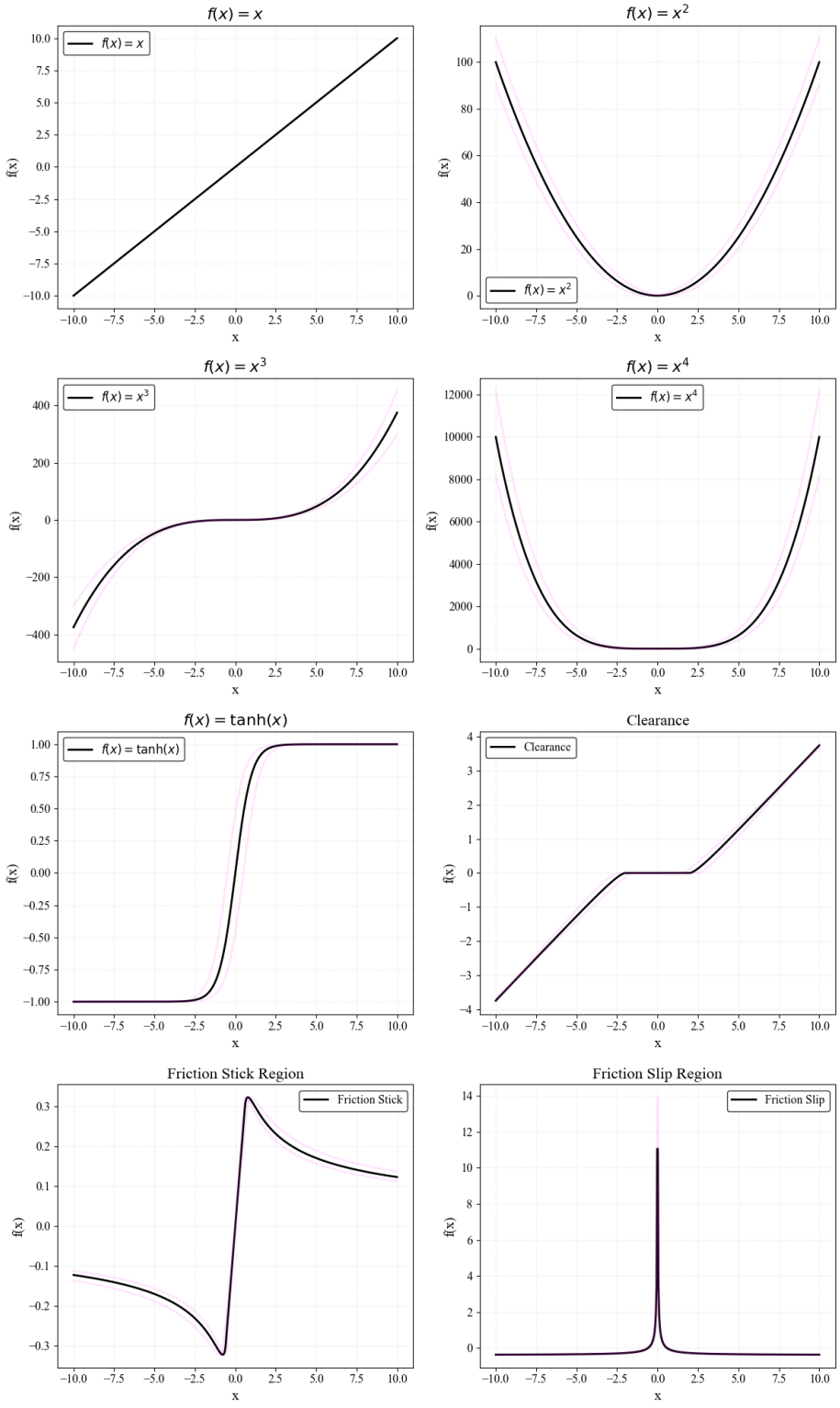


Figure 3.4: Portion of the function library used, from literature based nonlinearities to basic polynomials

Then finally, some customized functions based on the neural network derived restoring force surfaces of Tests 1 and 3 were added as candidates as seen in figure 3.5

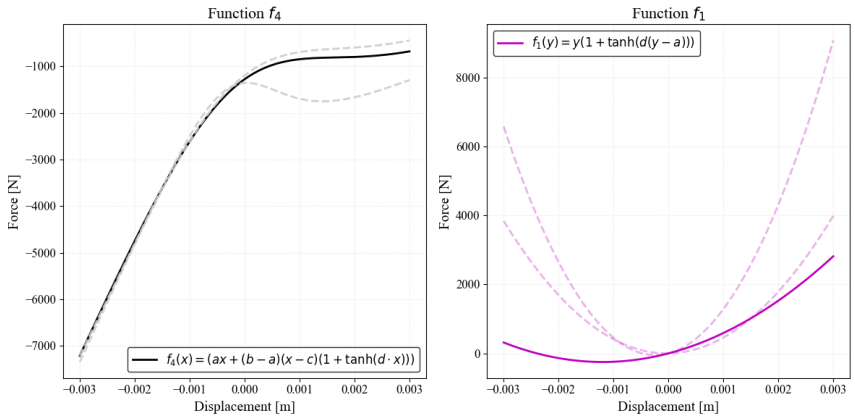


Figure 3.5: Custom candidate functions based on the RFS of shaft and tip forces

Windowing the Signals

Across the test data, the initial approach was to window the response by excluding the first 1-2 seconds of the transient response and focusing on windows highlighting significant variations (e.g increase in force, velocity or displacement) and find fitting terms for each of the segments. The goal was to observe how parameters, such as linear stiffness ky, change over time. However, this approach proved ineffective, as the symbolic expressions for each segment did not consistently include the same terms. To account for time-variant effects, the data cannot be divided into segments and must be analyzed as a whole.

For the four tests conducted, the best fits were obtained from the slower tests (Tests 1 and 3). The following results are based on fits derived from Test 1 and Test 3, using data from the interval 2–$10\,\mathrm{s}$.

Results for Tip Force from Test 1

Custom RX :

$$(f(t) - m\ddot{y})/m = -1.657 \cdot 10^4 y^3 + 1.474 \cdot 10^5 y^2\dot{y} - 3.417 \cdot 10^4 y\dot{y}^2 - 3.427 \cdot 10^4 y\dot{y} - 1.763 \cdot 10^6 y + 76.62\dot{y}^3 + 20.54\dot{y} + 3.544 \cdot 10^6\, \mathsf{f}_1\,(y, -0.0056, 2)$$

pySINDy :

$$(f(t) - m\ddot{y})/m = -2.348 \cdot 10^9 y^3 - 2.44 \cdot 10^6 y^2\dot{y} + 6.529 \cdot 10^4 y - 257.9\dot{y}^3 - 80.07\dot{y}$$

RFS :

$$(f(t) - m\ddot{y})/m = 2.523y^2\dot{y}^2 + 3.967y^2\dot{y} - 27.83y^2 + 3.342y\dot{y}^2 + 5.638y\dot{y} - 18.48y + 2.463\dot{y}^2 - 19.11\dot{y} - 158.7$$

The restoring force surface sections appear as follow:

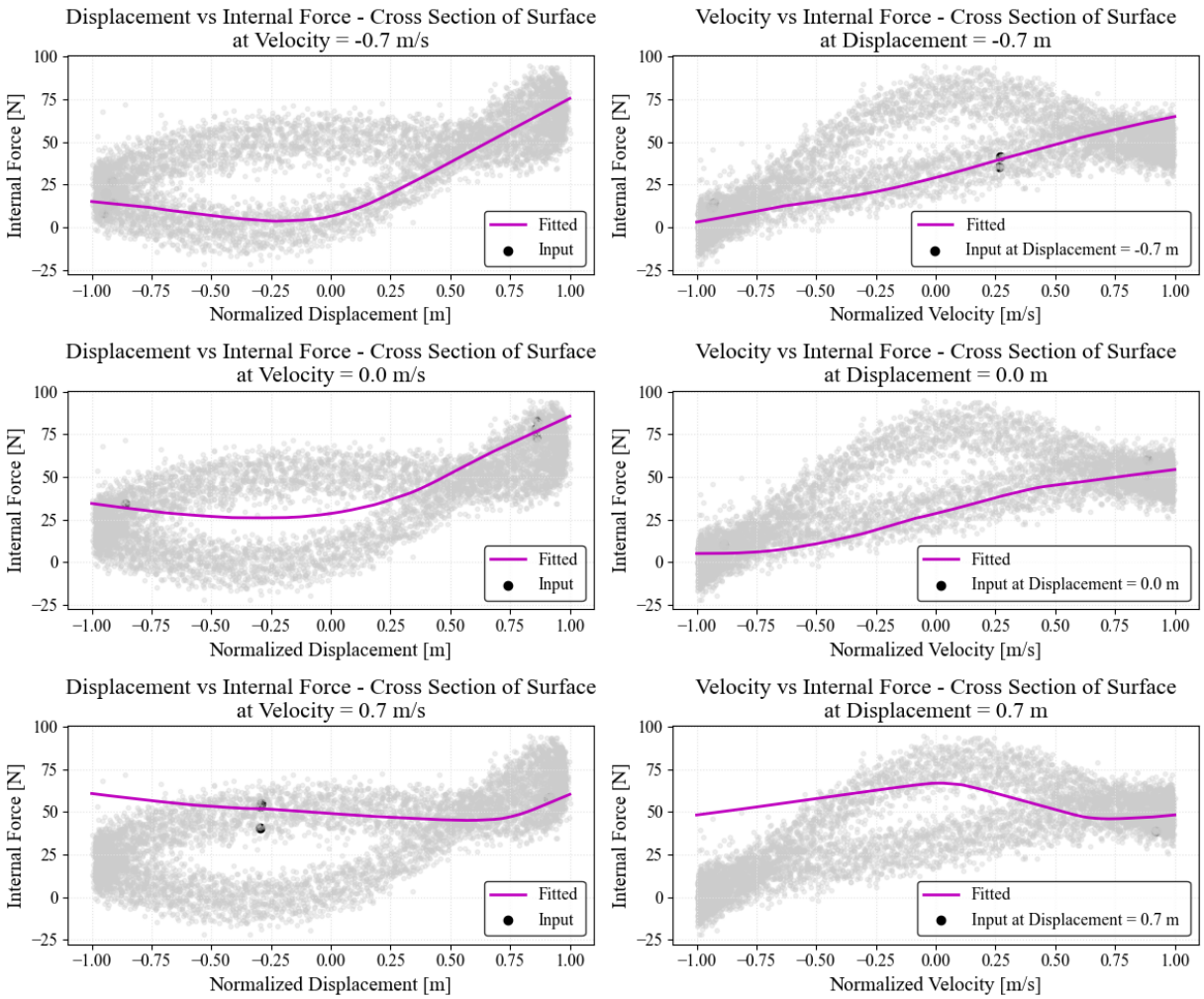


Figure 3.6: Tip Force from Test 1 : Cross Sections of the surface including side views of the measured 3D dataset (tip force vs displacement and velocity)

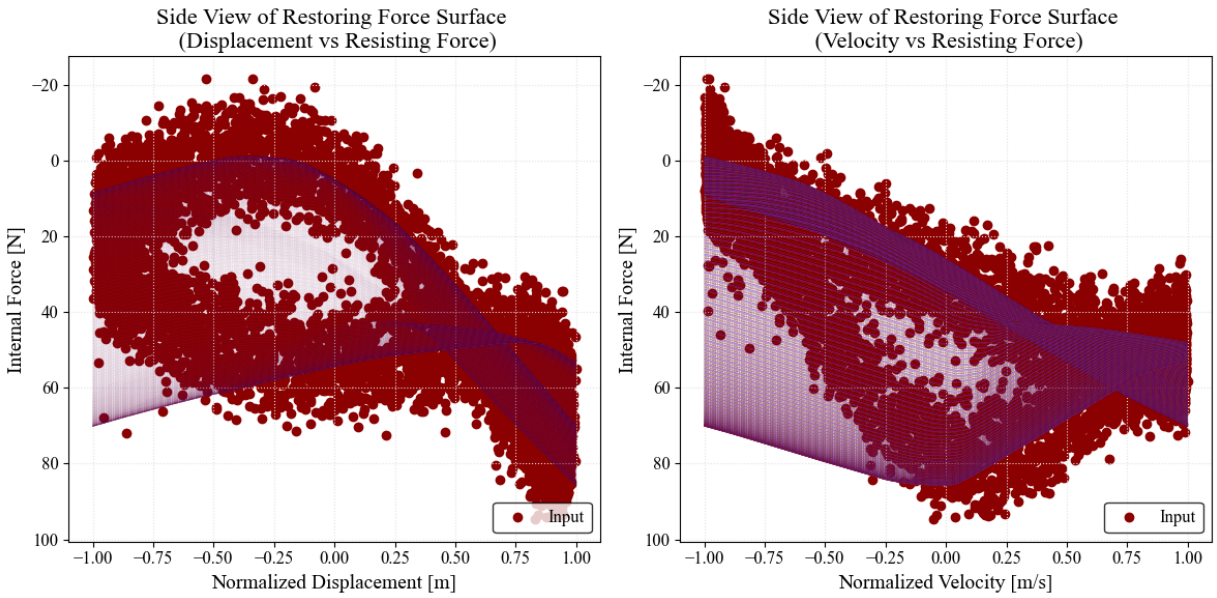


Figure 3.7: Tip Force from Test 1 : Side Views of Surface

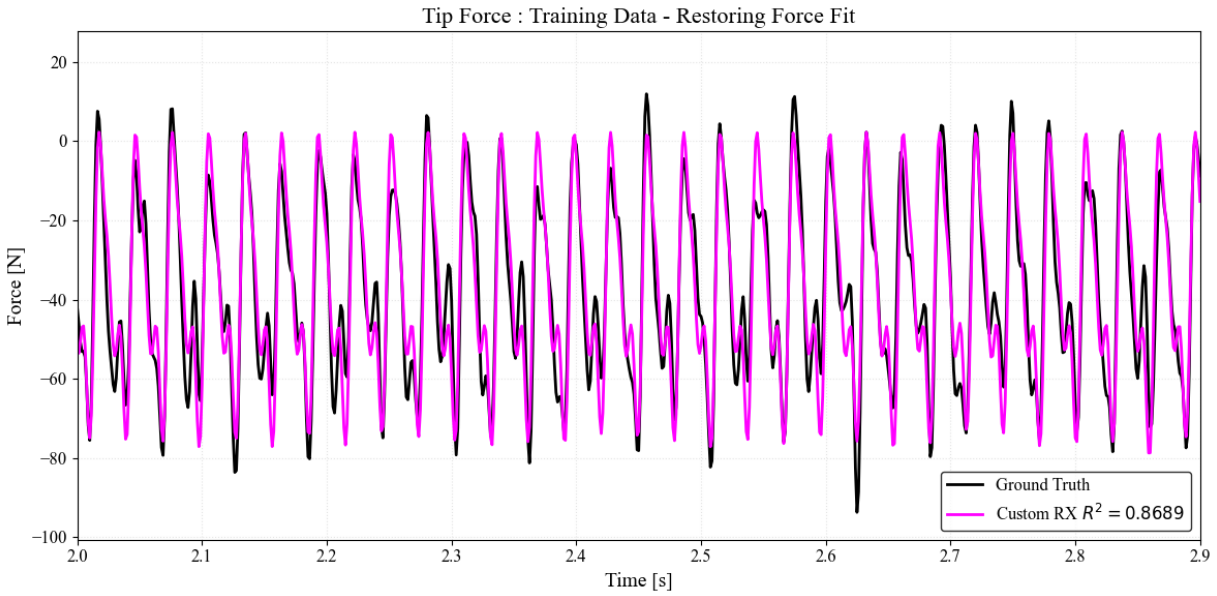


Figure 3.8: Tip Force from Test 1 : Training Data - Restoring Force Fit

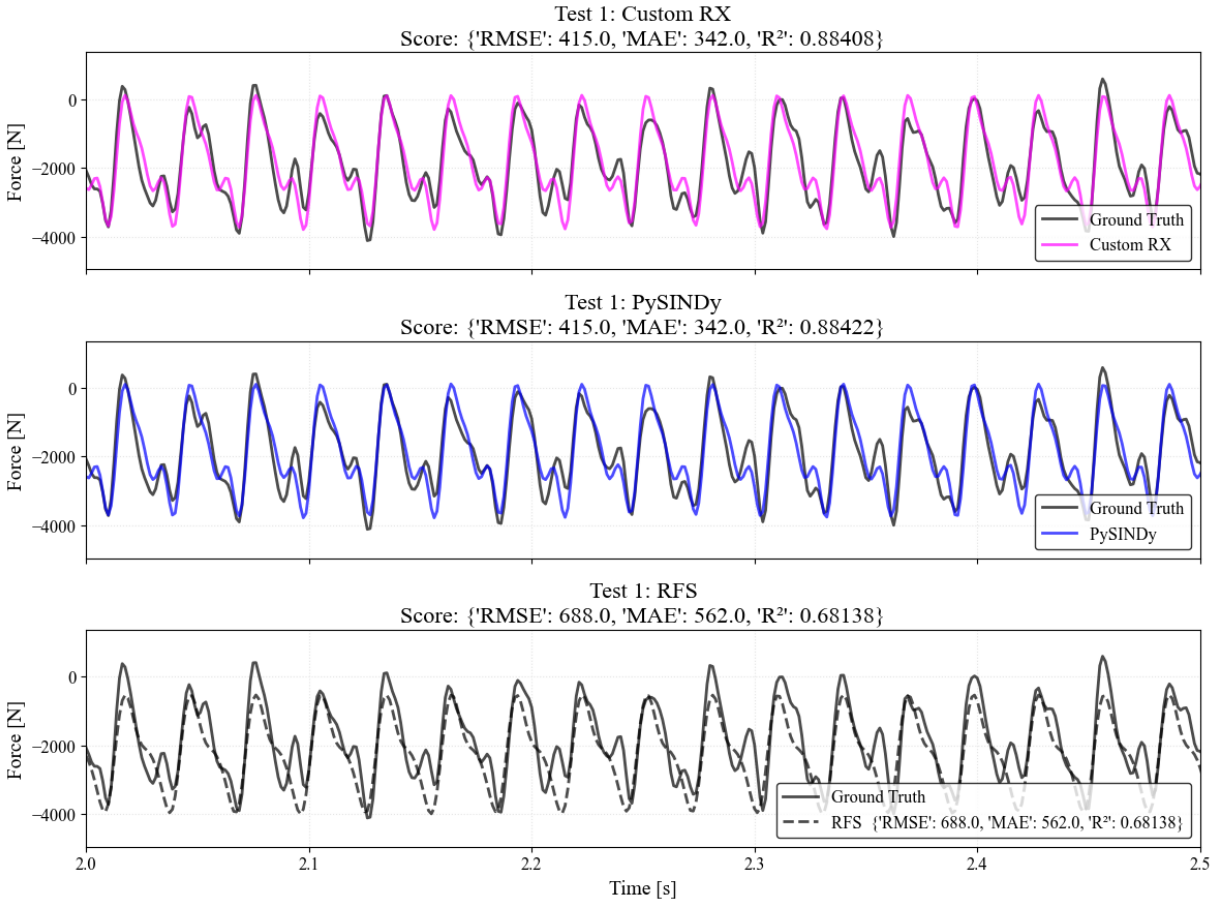


Figure 3.9: Tip Force from Test 1 : Test 1 Data - Fit for t =2-10s

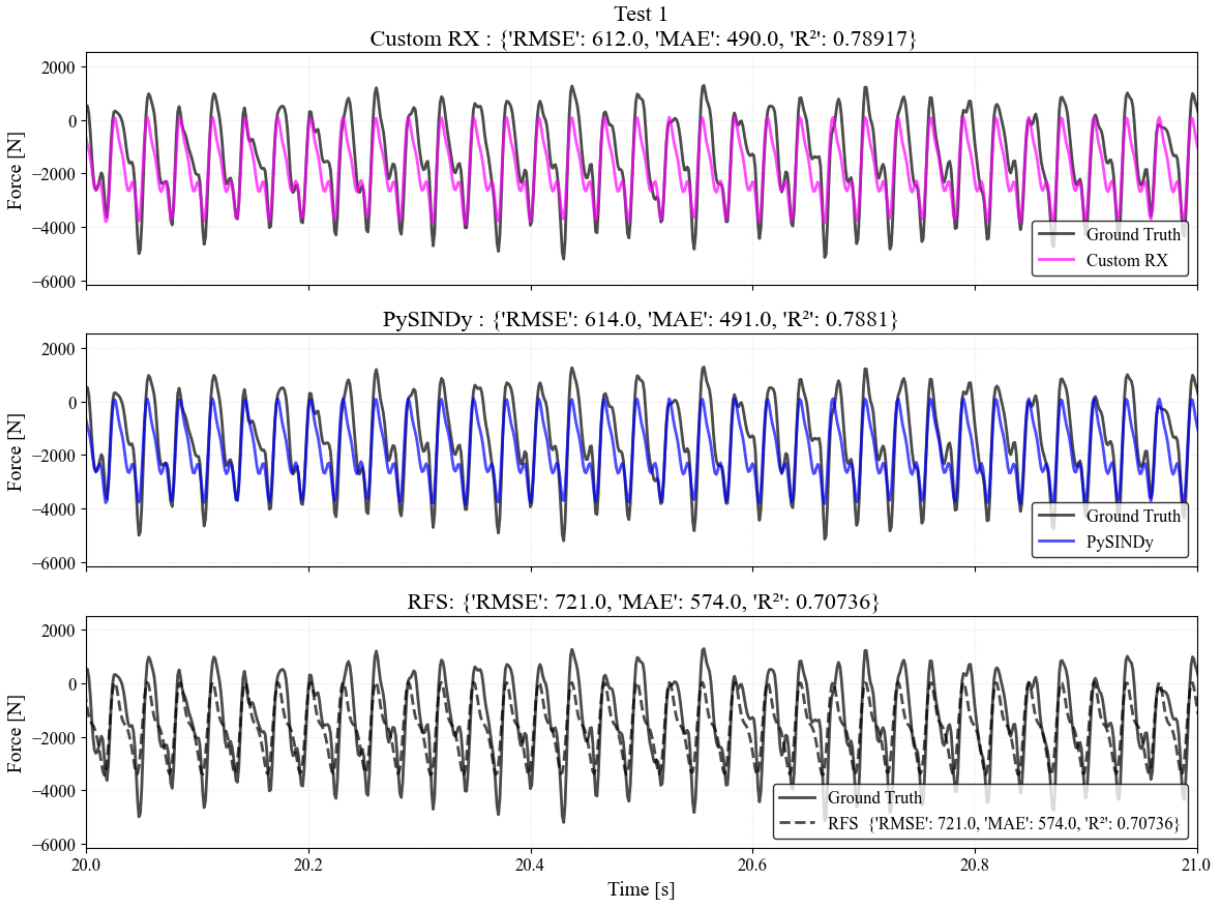


Figure 3.10: Tip Force from Test 1 : Test 1 Data - Fit for t =2-40s

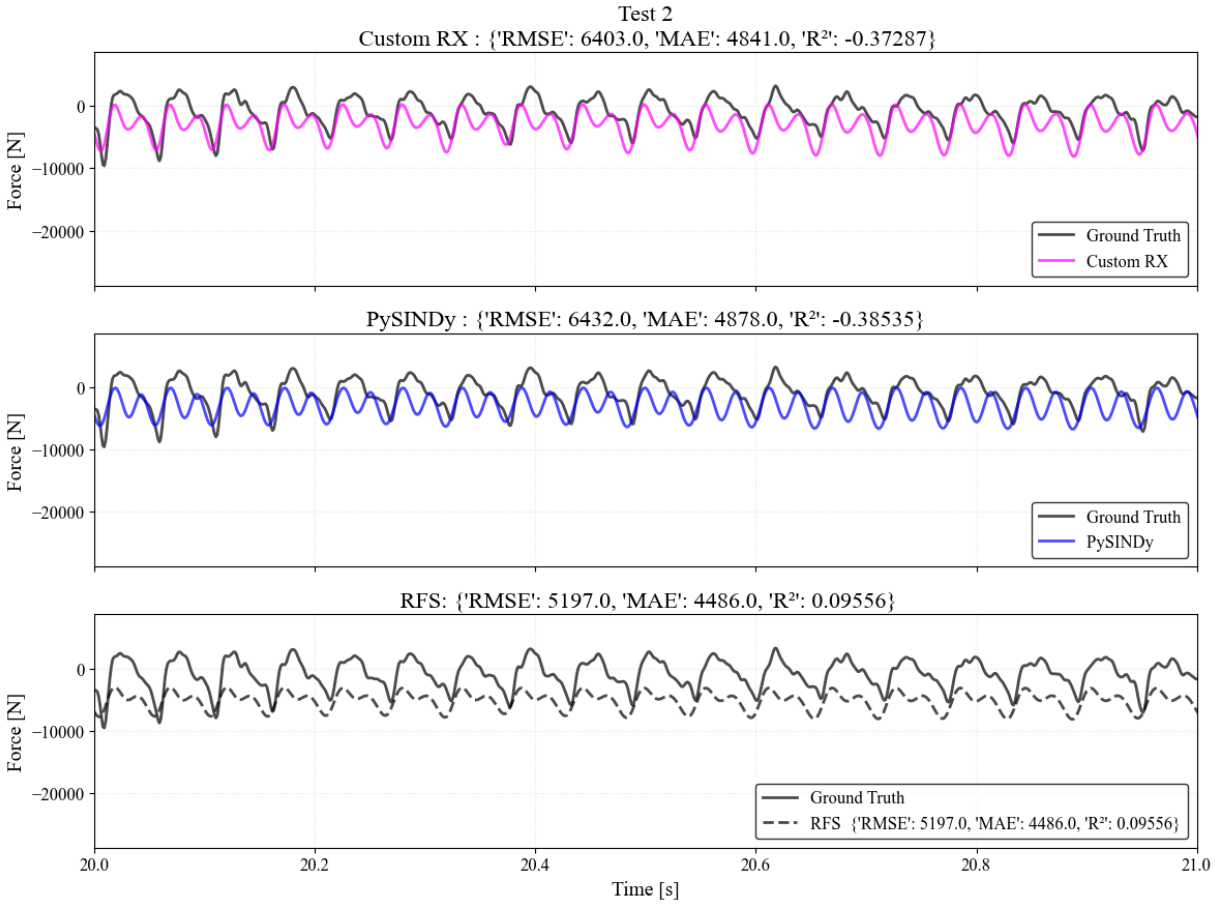


Figure 3.11: Tip Force from Test 1 : Test 2 Data - Fit for t =2-40s

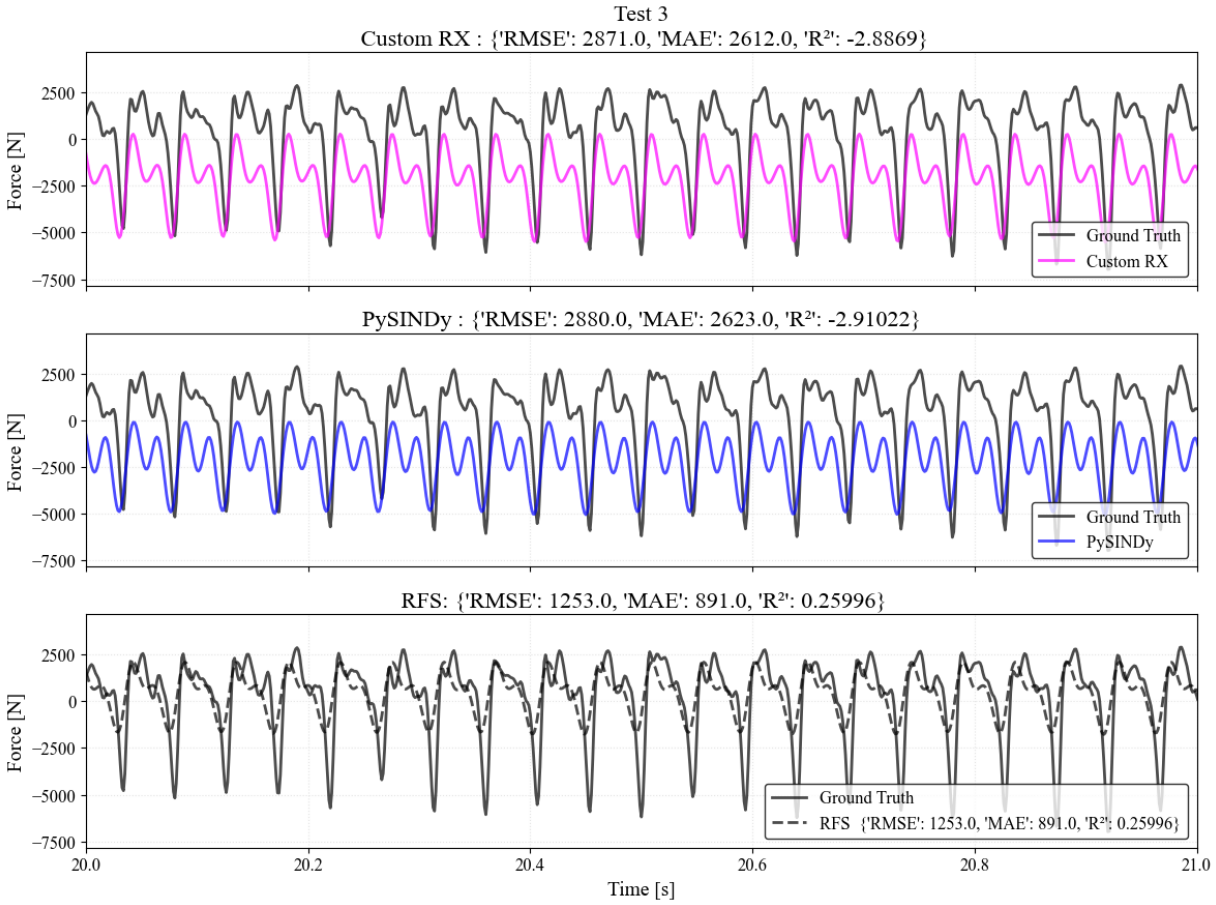


Figure 3.12: Tip Force from Test 1 : Test 3 Data - Fit for t =2-40s

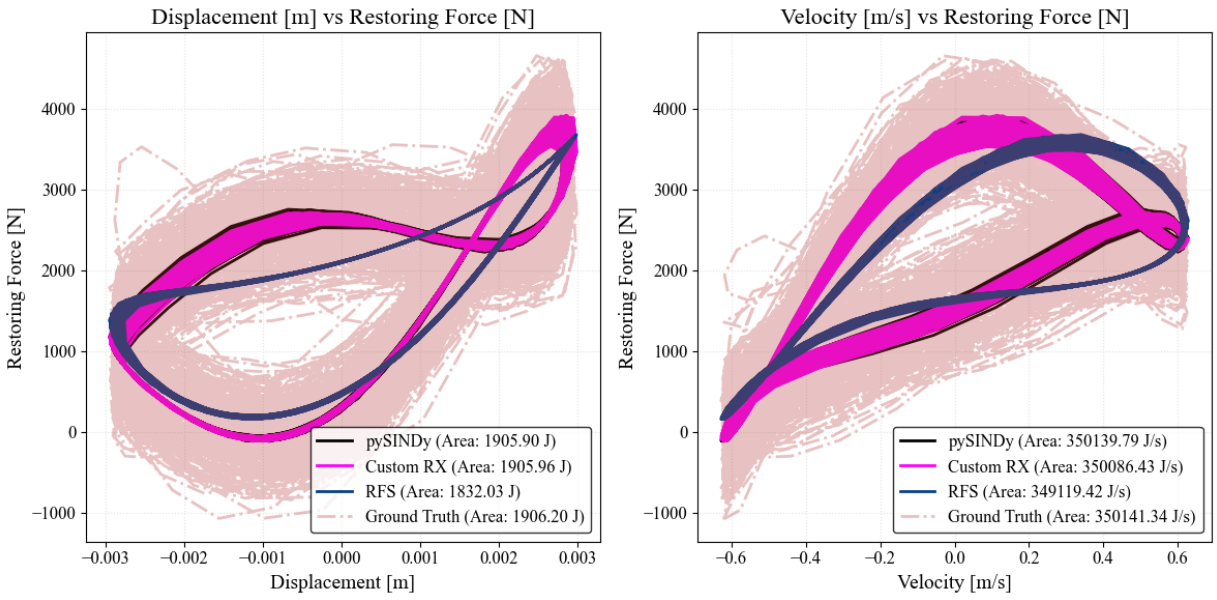


Figure 3.13: Tip Force from Test 1: Fitted loading Cycles

Results for Shaft Force

Following the exact procedure of the benchmark cases, the derived force expressions are as follows: For brevity RFS expression is not included as it quite extensive, refer to appendix B

Custom RX :

$$(f(t) - m\ddot{y})/m = \\ -1.92 \cdot 10^4 y^2\dot{y} - 5.163 \cdot 10^6 y^2 - 2944.0 y\dot{y}^2 + \\ 1.822 \cdot 10^4 y\dot{y} + 4.324 \cdot 10^4 y + 50.68\dot{y}^3 - 79.02\dot{y}^2 - 204.1\dot{y}$$

pySINDy :

$$(f(t) - m\ddot{y})/m = \\ -3.08 \cdot 10^7 y^2\dot{y} - 5.174 \cdot 10^6 y^2 + \\ 1.277 \cdot 10^5 y\dot{y}^2 + 1.879 \cdot 10^4 y\dot{y} + 4.323 \cdot 10^4 y - 613.1\dot{y}^3 - 80.38\dot{y}^2$$

It is to be noted from the two expressions that the two models more or less agree on the linear stiffness. a proposed function in the function library (f_4) (see Figure 3.5) was promising (two linear stiffness connected with hyperbolic tangent transition, can be as sharp as a bi-linear stiffness or look more like a plastic yield curve depending on the parameters) however the polynomial option seemed to outperform it, that may be due to lack of intermediate data points within the transition zone.

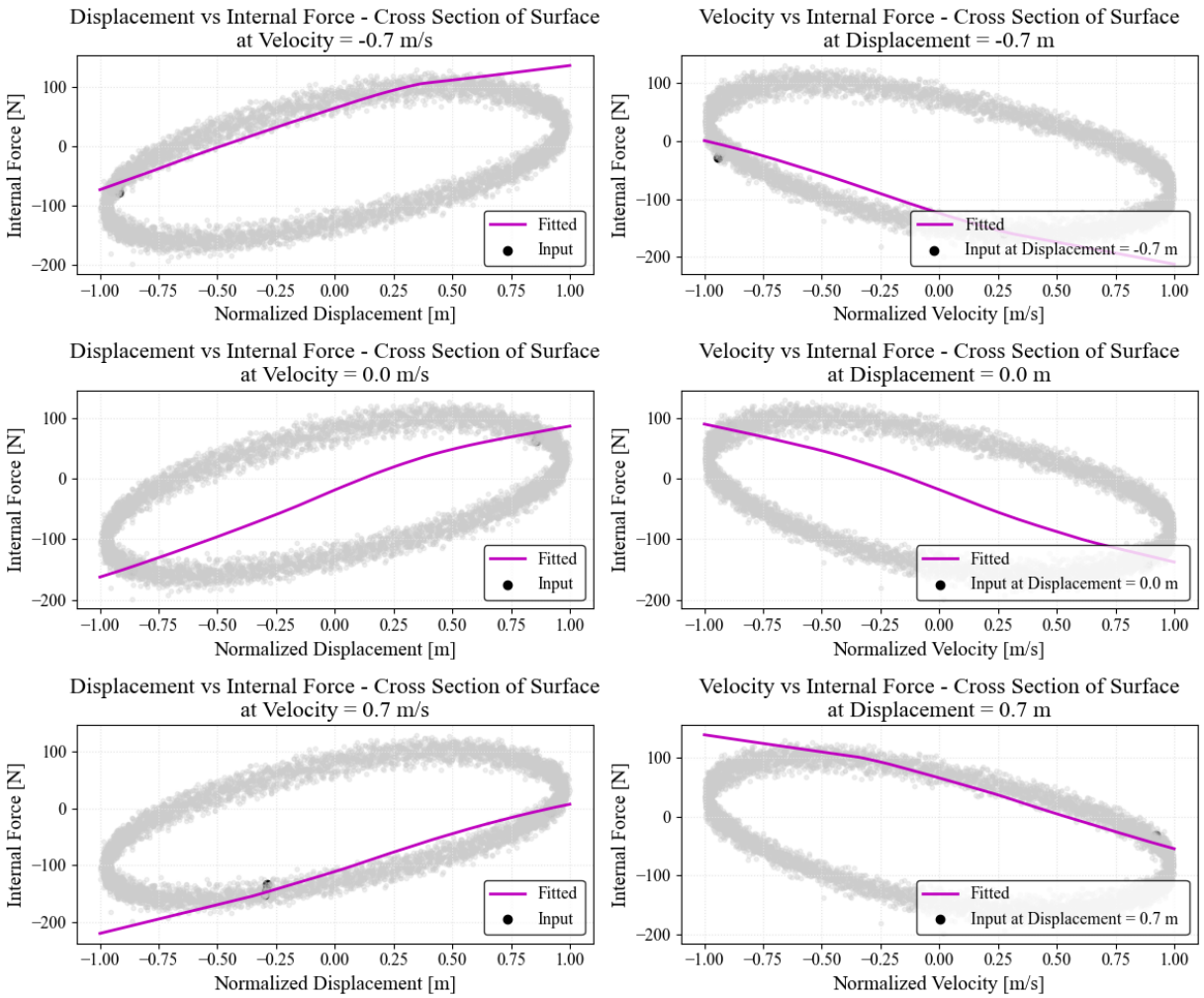


Figure 3.14: Shaft Force from Test 1 : Cross Sections of the surface including side views of the measured 3D dataset (shaft force vs displacement and velocity)

Figure 3.14shows the shaft force appearing quite linear , almost an SDOF but some softening is observed, however not many data points exist around that transition.

Then to validate, the force expressions were cross validated using the other experiments, for brevity focus is only on the expression derived from Test 1, and how it cross-validates into the forces of Test 3

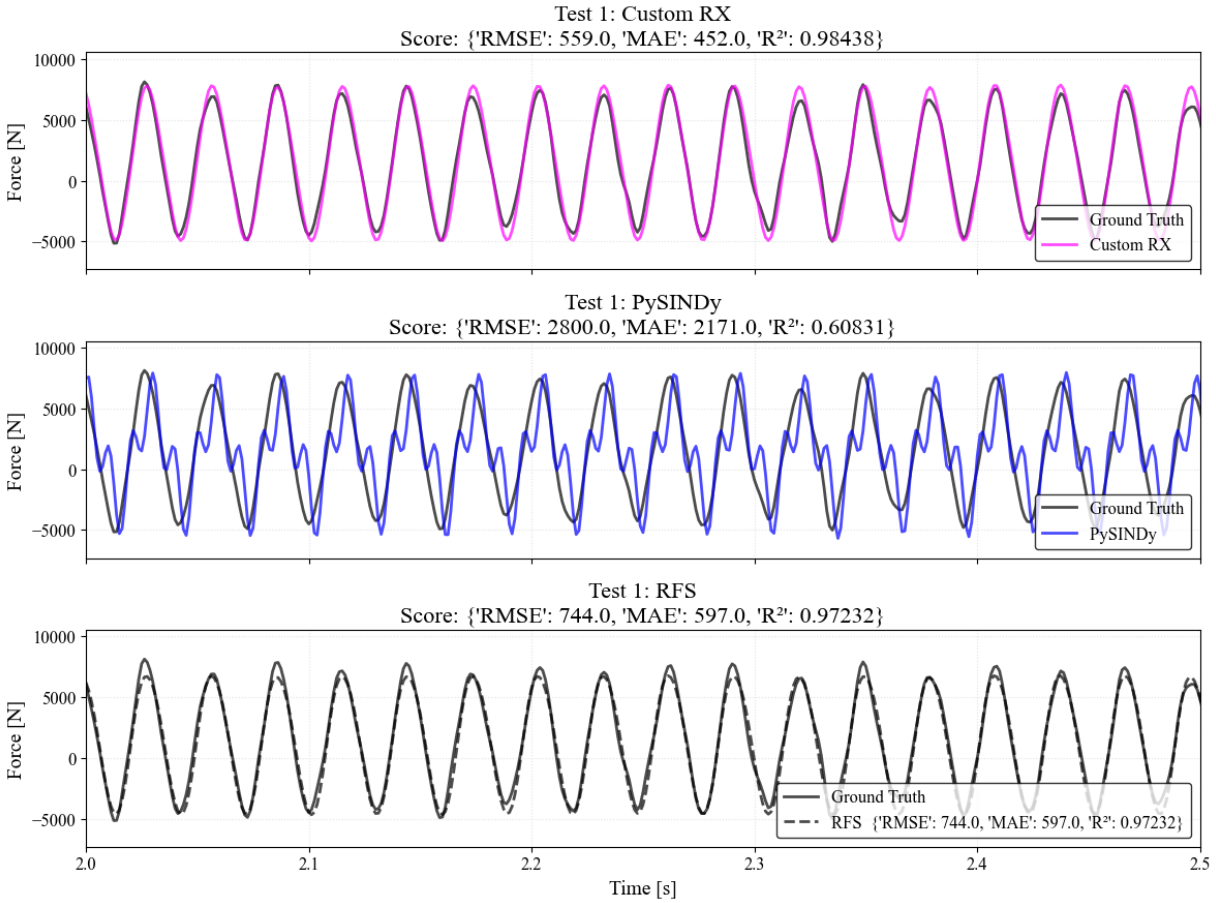


Figure 3.15: Shaft Force from Test 1 : Test 1 Data - Fit for t =2-10s

Figure 3.15above shows a great fit of the custom RX value, followed by the RFS approach,but that's not impressive seeing that the shaft force within the first 10 seconds is almost a perfect single sinusoidal signal without noticeable higher harmonics.

Another way of viewing the results is by showing the loading cycles, inspecting shape visually and the area encompassed can be alternative to the RMSE value.

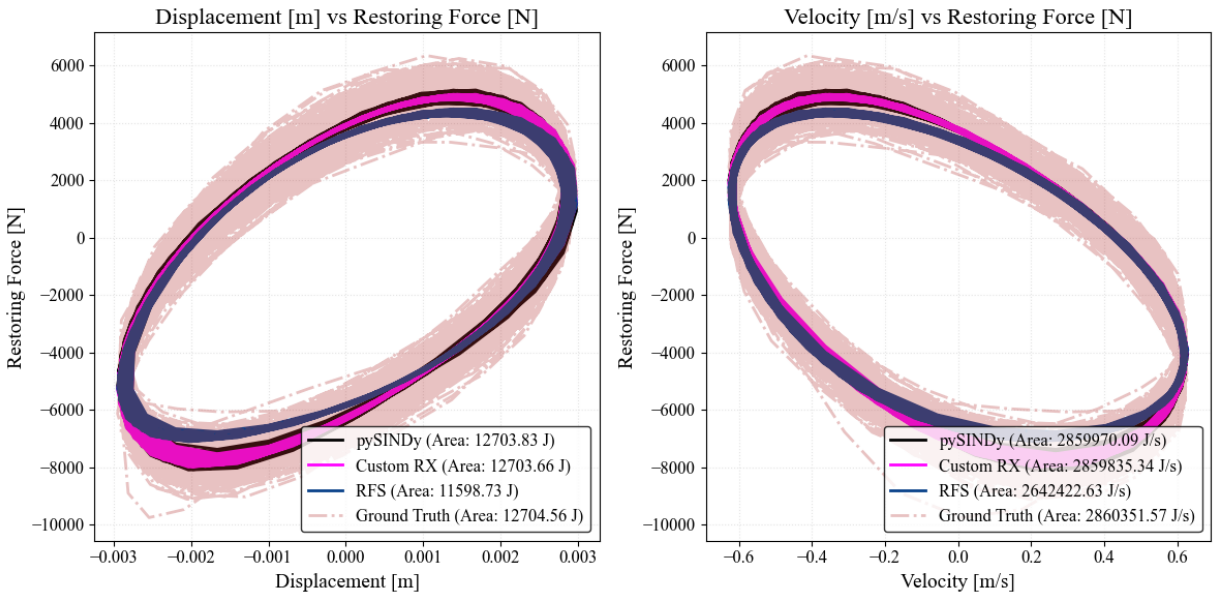


Figure 3.16: Shaft Force from Test 1: Fitted loading Cycles

Figure 3.16 shows a good agreement in terms of the fit onto the training data (shaft force of Test) however the cycles are not as tight in the experiment, it is possible that it is a time variant effect, however for our intents and purposed, matching the shape (physics) and magnitude would in itself be of great use, and getting the exact cycles would be alarming as a sign of over-fitting instead of capturing the physics.

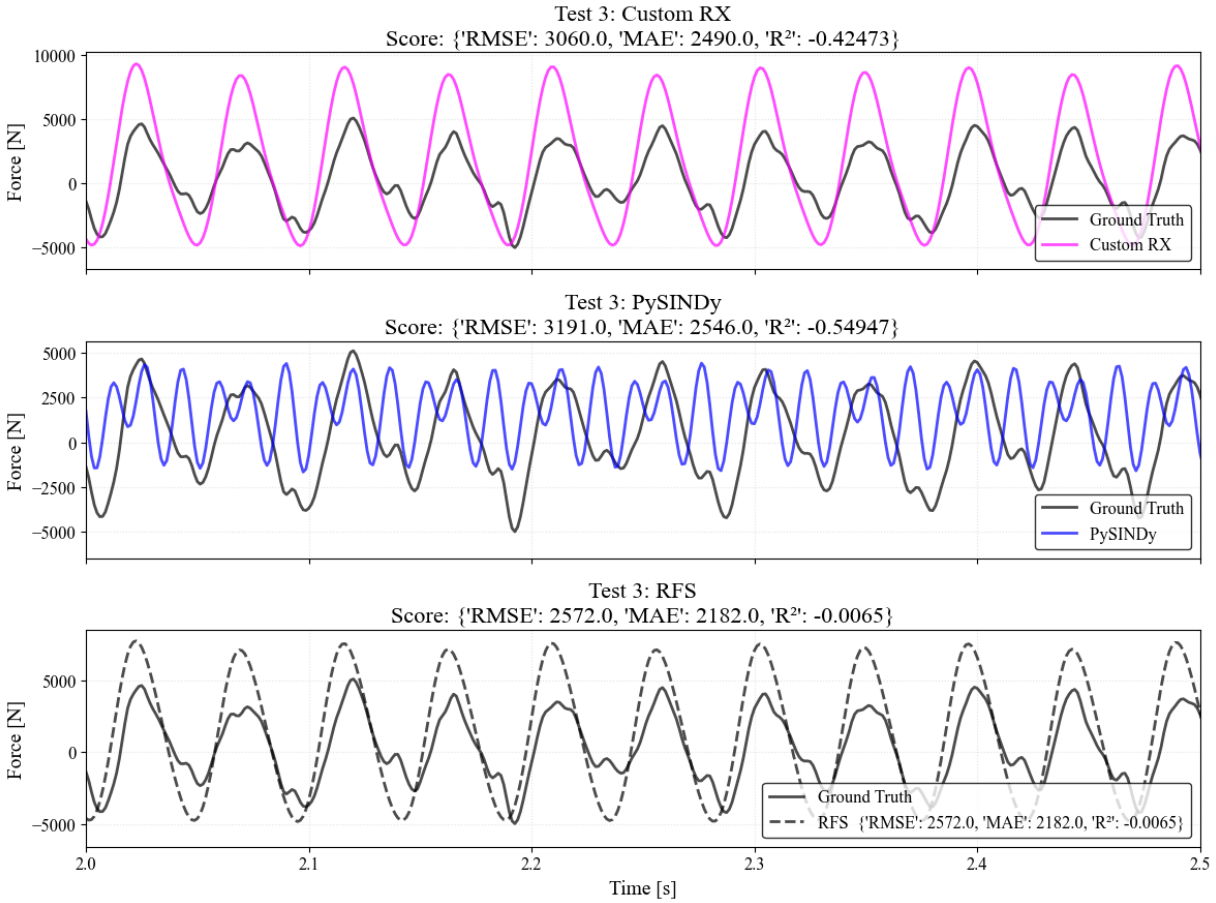


Figure 3.17: Shaft Force from Test 1 : Test 3 Data - Fit for t =2-10s

Validating by direct substitution of Test 3's displacement and velocity shows a drop in quality of fit, as not representing the entire harmonics present

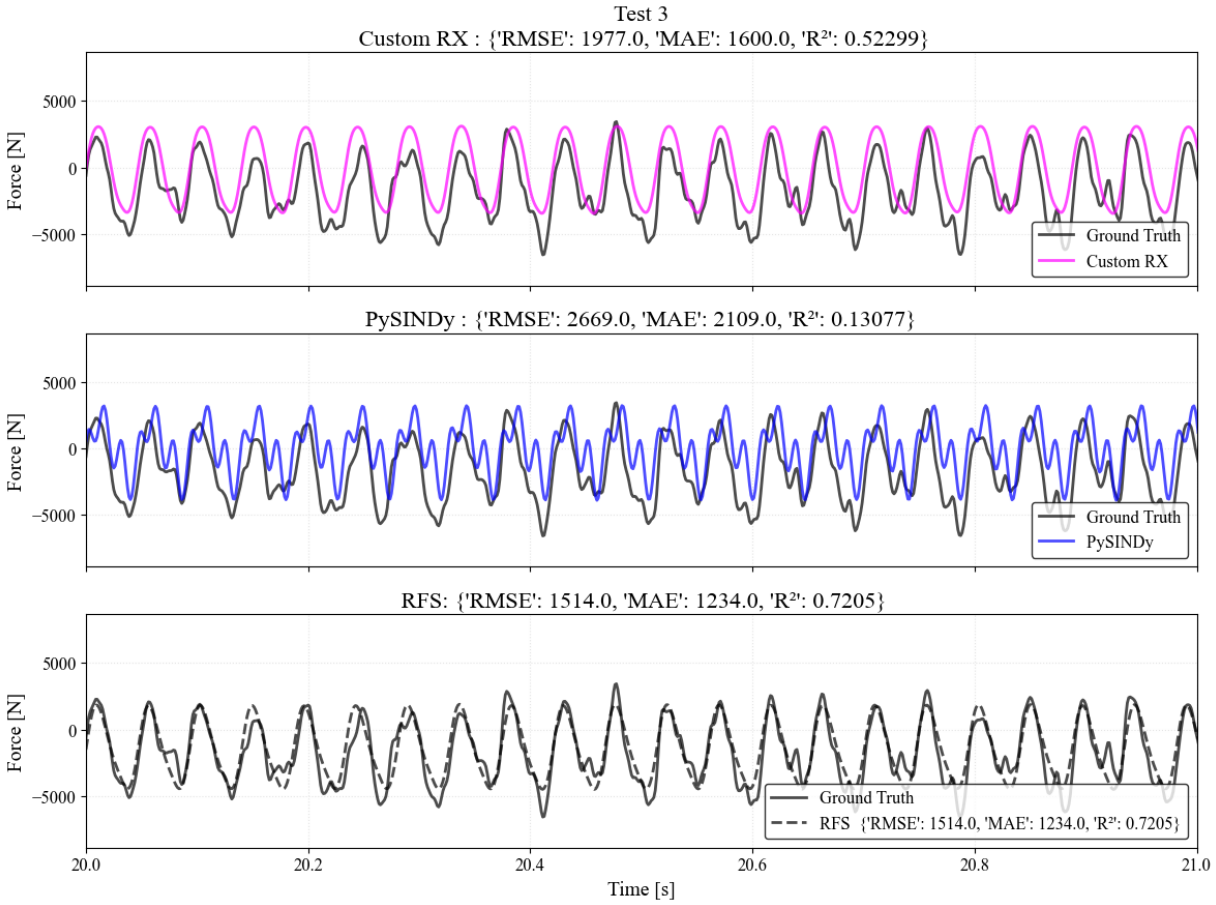


Figure 3.18: Shaft Force from Test 1 : Test 3 Data - Fit for t =2-40s

Then finally looking at a different time window of Test 3,and comparing statistics for a longer signal actually shows an improvement of fit, that may be due to an additional nonlinear component not present in Test 1 or just simply that Test 3 took longer to reach a steady state due to the lower frequency compared to Test 1.

3.2. Alternative Stable Equation of Motion

The previous expressions caused the initial value problem solvers to either get stuck or produce a simple exponentially growing response, which is clearly incorrect. As a result, further iterations of fitting were carried out, involving alternative functions (Chebyshev polynomials) and parameter adjustments, until a stable equation of motion was obtained as shown below:

Combined:

$$\begin{aligned}(f(t) - m\ddot{y})/m = {} & 39.48y^2\dot{y} + 3652.0y\dot{y}^3 + 682.4T(y,1) + 5.792 \cdot 10^4 T(y,2) + 1059.0T(y,3) \\ & + 3.478 \cdot 10^4 T(y,4) + 3247.0T(y,5) + 1.654 \cdot 10^4 T(y,6) + 2499.0T(y,7) \\ & + 3.971 \cdot 10^4 T(y,8) + 5767.0T(y,9) + 89.58T(\dot{y},1) + 29.41T(\dot{y},2) \\ & + 185.0T(\dot{y},3) + 37.34T(\dot{y},4) + 95.46T(\dot{y},5) + 13.98T(\dot{y},6) \\ & + 35.5T(\dot{y},7) + 3.001T(\dot{y},8) + 7.152T(\dot{y},9)\end{aligned}$$

Shaft:

$$\begin{aligned}F_{shaft} = {} & 31.5y^2\dot{y} + 24.32y\dot{y}^3 + 645.3T(y,1) + 4.89 \cdot 10^4 T(y,2) + 995.3T(y,3) \\ & + 3.411 \cdot 10^4 T(y,4) + 2927.0T(y,5) + 1.516 \cdot 10^4 T(y,6) + 2326.0T(y,7) \\ & + 2.994 \cdot 10^4 T(y,8) + 5210.0T(y,9) + 59.94T(\dot{y},1) + 29.41T(\dot{y},2) \\ & + 184.5T(\dot{y},3) + 16.53T(\dot{y},4) + 95.46T(\dot{y},5) + 9.004T(\dot{y},6) \\ & + 35.37T(\dot{y},7) + 1.733T(\dot{y},8) + 6.807T(\dot{y},9)\end{aligned}$$

Tip:

$$\begin{aligned}F_{Tip} = {} & 39.48y^2\dot{y} + 3652.0y\dot{y}^3 + 682.4T(y,1) + 5.792 \cdot 10^4 T(y,2) + 1059.0T(y,3) \\ & + 3.478 \cdot 10^4 T(y,4) + 3247.0T(y,5) + 1.654 \cdot 10^4 T(y,6) + 2499.0T(y,7) \\ & + 3.971 \cdot 10^4 T(y,8) + 5767.0T(y,9) + 89.58T(\dot{y},1) + 29.41T(\dot{y},2) \\ & + 185.0T(\dot{y},3) + 37.34T(\dot{y},4) + 95.46T(\dot{y},5) + 13.98T(\dot{y},6) \\ & + 35.5T(\dot{y},7) + 3.001T(\dot{y},8) + 7.152T(\dot{y},9)\end{aligned}$$

The fit performs worse than the previous expressions, however it is a stable version the current fit is a combination of Chebyshev polynomials in addition to cross terms of velocity & displacement, the cross terms seem to be the culprit for instability when trying to simulate our most fitting expression (see chapter 4) Thus this fit although lesser in accuracy in terms of RMSE and R^2 statistics it is in fact stable and can enable us to simulate responses without feeding in displacement/velocity. A validation simulation was performed, the expression was found by using the Data for test 1, thus the validation is made by applying the harmonic force from Test 3 (the slow tests are the most promising based on previous results) It is to be noted that Backwards Differentiation Formula (BDF) was used to compute the response, instead of the RK45 solver, and with using adaptive time step-size control (integration based on the rate of change of the system), the final results however was still not satisfactory as numerical noise was noticed (when changing the time step, the magnitude of the overall series does not change but the error does confirming the noise being numerical) thus a Savitzky-Golay Filter was used (see appendixK)), which helps smoothen the signal by windowing and fitting a polynomial along aforementioned windows.

The resulting restoring forces are as follows in figures 3.19and 3.20 below.

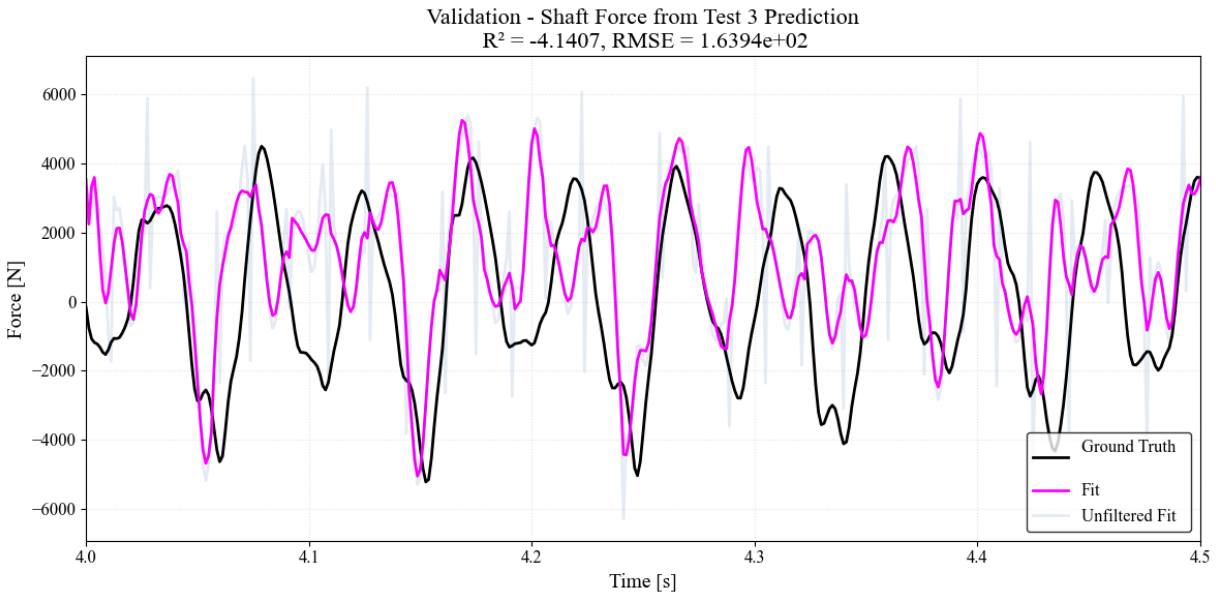


Figure 3.19: Simulated Shaft Force Cross Validation

The simulated shaft for the period (2-10sec) seems to be a really great fit in terms of RMSE, i.e. magnitude, however shape is off and not all peaks are maintained and that is not attributed to the noise filter, as a clear difference is seen without it.

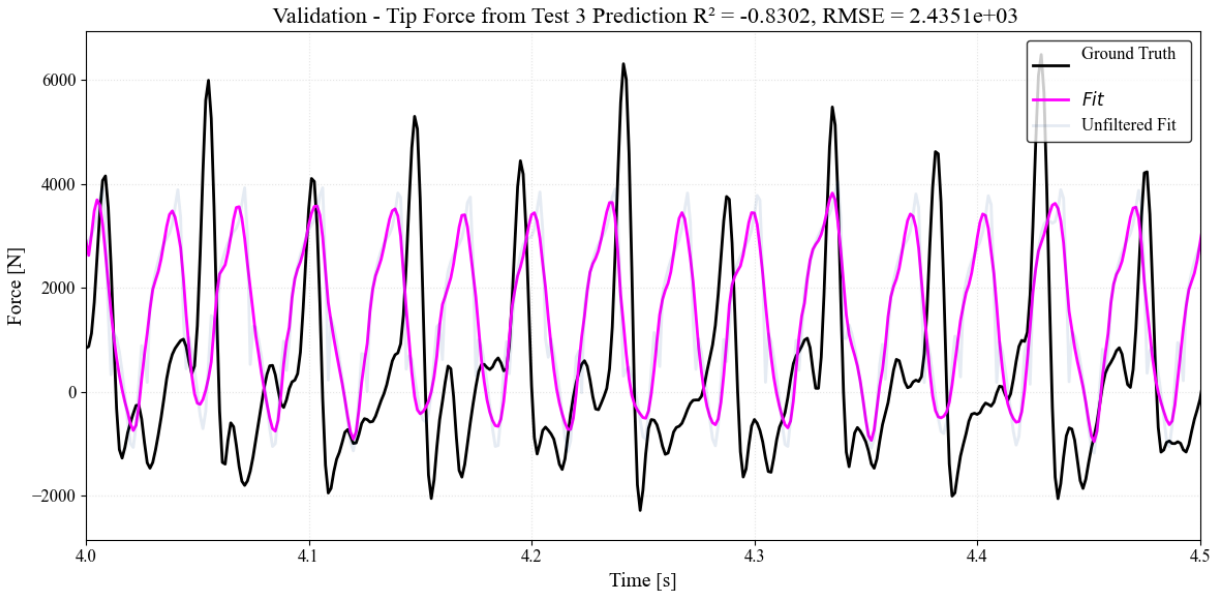


Figure 3.20: Simulated Tip Force Cross Validation

as for the Tip force, the fit is completely inadequate, neither frequency or amplitude was captured

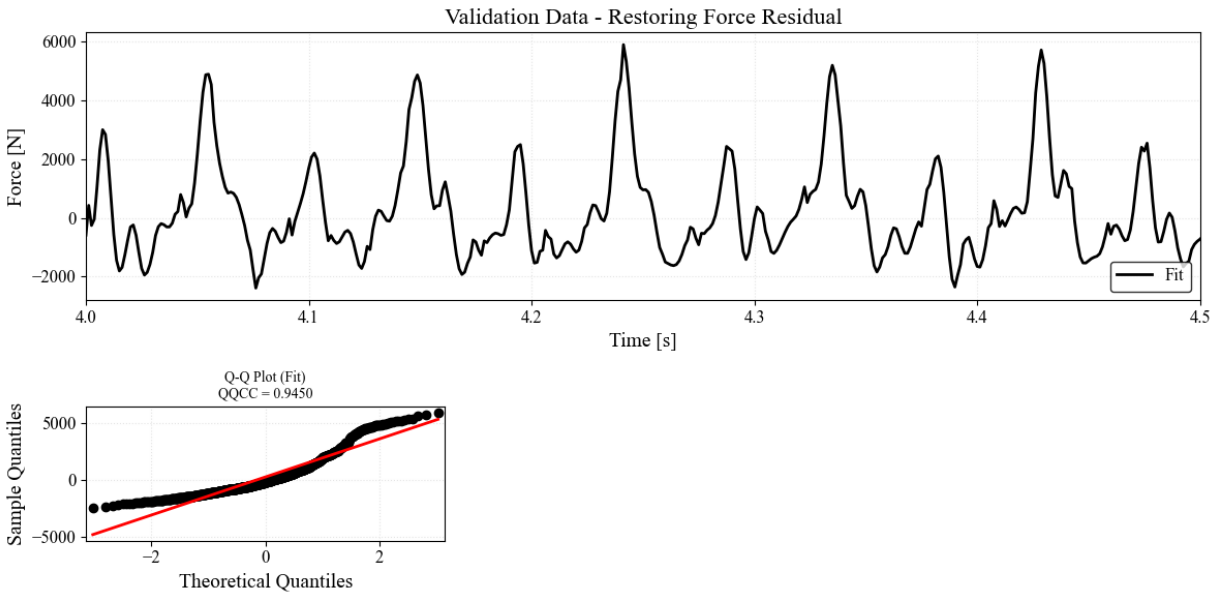


Figure 3.21: Simulated Pile Experiment - Residual

The residual shows a significant harmonic that's missed out by the fitted system and it not likely to be attributed to noise, additionally is not normally distributed as evident by the QQ plot.

for a clearer representation the loading cycles are shown in figures 3.22 and 3.23 below showing that the physics were not captured at all in those alternative equations of motion:

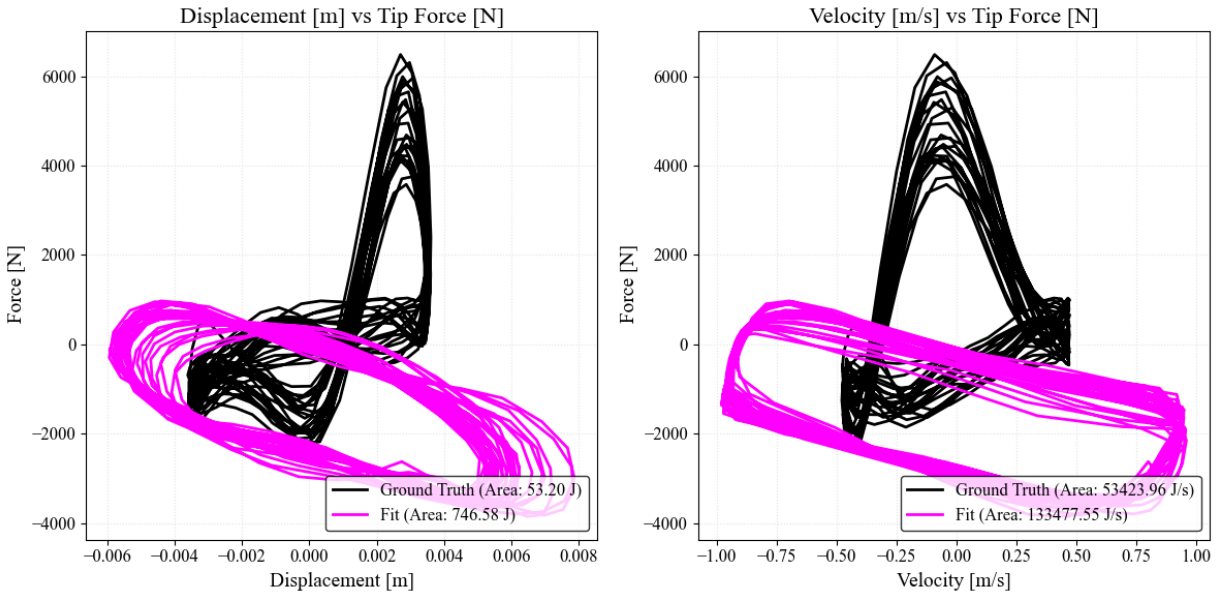


Figure 3.22: Simulated Pile Experiment : Tip Force - Loading Cycles

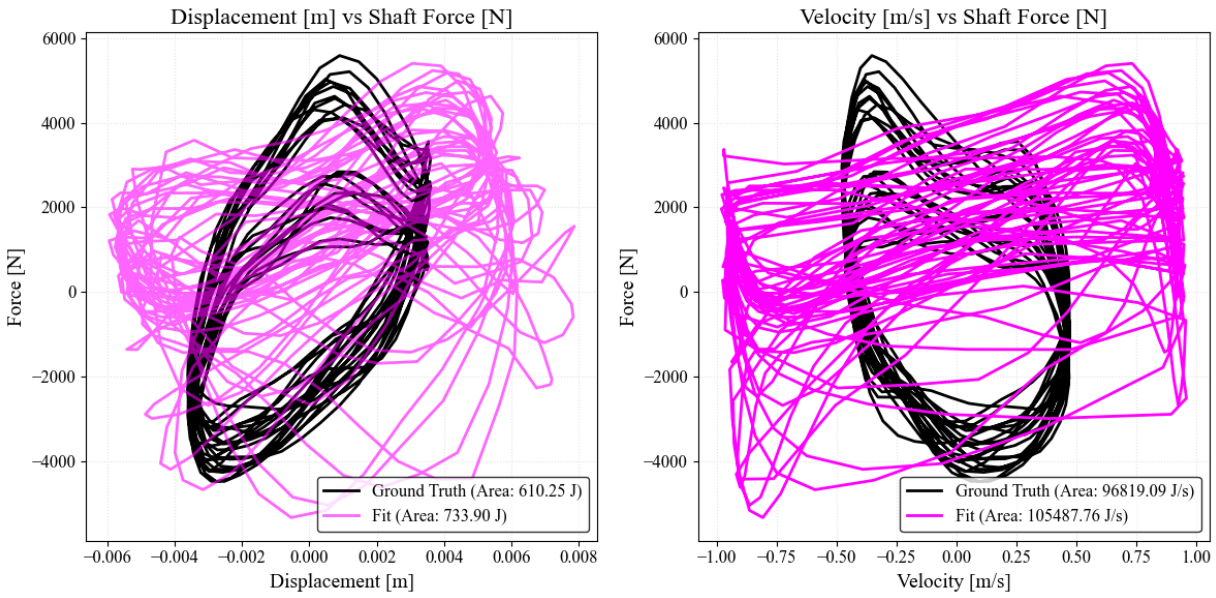


Figure 3.23: Simulated Pile Experiment : Shaft Force - Loading Cycles

Although the magnitude of the shaft force seems reasonable, the loading cycles are clearly not.

A note on simulating: when simulating the expressions found seem to be stable for many frequencies, however the most important criteria is that the system starts from a stable initial condition (taken from Test 1 used to fit after trimming the transient response at the start), and since the system was identified under the assumption it being time invariant, any region of the response can be simulated, as there history is not considered in the RX model.

The Validation case seems to be a failure in terms of overall response, besides the magnitude of the shaft contribution.

To observe more results from the stable formulation of shaft/tip resisting forces, 3 noise-free harmonics with an amplitude similar to Test 1's forcing were simulated, to analyse changes in response with frequency:

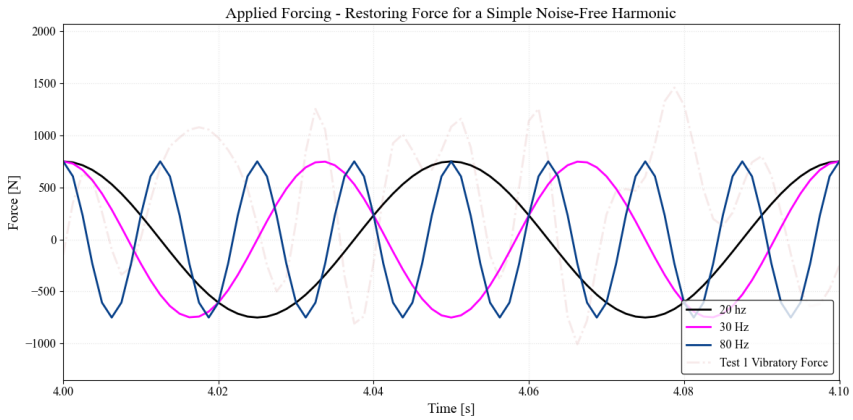


Figure 3.24: Simulated Experiments : Harmonics

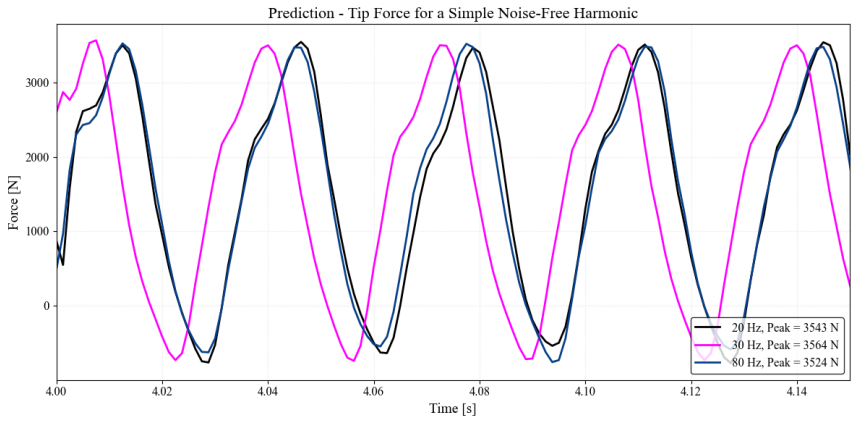


Figure 3.25: Simulated Experiments : Tip Forces

The tip forces seem to only show a time delay with no significant variation in magnitude, which is problematic, specially considering that the tip force for Test 1, (similar amplitude but with a frequency of around 33 Hz) shows higher harmonics besides a governing one, with a peak of around 4600 Hz. and since the Tip fit for Test 3, it's certain that the tip force being frequency independent is an error because of fit not an actual phenomena

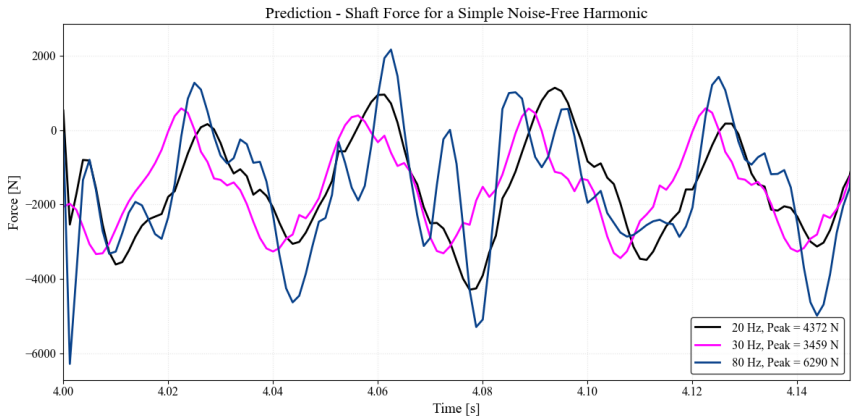


Figure 3.26: Simulated Experiments : Shaft Forces

The simulated shaft forces do indeed show higher harmonics (even after filtering), common sense would suggest that the resisting friction for really slow frequencies should be greater , thus 20Hz being higher makes sense, for higher harmonics one might expect it to be less, as the pile at extreme high frequencies can basically slide into the soil, however the resisting force at the higher frequency of 80Hz (which is already impractical to achieve in practice) does not indicate that in terms of having less shaft friction

4

Conclusions and Recommendations

The research work attempted to identify a surrogate model (equation of motion) for pile-soil interaction for the driving of a short and stiff pile (modelled as a SDOF) , as a first step towards that simpler 1-DOF non-linear systems were simulated and identified at first, the simpler cases provided insights but in the end however no running surrogate model was established as validation was not successful.

General Findings

- Using R^2 as an objective function instead of RMSE for the internal force is not effective since R^2 is sensitive to noise and outliers.
- Adding a penalty to the objective function related to difference of the average force proved helpful to prevent shift, it was simple addition with a scaler making it around 25% of the objective function next to the RMSE. Another penalty was tried for number/location of peaks however that did not prove to be useful.
- In parametric models (e.g., custom RX and pySINDy):
  - Adding a large number of candidate functions with implicit parameters is unproductive.
  - Using RFS to fit a rough guess of the shape of y and $\dot{y}$ dependencies helps identify key terms, such as:
    - Increases in force at high velocity and high displacement.
    - Changes in sign or symmetry of the response.
    - existence of cross products (e.g. increase of y with an increase in $\dot{y}$
- Chebyshev polynomials:
  - Are robust for representing many functions but lack cross-product terms (e.g., $y\dot{y}$).
  - Are not a universally optimal method for fitting any restoring force surface and need supplementing by getting a prior better idea whether cross terms are important.

Insights from Benchmark Cases

Case 1.0 (Duffing oscillator): Predictions Without Exact Equations

- The inability to find the exact equation of a system does not hinder accurate predictions.

Case 2.0 (Modified Duffing oscillator): Detecting Minute Components

- Components with minimal contributions remain hidden unless testing excites them sufficiently.
- Predictions rely heavily on the assumption that all possible phenomena and frequencies have been sufficiently excited. This includes nonlinear effects that depend on force magnitude.

Case 3.0 (SDOF with slip): Yield-Like Behaviours

- Simple polynomials fail to capture abrupt behaviours such as yielding.
- Alternative functions like exponentials or piecewise functions are better suited for such cases.

- Piecewise functions, however, can cause numerical instability in solvers. Continuous, smooth functions are preferable for stability and consistency with state-space formulations. Chebyshev polynomials are promising in simulating such behaviour and converge quickly, thus a good compromise between adding too many features and still getting the desired functions, as can be seen in 4.1

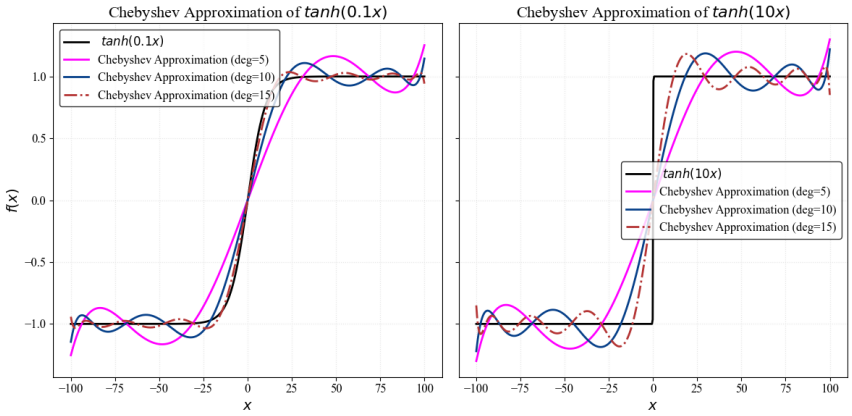


Figure 4.1: Chebyshev Approximation of A hyperbolic tangent

A Taylor expansion of 10 or 20 terms can not represent the desired plateau, but Chebyshev polynomials can to a very reasonable degree

Case 3.2 (SDOF with slip): Incorporating Constitutive Relationships as First Guess

- Introducing basic shapes based on constitutive relationships improves system fit.
- For example, the RX model identified destabilizing coefficients for the hyperbolic tangent, yet the combination of terms still produced the correct general shape.
- Fine-tuning implicit parameters adds complexity, particularly when data points are sparse near behavioural transitions.

Would it be wise to assume the outcomes of the simple benchmark cases applicable in the main problem? I would presume yes, many of the principles/guidance based on the benchmark cases proved useful in the pile-driving problem, as it's represented as a 1 DOF system, a different kind of response but the procedure remains the same.

Research questions revisited:

1. Can a non-linear dynamic system be identified with limited measurements? **Benchmarks clearly show that it is possible for simple systems, despite the limited range of excitation frequencies (only two were used) and gaps in the restoring force surface.**
2. Which identification methods are best suited in the context of the research problem? **Inconclusive, as no surrogate model was achieved.**
3. How is the system identification process physics-informed, and can more prior knowledge be included? **The formulation of the main problem as a regression problem (or similar) inherently includes prior knowledge, such as the general form of the equation of motion and the choice of unknown variables. Additional physics can be incorporated through the use of candidate functions, which may be inferred using the Restoring Force Surface (RFS) method.**
4. What criteria qualify an identified system's fidelity to actual physics? **Fidelity is determined by the ability to independently simulate new results with comparable shape (R^2 value) and magnitude (e.g., $RMSE$), as well as necessitates numerical stability. The equations of motion formulated in this work seems to fail this last and most important requirements.**
5. Would a time-invariant identified system produce reasonably close responses, to be later refined by incorporating time-variant components? **Inconclusive, as no surrogate model was achieved.**

Comparison between RFS & RX models (pySINDy or the custom built RX model)

In context of benchmark cases:
RFS is more intuitive, easier to visualize, however its nature as a polynomial expansion leads to it not having as much physical meaning, as well as producing models that are often numerically unstable, even for simple systems. where as RX has the advantages of being able to inform more physics by virtue of the user-specified candidate libraries.

In context of benchmark cases
A comparison in the context of the research problem can't be made as no functional model was made based on any of the methods, however it's worth noting that the RFS still tends to be likely to produce numerically unstable models. nevertheless the two approaches can be utilized hand in hand, the shape visualized by the restoring force surface can inform the user to possibly fruitful candidate functions.

Implications for Pile-Soil Interaction Identification

Various equations of reasonable fit were found for the resisting tip and shaft forces when driving piles for the slow tests (Test 1 and Test 3) with a reasonably good fit, however with a numerically unstable expression, a lesser fitting stable formulation was found however lacked the ability to accurately simulate a new response.

- The fast-driven pile tests did not provide useful data for determining an initial time-invariant equation of motion, as their resulting fits do not cross-validate well with each other. I hypothesize that the history dependence develops too rapidly for us to extract meaningful results.
- Analysing behaviour over individual time segments does not reveal the evolution of stiffness and energy dissipation components. While a single segment can serve as a starting point, the identification process must capture the full response as a whole. This means starting with time windows and progressively merging them to refine and improve a unified equation, rather than treating each segment separately.
- The time-invariant identification approach failed to produce reliable surrogate models for validation:
  - Multiple expressions fit shaft and tip forces well and cross-validate using measured displacement and velocities.However, simulations often lead to instability, producing exponential or non-harmonic responses.
  - Forcing stability constraints produces severely inaccurate expressions for shaft and friction forces, the only stable equation of motion (using Chebyshev polynomials to fit) results in a severely lacking validation making its simulations unreliable.

- Tip bearing seems to be asymmetric, and the time-variant dependence will likely to relate to the increase in downwards displacement as that's where the force increases, which is obvious as the further down you got the more you face resistance.
- Fitting multiple tests together proved unfruitful; however, some components, such as the linear stiffness, were easily observable in the dataset. How these components relate to frequency or the system's time variant behaviour remains to be explored.
- Occasionally, numerical instability occurs due to expressions overflowing , Chebyshev polynomials are powerful on this matter as you can go to a power of y^{10} while still not have overflow. cross products (can't be expressed as Chebyshev polynomials) are the most problematic in terms of overflow, thus were limited to a power of 3 or 4.

Possible Reasons for Failure in Surrogate Model Creation

- Fitting an expression for force over a time period involves the assumption that all its component are continuous in time and instability may arise because identified terms act conditionally rather than simultaneously.
- Lack of correlation between shaft and tip forces over time could hinder the identification process. Combining forces sacrifices the ability to distinguish components without yielding useful results. it's not realistic to assume the two forces do not co-relate, any give in shaft friction is counteracted by end bearing and vice-versa, but since the instrumentation allowed us to measure them both separately I deemed it wise to try to figure them out individually and then combine their models
- Not considering possibility of additional mass (e.g. soil around the pile moving with it in unison)
- Lack of data points in terms of frequency, i.e. more driving tests at different frequencies to excite different phenomena, but in terms of sampling the dt of 0.00125s appears sufficient when judging based on the benchmark cases.
- The fact that the tests included a fixed rate of lowering by the crane, we also have that motion into our system as another degree of freedom which was ignored.
- Uncertainty in measurements should be further investigated and see how it propagates to the model. however the measurement appear decently clean in terms of velocity and displacement, not so much for the noise in acceleration.

Bibliography

[1] S. A. Billings. *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. John Wiley & Sons, 2013.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 978-0-387-31073-2.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 978-0-521-83378-3. URL `https://web.stanford.edu/~boyd/cvxbook/`.

[4] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113. Edited by William Bialek, Princeton University, Princeton, NJ, and approved March 1, 2016 (received for review August 31, 2015).

[5] S. Chen and S. A. Billings. Representations of nonlinear systems: the narmax model. *International Journal of Control*, 49(3):1013–1032, 1989.

[6] Yinfeng Dong, Yingmin Li, Ming Lai, and Mingkui Xiao. Nonlinear structural response prediction based on support vector machines. *Computers & Structures*, 2007. Available online 8 November 2007.

[7] Russell Eberhart and James Kennedy. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE, 1995.

[8] Aditya Chandrashekhar Gondhalekar. *Strategies for Non-Linear System Identification*. Phd thesis, Imperial College London, University of London, October 2009.

[9] John H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[10] Ian T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002. ISBN 978-0-387-95442-4.

[11] Konstantin Savkov Kazakov. Dynamic response of a single degree of freedom (sdof) system in some special load cases, based on the duhamel integral. In *Proceedings of the International Conference on Engineering Optimization (EngOpt)*, Rio de Janeiro, Brazil, June 1–5 2008. VSU “L. Karavelov”, Sofia, Bulgaria, kazakov@vsu.bg.

[12] S. Khandelwal, J. Schoukens, and R. Tóth. Polynomial narmax modeling: A comparative study. *Automatica*, 121:109177, 2020.

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 (4598):671–680, 1983. doi: 10.1126/science.220.4598.671.

[14] Anastasis Kratsios. The universal approximation property. *Annals of Mathematics and Artificial Intelligence*, 89:435–469, 2021. doi: 10.1007/s10472-020-09723-1. URL `https://doi.org/10.1007/s10472-020-09723-1`. Accepted: 27 November 2020, Published: 22 January 2021, Issue Date: June 2021.

[15] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM, Philadelphia, PA, 2007.

[16] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 2nd edition, 1999.

[17] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[18] S. F. Masri, H. Sassi, and T. K. Caughey. Nonparametric identification of nearly arbitrary nonlinear systems. *Journal of Applied Mechanics*, 49(3):619–628, 1982. doi: 10.1115/1.3162537.

[19] S.F. Masri, A.G. Chassiakos, and T.K. Caughey. Identification of nonlinear dynamic systems using neural networks. *Journal of Applied Mechanics*, 1993. University of Southern California, California State University, California Institute of Technology.

[20] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[21] C. H. Menq and J. H. Griffin. A comparison of transient and steady state finite element analyses of the forced response of a frictionally damped beam. *Journal of Vibration, Acoustics, Stress, and Reliability in Design*, 107:19–25, 1985.

[22] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[23] Marc Mézard, Giorgio Parisi, and Miguel Angel Virasoro. *Spin Glass Theory and Beyond*. Lecture Notes in Physics. World Scientific, 1987. ISBN 978-9971-5-0215-7.

[24] Oliver Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks, Fuzzy Models, and Gaussian Processes*. Springer, 2021.

[25] X. Niu, Z. Yang, Y. Chen, and Q. Li. Assessing differences in acoustic characteristics from impact and vibratory pile driving. *Journal of Environmental Management*, 327:116626, 2023. doi: 10.1016/j.jenvman.2023.116626.

[26] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006. ISBN 978-0-387-30303-1.

[27] J. P. Noël and G. Kerschen. Nonlinear system identification in structural dynamics: 10 more years of progress. *Mechanical Systems and Signal Processing*, 83:2–35, 2016. doi: 10.1016/j.ymssp.2016.07.008.

[28] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid Nawab. *Signals and Systems*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 1997. ISBN 9780138147570.

[29] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[30] ML Science. Bias-variance tradeoff, 2024. URL `www.ml-science.com/bias-variance-tradeoff`. Accessed: 2024-07-04.

[31] A. Tsetas, A. Tsouvalas, and A. V. Metrikine. A non-linear three-dimensional pile–soil model for vibratory pile installation in layered media. *International Journal of Solids and Structures*, 269:112202, 2023. doi: 10.1016/j.ijsolstr.2023.112202. URL `https://doi.org/10.1016/j.ijsolstr.2023.112202`.

[32] K. Worden and G. R. Tomlinson. *Nonlinearity in Structural Dynamics: Detection, Identification and Modelling*. Institute of Physics Publishing, Bristol, UK, 2001.

[33] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997. doi: 10.1145/279232.279236.

A

Best fitting formulations for shaft friction & end bearing forces

Listed below are the best formulations to represent the resisting forces during driving.

Test Type	Equation
Shaft Force 1 (Test 1 - Custom RX)	$-1.92 \cdot 10^4 y^2 \dot{y} - 5.163 \cdot 10^6 y^2 - 2944.0 y \dot{y}^2 + 1.822 \cdot 10^4 y \dot{y} + 4.324 \cdot 10^4 y + 50.68 \dot{y}^3 - 79.02 \dot{y}^2 - 204.1 \dot{y}$
Shaft Force 2 (Test 3 - pySINDy)	$-3.08 \cdot 10^7 y^2 \dot{y} - 5.174 \cdot 10^6 y^2 + 1.277 \cdot 10^5 y \dot{y}^2 + 1.879 \cdot 10^4 y \dot{y} + 4.323 \cdot 10^4 y - 613.1 \dot{y}^3 - 80.38 \dot{y}^2$
Tip Force 1 (Test 1 - Custom RX)	$-1.657 \cdot 10^4 y^3 + 1.474 \cdot 10^5 y^2 \dot{y} - 3.417 \cdot 10^4 y \dot{y}^2 - 3.427 \cdot 10^4 y \dot{y} - 1.763 \cdot 10^6 y + 76.62 \dot{y}^3 + 20.54 \dot{y} + 3.544 \cdot 10^6 \, \mathsf{f}_1(y, -0.0056, 2)$
Tip Force 2 (Test 3 - pySINDy)	$-2.348 \cdot 10^9 y^3 - 2.44 \cdot 10^6 y^2 \dot{y} + 6.529 \cdot 10^4 y - 257.9 \dot{y}^3 - 80.07 \dot{y}$
Shaft Force 3 (Test 1 - Custom RX)	$31.5 y^2 \dot{y} + 24.32 y \dot{y}^3 + 645.3 T(y,1) + 4.89 \cdot 10^4 T(y,2) + 995.3 T(y,3) + 3.411 \cdot 10^4 T(y,4) + 2927.0 T(y,5) + 1.516 \cdot 10^4 T(y,6) + 2326.0 T(y,7) + 2.994 \cdot 10^4 T(y,8) + 5210.0 T(y,9) + 59.94 T(\dot{y},1) + 29.41 T(\dot{y},2) + 184.5 T(\dot{y},3) + 16.53 T(\dot{y},4) + 95.46 T(\dot{y},5) + 9.004 T(\dot{y},6) + 35.37 T(\dot{y},7) + 1.733 T(\dot{y},8) + 6.807 T(\dot{y},9)$
Tip Force 3 (Test 1 - Custom RX)	$39.48 y^2 \dot{y} + 3652.0 y \dot{y}^3 + 682.4 T(y,1) + 5.792 \cdot 10^4 T(y,2) + 1059.0 T(y,3) + 3.478 \cdot 10^4 T(y,4) + 3247.0 T(y,5) + 1.654 \cdot 10^4 T(y,6) + 2499.0 T(y,7) + 3.971 \cdot 10^4 T(y,8) + 5767.0 T(y,9) + 89.58 T(\dot{y},1) + 29.41 T(\dot{y},2) + 185.0 T(\dot{y},3) + 37.34 T(\dot{y},4) + 95.46 T(\dot{y},5) + 13.98 T(\dot{y},6) + 35.5 T(\dot{y},7) + 3.001 T(\dot{y},8) + 7.152 T(\dot{y},9)$

Table A.1: Shaft and Tip Force Equations from Various Tests

note: refer to figure 3.5 for definition of the function f_1. for definition of Chebyshev polynomial refer to appendix D.

B

Duffing FRF via harmonic balance method

Frequency Response Function of the Duffing Oscillator using Harmonic Balance Method

The equation of motion for the Duffing oscillator is given by:

$$\ddot{y} + \delta\dot{y} + \alpha y + \beta y^3 = F_0 \cos(\omega t), \tag{B.1}$$

where:

- $y(t)$ is the displacement.
- δ is the damping coefficient.
- α is the linear stiffness coefficient.
- β is the nonlinear stiffness coefficient.
- F_0 is the amplitude of the external force.
- ω is the frequency of the external force.

Harmonic Balance Method

We assume a steady-state solution where the displacement $y(t)$ can be approximated by a single harmonic component:

$$y(t) \approx A \cos(\omega t). \tag{B.2}$$

The first and second derivatives of $y(t)$ are:

$$\dot{y}(t) = -A\omega \sin(\omega t), \tag{B.3}$$

$$\ddot{y}(t) = -A\omega^2 \cos(\omega t). \tag{B.4}$$

Substituting the assumed solution and its derivatives into the Duffing equation:

$$-A\omega^2 \cos(\omega t) + \delta(-A\omega \sin(\omega t)) + \alpha A \cos(\omega t) + \beta A^3 \cos^3(\omega t) = F_0 \cos(\omega t). \tag{B.5}$$

Using the trigonometric identity:

$$\cos^3(\omega t) = \frac{1}{4}\cos(3\omega t) + \frac{3}{4}\cos(\omega t),$$

we get:

$$\beta A^3 \cos^3(\omega t) = \beta A^3 \left(\frac{1}{4}\cos(3\omega t) + \frac{3}{4}\cos(\omega t)\right). \tag{B.6}$$

Since we are using the harmonic balance method, we focus on the fundamental harmonic $\cos(\omega t)$ and ignore the higher harmonic $\cos(3\omega t)$. Thus, the equation simplifies to:

$$-A\omega^2 \cos(\omega t) - \delta A\omega \sin(\omega t) + \alpha A \cos(\omega t) + \frac{3}{4}\beta A^3 \cos(\omega t) = F_0 \cos(\omega t). \tag{B.7}$$

To balance the harmonics, we equate the coefficients of $\cos(\omega t)$ on both sides:

$$\left(-A\omega^2 + \alpha A + \frac{3}{4}\beta A^3\right) = F_0. \tag{B.8}$$

The coefficients of $\sin(\omega t)$ give us the damping term:

$$\delta A\omega = 0. \tag{B.9}$$

The equation for the amplitude A in terms of the system parameters and forcing frequency is:

$$A = \frac{F_0}{\sqrt{\left(\alpha - \omega^2 + \frac{3}{4}\beta A^2\right)^2 + (\delta\omega)^2}}. \tag{B.10}$$

Since the amplitude A appears on both sides of the equation, this equation must generally be solved numerically for A given values of the parameters $\alpha, \beta, \delta, F_0$, and ω.

C

pySINDy library

PySINDy: Sparse Identification of Nonlinear Dynamical Systems

PySINDy is a Python library developed for the Sparse Identification of Nonlinear Dynamical Systems (SINDy) framework. It automates the discovery of governing equations for dynamical systems from data, extending the methodology introduced by Steven L. Brunton et al. (2016)[4]. This approach addresses challenges in nonlinear system identification by leveraging sparse regression techniques to select only the most significant terms in a potential library of functions.

Approach

The PySINDy framework assumes that the dynamics of a system can be expressed as:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}), \tag{C.1}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ represents the state variables, and $\mathbf{f}(\mathbf{x})$ is the vector of governing equations. The objective is to identify $\mathbf{f}(\mathbf{x})$ directly from observed data.

Data Representation and Library Construction

PySINDy constructs a library of candidate functions, $\Theta(\mathbf{x})$, from the observed data. This library may include polynomial, trigonometric, or other basis functions:

$$\Theta(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & \ldots & x_n & x_1^2 & x_1x_2 & \ldots & \sin(x_1) & \ldots \end{bmatrix}. \tag{C.2}$$

The dynamics are then approximated as:

$$\frac{d\mathbf{x}}{dt} = \Theta(\mathbf{x})\Xi, \tag{C.3}$$

where Ξ is a sparse matrix containing the coefficients of the active terms in the library.

Sparse Regression for Model Discovery

To identify the governing equations, PySINDy applies sparse regression to solve for Ξ:

$$\min_{\Xi} \|\Theta(\mathbf{x})\Xi - \frac{d\mathbf{x}}{dt}\|_2^2 + \lambda\|\Xi\|_1. \tag{C.4}$$

Here:

- $\|\Theta(\mathbf{x})\Xi - \frac{d\mathbf{x}}{dt}\|_2^2$ ensures that the identified model fits the data.
- $\lambda\|\Xi\|_1$ promotes sparsity, penalizing less significant terms.

PySINDy uses algorithms such as Sequentially Thresholded Least Squares (STLS) to iteratively refine Ξ by removing small coefficients.

Workflow of PySINDy

The workflow of PySINDy typically involves the following steps:

1. extbfData Collection: Collect time-series data of the state variables **x**.
2. extbfLibrary Construction: Build a library of candidate functions $\Theta(\mathbf{x})$.
3. extbfSparse Regression: Solve the sparse regression problem to identify the non-zero terms in Ξ.
4. extbfModel Validation: Validate the discovered model against the observed data.

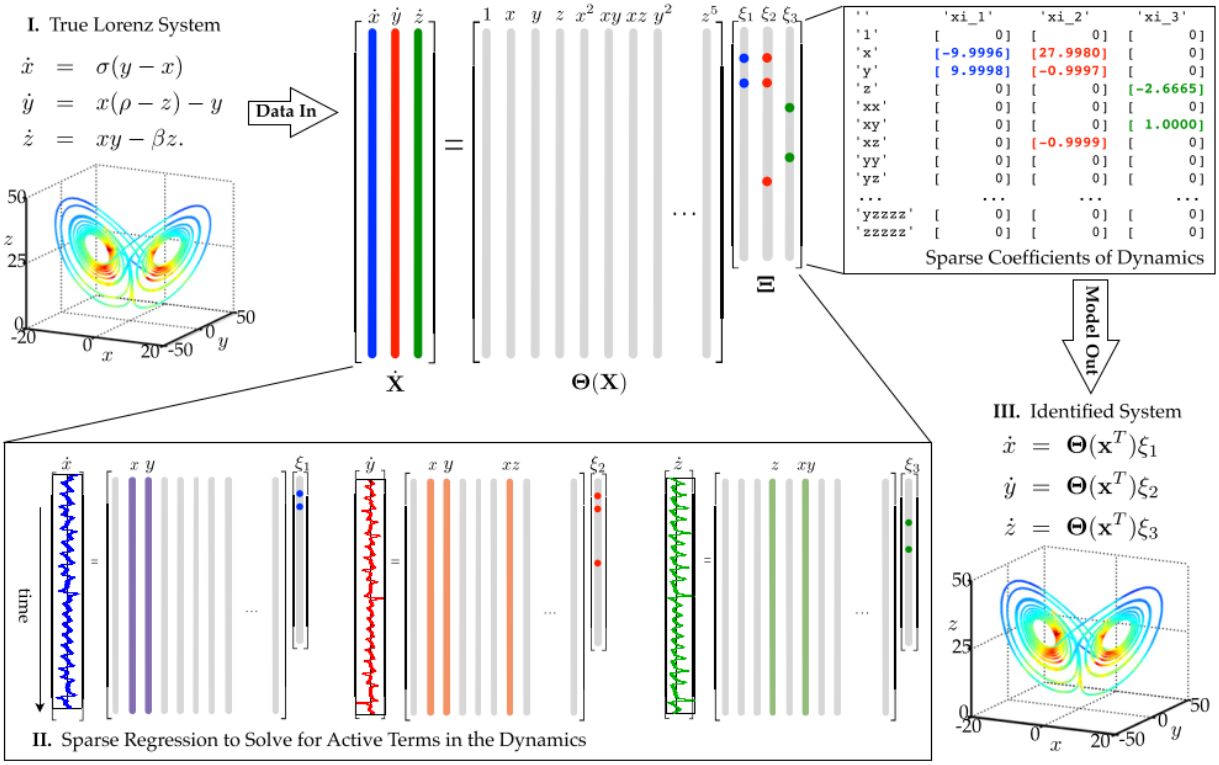


Figure C.1: Schematic of the SINDy algorithm, demonstrated on the Lorenz equations[4]

D

Chebyshev Polynomials

Definition of Chebyshev Polynomials

The Chebyshev polynomials of the first kind, $T_n(x)$, are defined recursively as:

$$T_0(x) = 1, \quad T_1(x) = x$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad \text{for } n \geq 1$$

Alternatively, they can be expressed explicitly as:

$$T_n(x) = \cos\left(n \arccos(x)\right)$$

for $x \in [-1, 1]$.

The Chebyshev polynomials form an orthogonal basis with respect to the weight function $w(x) = \frac{1}{\sqrt{1-x^2}}$ on the interval $[-1, 1]$:

$$\int_{-1}^{1} T_m(x)T_n(x)w(x)\,dx = \begin{cases} 0, & m \neq n \\ \frac{\pi}{2}, & m = n \neq 0 \\ \pi, & m = n = 0 \end{cases}$$

One of the key properties of Chebyshev polynomials is their rapid convergence when used for approximation. When a function is expanded in terms of Chebyshev polynomials, the resulting series converges uniformly on the interval [−1,1][-1, 1][−1,1], provided the function is sufficiently smooth. This convergence is due to the minimization of the polynomial's maximum deviation from zero over the interval, a characteristic that arises from the equi-oscillatory behaviour of Chebyshev polynomials. Moreover, the roots of these polynomials are distributed as the extrema of the cosine function, which ensures optimal node placement for interpolation, reducing Runge's phenomenon and making them highly efficient for numerical integration and function approximation[18].

E

Regression Methods

Common Regression Methods

Least Squares Regression (LSR)

The Least Squares method minimizes the sum of squared residuals between observed values (y_i) and predicted values ($\hat{y}_i$):

$$\text{Objective: } \min_{\beta} \sum_{i=1}^{n} \left(F_i - \mathbf{x}_i^T \beta\right)^2, \tag{E.1}$$

where $\mathbf{x}_i$ represents the predictor variables, and β is the vector of coefficients.

This method is computationally efficient and effective when the predictors are not highly correlated. However, it is sensitive to multicollinearity and overfitting, particularly in high-dimensional datasets.

Lasso Regression

Lasso introduces an ℓ_1 regularization term, penalizing the absolute values of coefficients to encourage sparsity:

$$\text{Objective: } \min_{\beta} \left[\sum_{i=1}^{n} \left(F - \mathbf{x}_i^T \beta\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right]. \tag{E.2}$$

This approach is particularly suited for sparse systems, where only a subset of predictors significantly impacts the response. However, it can overly simplify models, missing critical dynamics in nonlinear systems.

Elastic Net Regression

Elastic Net combines ℓ_1 and ℓ_2 regularization terms to balance variable selection and shrinkage:

$$\text{Objective: } \min_{\beta} \left[\sum_{i=1}^{n} \left(F_i - \mathbf{x}_i^T \beta\right)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right]. \tag{E.3}$$

This method is robust against multicollinearity but requires tuning multiple hyperparameters, making it less practical for iterative model identification in real-time scenarios.

Bias-Variance Trade-off and Nonlinear ODEs

In nonlinear ODE identification, the trade-off between bias and variance is crucial for selecting a regression method:

- **High Variance:** Complex models like Elastic Net and Lasso may overfit noise in the data, leading to unstable predictions.
- **High Bias:** Excessive regularization can oversimplify models, introducing systematic errors. least squares is a good balance of both.

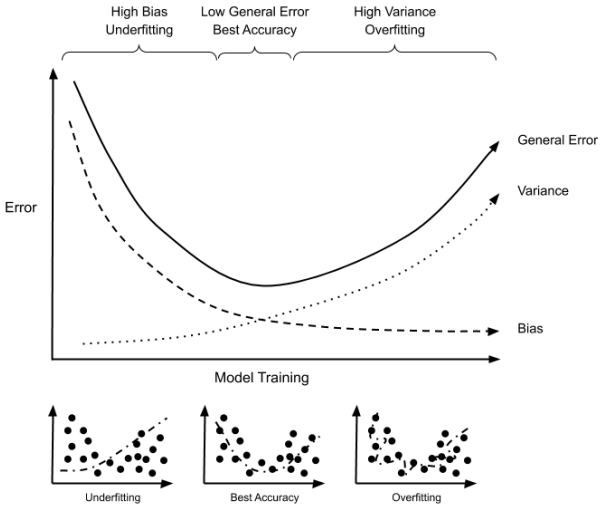


Figure E.1: Bias Variance Tradeoff [30]

Having an expansive library of functions to try and lasso's ability to encourage minimizing fit parameters match perfectly, thus was the method of choice for the RX model, pySINDy's Sparse detection (modified least squares with parameter elimination) is far superior and serves as comparison.

F

Statistics

RMSE (Root Mean Square Error)

Definition: RMSE is a metric that quantifies the average magnitude of errors between predicted and actual forces, expressed in the same units as the force itself. It is calculated as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where y_i is the observed force, $\hat{y}_i$ is the predicted force, and n is the total number of observations.

R^2 (Coefficient of Determination)

Definition: The R^2 statistic, also known as the coefficient of determination, measures the proportion of variance in observed forces that is explained by the model. It is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where y_i is the observed force, $\hat{y}_i$ is the predicted force, and $\bar{y}$ is the mean of the observed forces.

Usefulness: R^2 is effective when the goal is to understand how well the model captures *the variability in the forces*. This metric provides an overall measure of fit quality, indicating how well the model explains the variance across a range of values.

Quantile-Quantile (Q-Q) Plot

A Quantile-Quantile (Q-Q) Plot is a graphical technique used to assess if a dataset follows a particular theoretical distribution. It is commonly applied to check for normality, but it can be used for other distributions as well. Q-Q plots are useful in statistics and data analysis for evaluating the goodness of fit, and they help identify deviations from the expected distribution.

How Q-Q Plots Work

In a Q-Q plot:

- The **X-axis** represents the quantiles of the theoretical distribution (e.g., normal distribution) you are comparing your data to.
- The **Y-axis** represents the quantiles of the observed dataset.

The quantiles of the theoretical distribution are plotted against the quantiles of the data. Each point on the Q-Q plot represents a quantile pair.

Interpreting a Q-Q Plot

To interpret a Q-Q plot:

- **Straight Line**: If the data follows the theoretical distribution closely, the points will align approximately along a 45-degree line (often called the ”line of equality”). This indicates that the observed quantiles are similar to the theoretical quantiles.

- **Deviations from the Line**: Deviations from the line can indicate skewness, kurtosis, or other departures from the theoretical distribution.
  - Points that systematically curve upward or downward may indicate skewness.
  - Points that deviate towards the ends (tails) of the plot may suggest heavier or lighter tails than the theoretical distribution (kurtosis).

Dynamic Time Warping (DTW) Distance in Phase Portraits

The Dynamic Time Warping (DTW) distance provides a measure of similarity between two sequences, which in this context are the phase portraits of velocity vs. displacement. DTW aligns two sequences optimally, even if there are minor timing or phase shifts, allowing for a robust comparison of shape and structure.

Given two sequences, $(y_{\text{valid}}, v_{\text{valid}})$ (ground truth) and $(y_{\text{pred}}, v_{\text{pred}})$ (predicted model output), the DTW distance is computed as:

$$\text{DTW} = \sum_{i=1}^{N} d\left((y_{\text{valid}}(i), v_{\text{valid}}(i)), (y_{\text{pred}}(j), v_{\text{pred}}(j))\right)$$

where d is the Euclidean distance between points (y, v) in the two trajectories, and i and j are indices aligned according to DTW's optimal path.

G

Feature engineering of linear Systems and Principal component analysis (PCA)

Feature selection is a crucial step in machine learning and statistical modelling that helps improve model accuracy, reduce complexity, and prevent over-fitting. One widely used method for feature selection is Mutual Information (MI), an information-theoretic measure that quantifies the amount of information obtained about one random variable through another. In this context, MI measures how much knowing a particular feature reduces uncertainty about the target variable.

There are some universal feature estimators[14], such as Fixed-Shape Approximators, Neural Networks and Trees that don't require human interference, but they were not chosen as it's desirable to infer as well as learn as much of the physical aspects of the system instead of a black box approach

Mutual Information

Mutual Information (MI) between two random variables X (a feature) and Y (the target variable) is defined as:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

where:

- $p(x,y)$ is the joint probability distribution of X and Y,
- $p(x)$ and $p(y)$ are the marginal distributions of X and Y,
- The logarithm measures the information gained when comparing the joint probability with the independent probabilities.

Mutual information quantifies the amount of information that X and Y share. If X and Y are independent, $p(x,y) = p(x)p(y)$, and the mutual information is zero, indicating no dependence between the feature and the target. A higher MI value indicates a stronger dependency between X and Y, suggesting that X provides useful information about Y.

Bounded Information Content

Mutual information measures the reduction in uncertainty about the target Y given knowledge of a feature X. The maximum value of MI is limited by the entropy of the variables. Since real-world data usually involve only moderate dependencies between features and targets, the shared information is often limited. If X only partially reduces uncertainty about Y, the MI value will remain below the maximum entropy.

Entropy of Variables Is Usually Low

The maximum possible value of MI is the minimum of the entropies of the variables:

$$I(X;Y) \leq \min(H(X), H(Y))$$

In practice, features and target variables typically have **low entropy**, especially when they are well-structured or highly predictable. This limits the maximum MI value that can be achieved.

In the real world, features rarely have a perfect relationship with the target. Mutual information is maximized when X and Y are perfectly dependent (i.e., knowing X completely determines Y). However, in most practical scenarios, there is some degree of noise or randomness in the data, reducing the MI value.

Mutual information for continuous variables often requires estimating the joint probability distribution, which can be challenging, particularly in high-dimensional data. These estimation difficulties can result in lower MI values, even when the underlying relationship is stronger.

Since mutual information is measured in bits (if using log base 2) or nats (if using log base e), the logarithmic scale limits the growth of MI values. Even strong relationships between variables might result in only a few bits or nats of shared information, leading to MI values typically below 2.

Mutual Information for Feature Selection

In feature selection, mutual information is used to quantify the relevance of a feature X_i to the target Y. The general process involves the following steps:

1. **Calculate Mutual Information**: Compute the mutual information $I(X_i; Y)$ between each feature X_i and the target variable Y.
2. **Rank Features**: Rank the features based on their mutual information values in descending order. Features with higher mutual information values are considered more relevant to predicting the target.
3. **Select Top k Features**: Select the top k features with the highest mutual information for use in the model.

The primary advantage of using mutual information in feature selection is that it captures both linear and nonlinear relationships between the feature and the target, which makes it more robust than methods such as Pearson correlation that only capture linear dependencies.

KNN can be used to estimate the marginal and joint PDFs for continuous variables. The KNN algorithm looks at the distances between points in the dataset to estimate the local density around each point. This technique is non-parametric and does not assume any specific form for the PDFs.

For a continuous variable X, KNN estimates the local density by looking at the distance to the k-nearest neighbors. Points in regions with higher density will have closer neighbours, implying a higher probability density.

For two variables, X and Y, we treat them as a single joint space. The KNN algorithm measures the distances between points in this joint space to estimate the joint PDF $p(x, y)$. These joint distances help compute the probability density at each point based on how many close neighbours it has.

Then using KNN to compute mutual information:

1. **Estimate Marginal PDFs**: For each variable X and Y, KNN estimates the probability densities $p(x)$ and $p(y)$.
2. **Estimate Joint PDF**: KNN computes the joint probability density $p(x, y)$ in the combined feature-target space.
3. **Compute MI**: Plug the estimated PDFs into the mutual information formula.

Spearman's Correlation and Kendall's Tau

Spearman's Rank Correlation Coefficient, denoted as ρ, is a non-parametric measure that assesses the strength and direction of a monotonic relationship between two variables. It is calculated using the formula:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where d_i is the difference between the ranks of corresponding values in the two variables, and n is the number of observations. Spearman's correlation evaluates how well the relationship between two variables can be described using a monotonic function. It is less sensitive to outliers compared to Pearson's correlation.

Kendall's Tau, denoted as τ, measures the association between two variables by comparing the concordance and discordance of rank-ordered pairs. It is given by:

$$\tau = \frac{C - D}{\frac{1}{2}n(n-1)}$$

where C is the number of concordant pairs, D is the number of discordant pairs, and n is the total number of observations. A pair is concordant if the ranks of both variables agree (i.e., both increase or both decrease), and discordant if the ranks disagree. Kendall's τ is particularly robust for small sample sizes and datasets with tied ranks.

a simple example can be made to test those coefficients ability to distringuish features or components of a response, a simple composite signal made of the simple summation of some of the candidates was made (based on a harmonic response representing displacement or displacement and velocity).

In this analysis, the components of the harmonic signal are defined as follows:

- **Primary Signal Component (y_1):**

  $$y_1 = \cos(10t) + \sin(10t)$$

  This represents a harmonic signal composed of cosine and sine terms with a frequency factor of 10. It serves as the baseline signal in the analysis.

- **Secondary Signal Component (y_2):**

  $$y_2 = -\sin(10t) + \cos(10t)$$

  This is another harmonic signal derived from the same frequency as y_1, but with phase and amplitude adjustments. It interacts with y_1 to generate additional signal components.

The two components, y_1 and y_2, are combined to produce higher-order interactions:

These combinations serve as features to evaluate their relationships with the target signal in terms of mutual information, Spearman's rank correlation, and Kendall's tau. The target signal is defined as:

$$\text{Target} = y_1 \cdot y_2 + y_1^2 \cdot y_2$$

Noise is added to the target to simulate real-world signal variability.

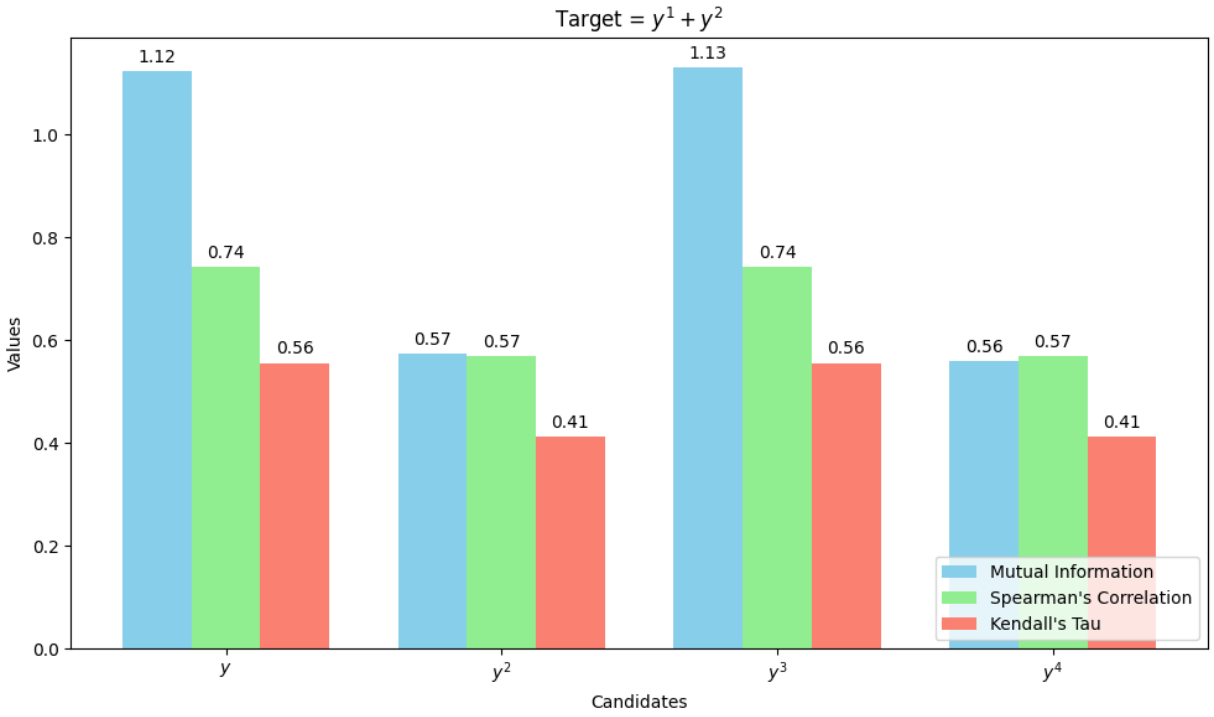


Figure G.1: Feature Selection via statistical coefficients - Example 1

Referring to figure G.1: it shows that for this particular signal, the MI approach as sees y and y^3 of the same significance, and likewise for y^2, y^4 and is not conclusive or helpful in picking candidates.

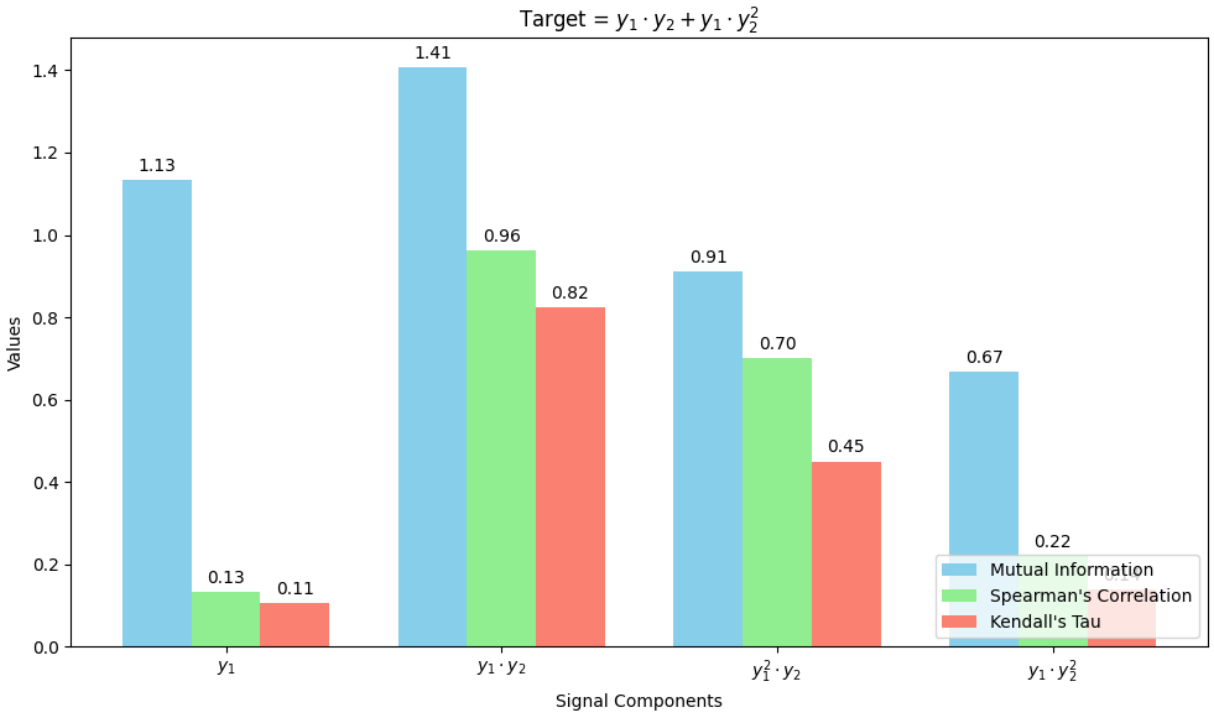


Figure G.2: Feature Selection via statistical coefficients - Example 2

Referring to figure G.2: it illustrates that all 3 approach are inconclusive, for some particular cases it

can clearly help with picking non-linear candidates however that does not work for all kinds of candidates and composite signals thus not reliable for nonlinear feature selection.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique widely used in data analysis and machine learning. It transforms a dataset with correlated features into a set of linearly uncorrelated variables called *principal components*, ordered by the amount of variance they explain.

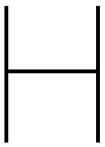
Process of PCA

1. **Standardization**: Center the data by subtracting the mean of each feature and, optionally, scale to unit variance [10].
2. **Covariance Matrix**: Compute the covariance matrix of the standardized data to capture relationships between features.
3. **Eigenvalues and Eigenvectors**: Decompose the covariance matrix to obtain eigenvalues and eigenvectors.
4. **Projection**: Project the data onto a subset of principal components, typically those that explain the majority of the variance [2].

Applications of PCA

- **Dimensionality Reduction**: Reduces the complexity of datasets while preserving as much variability as possible.
- **Noise Filtering**: Removes less significant components (low-variance directions), effectively reducing noise.
- **Visualization**: Facilitates data visualization by projecting high-dimensional data into 2D or 3D spaces.
- **Feature Extraction**: Identifies the most informative features for downstream machine learning tasks [10].

PCA is inherently a linear technique, as it assumes that the data can be represented as a linear combination of the principal components [10]. While PCA performs well for systems with linear dependencies, it may struggle to capture complex, nonlinear relationships within the data. For such cases, extensions like Kernel PCA [29] or other nonlinear dimensionality reduction techniques (e.g., t-SNE [17] or UMAP [20]) are often used to better handle the nonlinear structures in datasets. for the purpose of this research none of the dimensionality reduction approaches were used as it was not suitable for the problem (limited state variables) but for higher dimensional systems this would be a necessity.



Initial value problem (IVP) Solvers

Numerical solutions to ordinary differential equations (ODEs) are essential in various scientific and engineering applications. SciPy's {solve_ivp} function provides a range of solvers tailored for different problem types, including both stiff and non-stiff systems. Choosing the appropriate solver can significantly impact the efficiency and accuracy of computations.

This appendix provides an overview of the solvers available in {solve_ivp}, highlighting their strengths, limitations, and best-use scenarios. The discussion includes explicit and implicit methods, adaptive step-size approaches, and solvers designed specifically for stiff systems. Understanding these methods can help users make informed decisions when solving initial value problems (IVPs).

Overview of ODE Solvers in SciPy's `solve_ivp`

The `solve_ivp` function in SciPy offers several methods for solving initial value problems (IVPs) of ordinary differential equations (ODEs). Each method is suited to different types of problems, with variations in efficiency, accuracy, and suitability depending on factors like stiffness and precision requirements. Below is a summary of the most commonly used solvers, including `RK45`, along with insights on where each is best applied [15].

Solvers

RK45 (Default Method)

RK45 is an explicit Runge-Kutta method of order 5(4) that is best suited for non-stiff problems where moderate accuracy and speed are required. One of its main advantages is that it uses an adaptive step size to control the error, making it a versatile and general-purpose solver. However, RK45 can become inefficient or unstable when applied to stiff problems, which limits its effectiveness in such cases.

RK23

RK23 is an explicit Runge-Kutta method of order 3(2) that is typically used for non-stiff problems, particularly when faster computation is more important than high accuracy. Like RK45, it adapts its step size to control error, but it is generally faster for problems where high precision is not necessary. Its main drawback is that it offers lower accuracy than RK45 and is less efficient for solving complex or highly accurate solutions.

DOP853

DOP853 is an explicit Runge-Kutta method of order 8, ideal for non-stiff problems where very high accuracy is required. This method provides exceptional precision, making it efficient for problems that require high accuracy over long time intervals. However, it may be excessive for simpler problems where lower accuracy suffices, and it is not suitable for stiff problems.

LSODA

LSODA is unique in that it automatically switches between non-stiff and stiff methods, using the Adams method for non-stiff problems and the Backward Differentiation Formula (BDF) for stiff ones. This makes it an excellent choice for problems where the stiffness is unknown or varies over time. While it can efficiently

handle both stiff and non-stiff problems, it may be slower than methods designed specifically for one type of problem due to the overhead introduced by switching between solvers.

BDF (Backward Differentiation Formula)

BDF is an implicit multi-step method that is especially effective for stiff problems, where explicit methods fail or become inefficient. Its primary advantage is that it is highly stable and more efficient for stiff systems than explicit methods. However, BDF requires solving nonlinear systems at each step, which can be computationally expensive, and it is not efficient for non-stiff problems.

Radau

Radau is an implicit Runge-Kutta method of order 5, commonly used for stiff problems where high precision is needed. It is known for its stability in stiff systems and offers high accuracy. However, the computational cost of this method can be high due to its implicit nature, and it may be slower when applied to non-stiff problems.

Adams

Adams is an explicit multi-step method best suited for non-stiff problems. It is particularly efficient when solving non-stiff problems over long time spans. However, Adams is not suitable for stiff problems and can become unstable in such cases.

Optimization Techniques and Quasi-Newton Methods

Optimization with L-BFGS-B [33]

This appendix provides an overview of optimization techniques used for parameter estimation in internal force modelling. The primary method discussed is the Limited-memory Broyden-Fletcher-Goldfarb-Shanno with Box constraints (L-BFGS-B), a quasi-Newton optimization approach that efficiently handles large parameter spaces and bounded constraints. The objective function is formulated to minimize the discrepancy between predicted and observed internal forces, incorporating regularization to prevent over-fitting.

Additionally, alternative optimization methods are explored, including Stochastic Gradient Descent (SGD), Quasi-Newton methods, heuristic approaches such as Genetic Algorithms (GA) and Particle Swarm Optimization (PSO), and probabilistic techniques like Simulated Annealing and Monte Carlo methods.

Objective Function for Optimization

The optimization process minimizes the discrepancy between the predicted internal force and the observed internal force. This is achieved through the minimization of the mean squared error (MSE):

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left(F_{\text{int},i} - \sum_{j} c_j \phi_j(y, \dot{y}, \boldsymbol{\theta}) \right)^2, \tag{I.1}$$

where:

- $F_{\text{obs},i}$ is the observed internal force at sample i,
- $\phi_j(y, \dot{y}, \boldsymbol{\theta})$ represents features derived from displacement (y) and velocity ($\dot{y}$) with parameters $\boldsymbol{\theta}$,
- c_j are the coefficients corresponding to the features,
- N is the number of data points.

To prevent over-fitting and ensure the parameters remain physically meaningful, a regularization term can be added to the objective function. The regularized objective function becomes:

$$\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2^2, \tag{I.2}$$

where:

- λ is the regularization coefficient that controls the trade-off between the MSE and the magnitude of the parameters,
- $\|\boldsymbol{\theta}\|_2^2$ is the L_2 norm of the parameter vector.

The parameter vector $\boldsymbol{\theta}$ is optimized to minimize the objective function using the quasi-Newton method of L-BFGS-B [33].

The optimization of $\boldsymbol{\theta}$ is performed using the Limited-memory Broyden-Fletcher-Goldfarb-Shanno with Box constraints (L-BFGS-B) algorithm. This is a quasi-Newton optimization method that efficiently handles:

- large parameter spaces by using limited memory to approximate the Hessian matrix,
- bounded parameters, allowing for box constraints to ensure physically meaningful values.

Key Features of L-BFGS-B

1. Objective function minimization: The algorithm minimizes the objective function iteratively by updating the parameters $\boldsymbol{\theta}$ using gradient-based methods.
2. Box constraints: Enforces bounds on the parameters:

$$\theta_{\text{min}} \leq \theta_i \leq \theta_{\text{max}}. \tag{I.3}$$

3. Computational efficiency: L-BFGS-B avoids storing full Hessian matrices, making it suitable for high-dimensional problems.

Algorithm Steps

1. Initialization:

- Set initial parameter values, $\boldsymbol{\theta}_0$, typically chosen based on domain knowledge or heuristics.

2. Objective function evaluation:

- Compute the MSE or regularized MSE between the predicted and observed internal forces.

3. Gradient calculation:

- Approximate the gradient of $\mathcal{L}(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

4. Parameter update:

- Update parameters using a quasi-Newton approach, incorporating box constraints.

5. Convergence check:

- Stop when the objective function change is below a tolerance threshold or the maximum number of iterations is reached.

Applications in Features Model

For the Features Model, the optimized parameters $\boldsymbol{\theta}$ are used to:

1. refine the feature functions, $\phi_j(y, \dot{y}, \boldsymbol{\theta})$,
2. improve the accuracy of internal force predictions,
3. ensure the extracted features align with the physical properties of the system.

other optimization approaches are discussed although not used :

Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is an iterative method for optimizing the Lasso regression objective. It updates the coefficients by moving them in the direction that reduces the error, as determined by the gradient of the loss function . The update rule is:

$$\beta_j^{(t+1)} = \beta_j^{(t)} - \eta \left(\frac{\partial \mathcal{L}}{\partial \beta_j} + \alpha \cdot \text{sign}(\beta_j^{(t)}) \right)$$

where:

- η is the learning rate,
- $\mathcal{L}$ is the loss function, representing the prediction error,
- α controls the regularization term.

Several constraints are imposed on the coefficients during the optimization process:

- The first coefficient must be non-negative: $\beta_1 \geq 0$,
- The next three coefficients must be non-positive: $\beta_2, \beta_3, \beta_4 \leq 0$,
- The fourth coefficient must be less than or equal to the first coefficient: $\beta_4 \leq \beta_1$.

Quasi-Newton Methods

Quasi-Newton methods are optimization algorithms that approximate the Hessian matrix to efficiently find the minimum of a function. These methods are particularly effective for large-scale optimization problems [26]. The update rule for the coefficients can be expressed as:

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \boldsymbol{H}^{-1} \nabla \mathcal{L}(\boldsymbol{\beta}^{(t)})$$

where:

- $\boldsymbol{H}$ is an approximation to the Hessian matrix of second-order partial derivatives,
- $\nabla \mathcal{L}$ is the gradient of the loss function.

The same constraints on the coefficients, as described for SGD, are applied during the optimization process in Quasi-Newton methods [3].

Heuristic and Probabilistic Optimization Methods

Genetic Algorithms (GA)

Genetic Algorithms (GA) are optimization techniques inspired by the process of natural selection [9]. GAs use operations such as mutation, crossover, and selection to evolve a population of candidate solutions. This method is well-suited for non-convex and complex search spaces, especially when gradient information is unavailable.

Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a population-based heuristic optimization technique inspired by the social behavior of birds and fish [7]. PSO updates a swarm of candidate solutions (particles) based on their individual best positions and the global best position in the search space. It is commonly used for multidimensional and nonlinear optimization problems.

Crude Monte Carlo

Crude Monte Carlo methods are probabilistic techniques that rely on random sampling to approximate solutions to mathematical problems, particularly those involving integration or optimization [22]. In optimization, random samples are generated within the feasible space, and the best-performing sample is selected as the solution.

Puddle Jumping

Puddle Jumping is a heuristic method designed to escape local minima during optimization. It introduces random perturbations to the coefficients, enabling the process to avoid being trapped in suboptimal solutions [23]. The perturbation is applied as follows:

$$\beta_j = \beta_j + jump\_magnitude \times \text{rand}(-1, 1)$$

where:

- jump_magnitude controls the size of the random perturbation,
- rand$(-1, 1)$ generates a random number between -1 and 1.

After each perturbation, the same coefficient constraints as previously described are re-applied to ensure that the resulting coefficients remain within acceptable bounds.

Heuristic and Probabilistic Optimization Methods

Genetic Algorithms (GA)

Genetic Algorithms (GA) are optimization techniques inspired by the process of natural selection [9]. GAs use operations such as mutation, crossover, and selection to evolve a population of candidate solutions. This method is well-suited for non-convex and complex search spaces, especially when gradient information is unavailable.

Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a population-based heuristic optimization technique inspired by the social behavior of birds and fish [7]. PSO updates a swarm of candidate solutions (particles) based on their individual best positions and the global best position in the search space. It is commonly used for multidimensional and nonlinear optimization problems.

Simulated Annealing (SA)

Simulated Annealing (SA) is a probabilistic optimization algorithm inspired by the annealing process in metallurgy [13]. The method starts with a high ”temperature” that allows the algorithm to explore the search space freely, including accepting worse solutions. Over time, the temperature decreases, reducing the probability of accepting worse solutions, and focuses the search on improving the solution. SA is particularly effective for escaping local minima in non-convex optimization problems.

Crude Monte Carlo

Crude Monte Carlo methods are probabilistic techniques that rely on random sampling to approximate solutions to mathematical problems, particularly those involving integration or optimization [22]. In optimization, random samples are generated within the feasible space, and the best-performing sample is selected as the solution.

2 DOF Model Formulation

Model Formulation for a 2DOF System

Consider a 2 Degree of Freedom (2DOF) system with masses m_1 and m_2, damping coefficients c_1, c_2, and stiffnesses k_1, k_2. The equations of motion in matrix form are:

$$\begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix} \begin{pmatrix} \ddot{x}_1(t) \\ \ddot{x}_2(t) \end{pmatrix} + \begin{pmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 \end{pmatrix} \begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} + \begin{pmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} f_{unknown_1}(t) \\ f_{unknown_2}(t) \end{pmatrix} = \begin{pmatrix} F_{\text{ext1}}(t) \\ F_{\text{ext2}}(t) \end{pmatrix}.$$

Feature-Based Internal Force Reconstruction

The internal forces for each degree of freedom are reconstructed as a weighted sum of features:

$$F_{\text{int},1}(t) = \sum_i c_{1,i} f_{1,i}(x_1(t), \dot{x}_1(t), x_1(t) - x_2(t), \dot{x}_1(t) - \dot{x}_2(t)),$$
$$F_{\text{int},2}(t) = \sum_j c_{2,j} f_{2,j}(x_2(t), \dot{x}_2(t), x_2(t) - x_1(t), \dot{x}_2(t) - \dot{x}_1(t)).$$

Here: - $c_{1,i}$ and $c_{2,j}$ are the coefficients learned through regression, - $f_{1,i}(x_1(t), \dot{x}_1(t), x_1(t) - x_2(t), \dot{x}_1(t) - \dot{x}_2(t))$ and $f_{2,j}(x_2(t), \dot{x}_2(t), x_2(t) - x_1(t), \dot{x}_2(t) - \dot{x}_1(t))$ are the features, such as polynomials or trigonometric functions.

Modified Equations of Motion

Substituting the feature-based representation of internal forces into the equations of motion:

$$\begin{pmatrix} m_1 \ddot{x}_1(t) \\ m_2 \ddot{x}_2(t) \end{pmatrix} = \begin{pmatrix} \sum_i c_{1,i} f_{1,i}(x_1(t), \dot{x}_1(t), x_1(t) - x_2(t), \dot{x}_1(t) - \dot{x}_2(t)) \\ \sum_j c_{2,j} f_{2,j}(x_2(t), \dot{x}_2(t), x_2(t) - x_1(t), \dot{x}_2(t) - \dot{x}_1(t)) \end{pmatrix} + \begin{pmatrix} F_{\text{ext1}}(t) \\ F_{\text{ext2}}(t) \end{pmatrix}.$$

RX Model (Regression Formulation)

The internal force reconstruction can also be written :

$$\begin{bmatrix} F_{\text{int},1,t} \\ F_{\text{int},1,t+1} \\ F_{\text{int},1,t+2} \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & x_{1,t} & \dot{x}_{1,t} & x_{1,t} - x_{2,t} & \dot{x}_{1,t} - \dot{x}_{2,t} & f_{1,1}(x_{1,t}, \dot{x}_{1,t}, x_{1,t} - x_{2,t}, \dot{x}_{1,t} - \dot{x}_{2,t}) \\ 1 & x_{1,t+1} & \dot{x}_{1,t+1} & x_{1,t+1} - x_{2,t+1} & \dot{x}_{1,t+1} - \dot{x}_{2,t+1} & f_{1,1}(x_{1,t+1}, \dot{x}_{1,t+1}, x_{1,t+1} - x_{2,t+1}, \dot{x}_{1,t+1} - \dot{x}_{2,t+1}) \\ 1 & x_{1,t+2} & \dot{x}_{1,t+2} & x_{1,t+2} - x_{2,t+2} & \dot{x}_{1,t+2} - \dot{x}_{2,t+2} & f_{1,1}(x_{1,t+2}, \dot{x}_{1,t+2}, x_{1,t+2} - x_{2,t+2}, \dot{x}_{1,t+2} - \dot{x}_{2,t+2}) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$$\begin{bmatrix} F_{\text{int},2,t} \\ F_{\text{int},2,t+1} \\ F_{\text{int},2,t+2} \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & x_{2,t} & \dot{x}_{2,t} & x_{2,t} - x_{1,t} & \dot{x}_{2,t} - \dot{x}_{1,t} & f_{2,1}(x_{2,t}, \dot{x}_{2,t}, x_{2,t} - x_{1,t}, \dot{x}_{2,t} - \dot{x}_{1,t}) \\ 1 & x_{2,t+1} & \dot{x}_{2,t+1} & x_{2,t+1} - x_{1,t+1} & \dot{x}_{2,t+1} - \dot{x}_{1,t+1} & f_{2,1}(x_{2,t+1}, \dot{x}_{2,t+1}, x_{2,t+1} - x_{1,t+1}, \dot{x}_{2,t+1} - \dot{x}_{1,t+1}) \\ 1 & x_{2,t+2} & \dot{x}_{2,t+2} & x_{2,t+2} - x_{1,t+2} & \dot{x}_{2,t+2} - \dot{x}_{1,t+2} & f_{2,1}(x_{2,t+2}, \dot{x}_{2,t+2}, x_{2,t+2} - x_{1,t+2}, \dot{x}_{2,t+2} - \dot{x}_{1,t+2}) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

where you need to simultaneously solve for both internal forces. physical constraints as well as interaction between degrees of freedom is maintained via correct sign and addition/subtraction of the state variables. Prior knowledge of the system can be moved to the left hand side. and this formulation can be made into MDOF systems with or without measurements for the different degrees of freedom.

K

Savitzky-Golay Filter

Savitzky-Golay Filter: How It Works

The Savitzky-Golay filter is a data smoothing technique widely used to reduce noise in signals while preserving the shape and important features of the data, such as peaks and valleys. Unlike traditional moving average filters, which may blur sharp transitions, the Savitzky-Golay filter fits a polynomial to a sliding window of data points, providing a smooth approximation of the signal.

How It Works

The filter operates by:

- Selecting a **window length** (an odd number of data points) over which the signal is locally approximated.
- Fitting a **polynomial** of specified order (e.g., linear, quadratic, cubic) to the data within each window.
- Replacing the central point of the window with the value predicted by the polynomial fit, ensuring smooth transitions across the signal.

Key Parameters

- **Window Length:** Determines the number of points in the sliding window. Larger windows result in smoother signals but may oversimplify details.
- **Polynomial Order:** Defines the degree of the polynomial used for fitting. Higher orders can capture more complex trends but may overfit noisy data.

Applications

The Savitzky-Golay filter is ideal for:

- Smoothing experimental data to reduce high-frequency noise.
- Preserving sharp transitions in signals, such as in spectroscopy, ECG signals, and time series data.
- Preprocessing data for machine learning and feature extraction.

Example Usage

For a signal $y(t)$, the smoothed signal is obtained by sliding a window of size n and fitting a polynomial of order p to approximate the data:

$$\text{Smoothed value at time } t = \sum_{i=0}^{p} a_i t^i, \quad \text{where } t \in \text{window}.$$

Noise Magnification when Deriving Acceleration from Velocity or Displacement

Filtering Effects

Incorrectly applying derivative methods, especially in noisy data, can introduce unintended filtering effects. For example, a naive finite difference can amplify noise, leading to erroneous conclusions. Conversely, excessive smoothing may filter out genuine signal variations, obscuring critical dynamics. It is crucial to choose a derivative method that matches the characteristics of the data and the goals of the analysis. The identification problem assumes the internal force which uses acceleration calculated from measurement is sound and equations of motion derived follow that dependency.

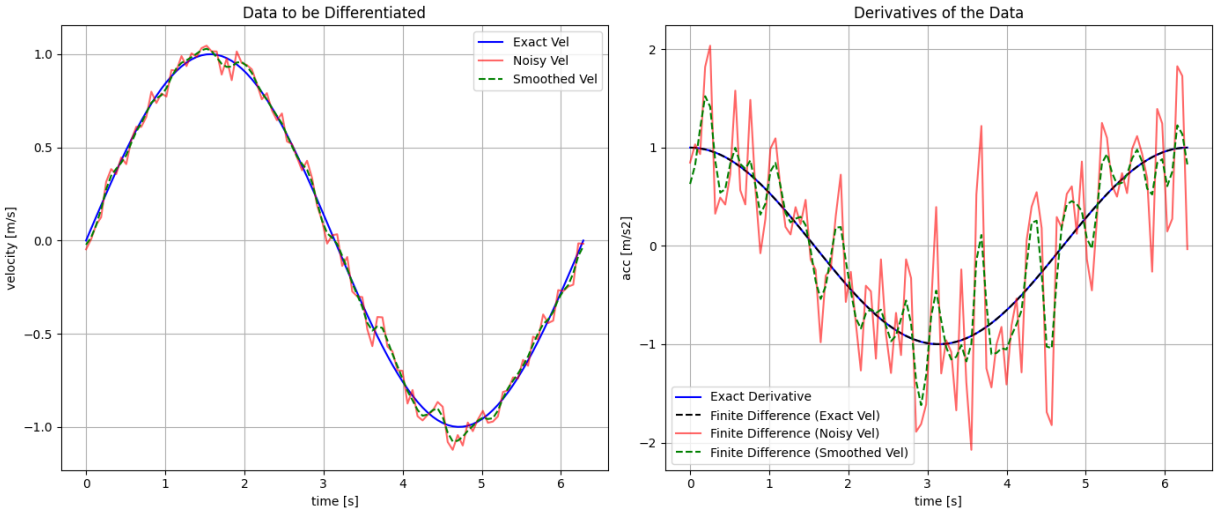


Figure L.1: Noise magnification when using finite difference to differentiate

refering to figure L.1 , it can be seen that a 5% noise in velocity can translate into huge error (sometimes over 100% difference) into acceleration when using finite difference, that effect can be reduced by smoothing the dataset, but will not disappear completely. some possible methods of differentiation with smoothening capabilities are discussed briefly below.

Methods

- **Finite Difference:** A simple approach where the derivative is approximated by the difference between consecutive data points. While easy to implement, it is highly sensitive to noise, which can lead to significant errors.

- **Savitzky-Golay Filter:** This method applies a polynomial smoothing to the data before differentiating, which reduces noise but requires careful selection of parameters like window length and polynomial order.
- **Fourier Transform:** Computes the derivative in the frequency domain. This approach smooths the signal but assumes periodicity and uniform sampling, making it unsuitable for non-stationary data.
- **Cubic Spline:** Fits a smooth spline to the data and computes its derivative. It is effective for unevenly spaced data but can be computationally intensive.
- **Total Variation Regularization:** Minimizes the total variation of the derivative, preserving edges while smoothing. This method is robust to noise but requires tuning of regularization parameters.
- **Higher-Order Finite Difference:** Utilizes more points around the target point to compute a more accurate derivative, though still sensitive to noise.
- **Smoothed Finite Difference (PySINDy):** Combines smoothing with finite differences to obtain a smooth derivative, balancing noise reduction and accuracy.

M

Extended Benchmark Cases

Important note The **Custom RX** model is referred to in figures here as **NARMAX**, it was in hopes of extending the identification process to included, but the **RX** name is more apt. thus was adopted elsewhere but was not amended here.

Plots for Case 2.0

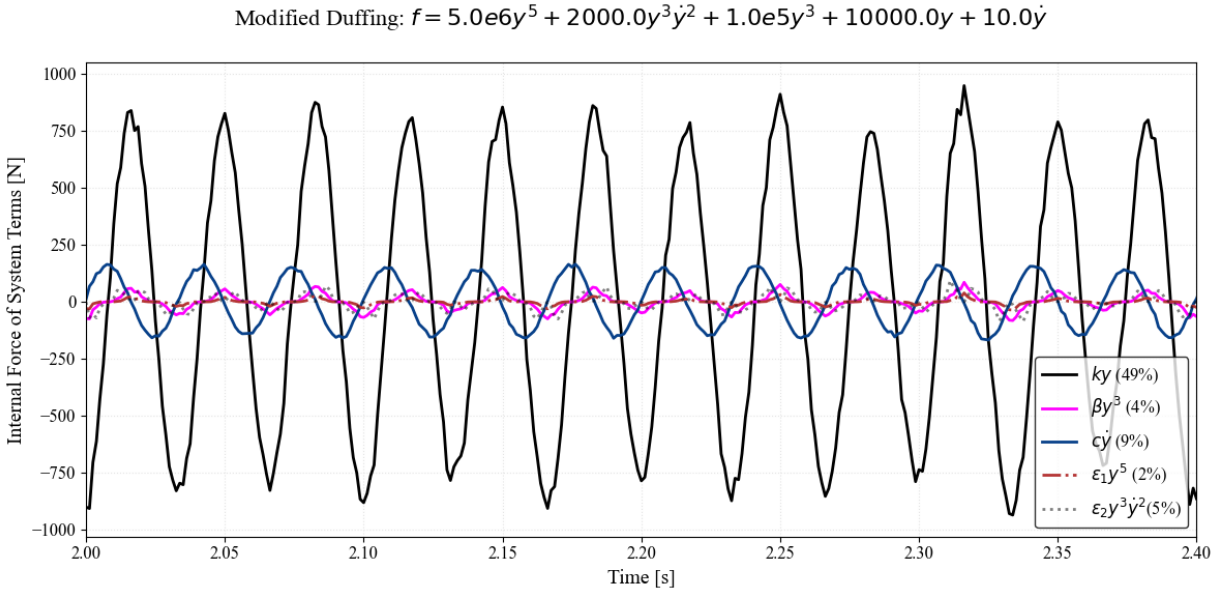


Figure M.1: Case 2.0 Components

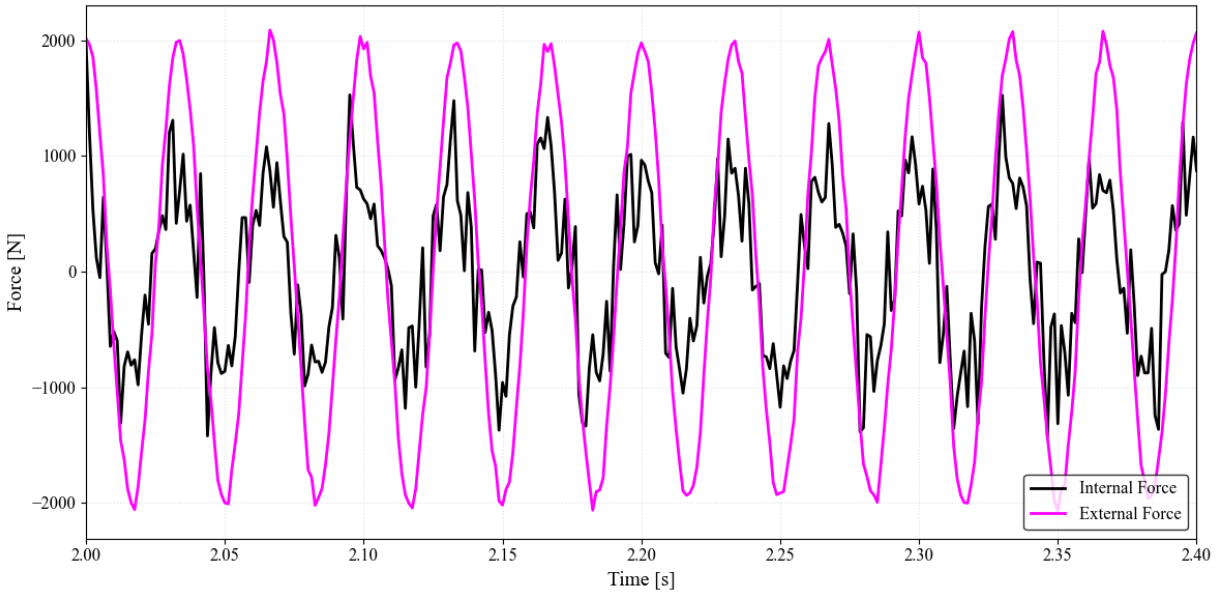


Figure M.2: Case 2.0 : Target Restoring Force

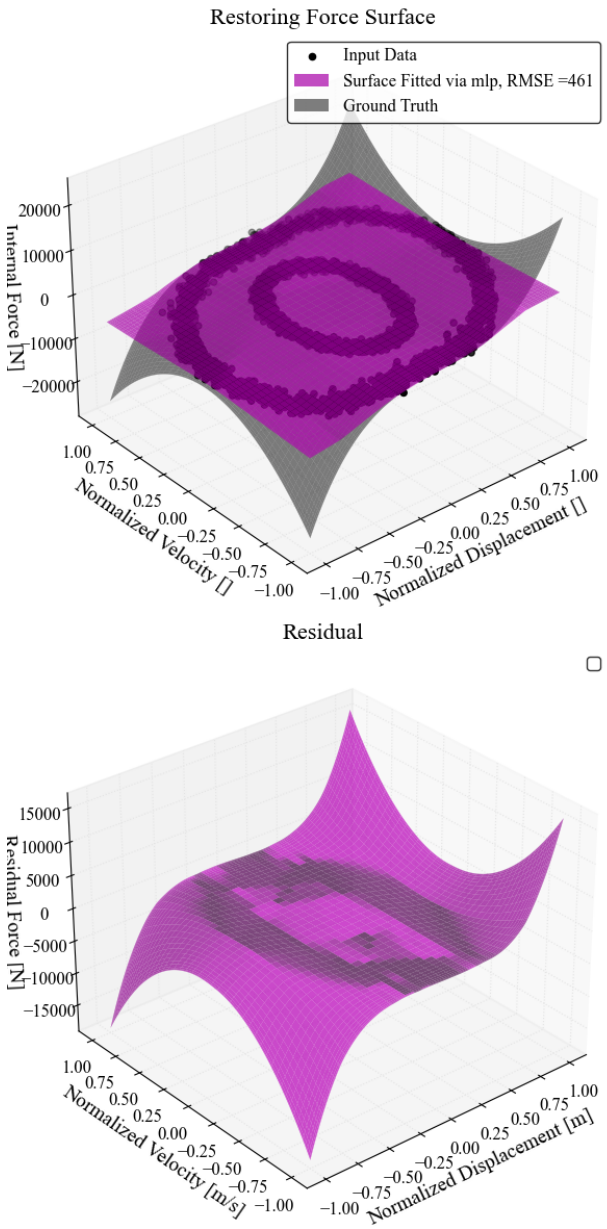


Figure M.3: Case 2.0: Selected Restoring Force Surface

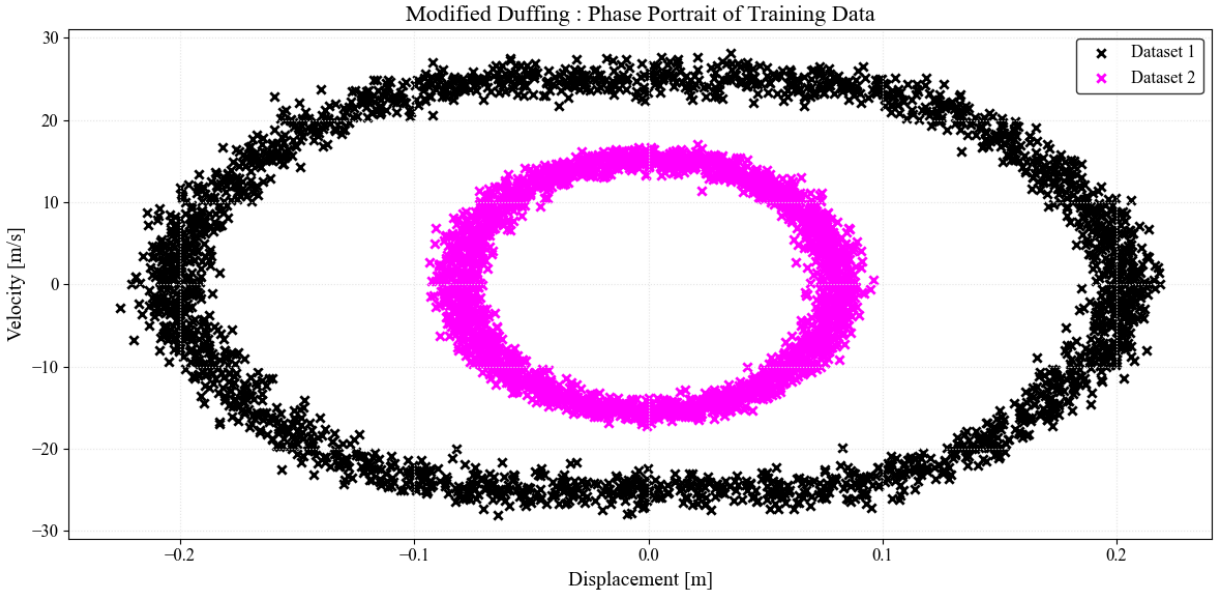


Figure M.4: Case 2.0 : Phase Portrait of Training Data

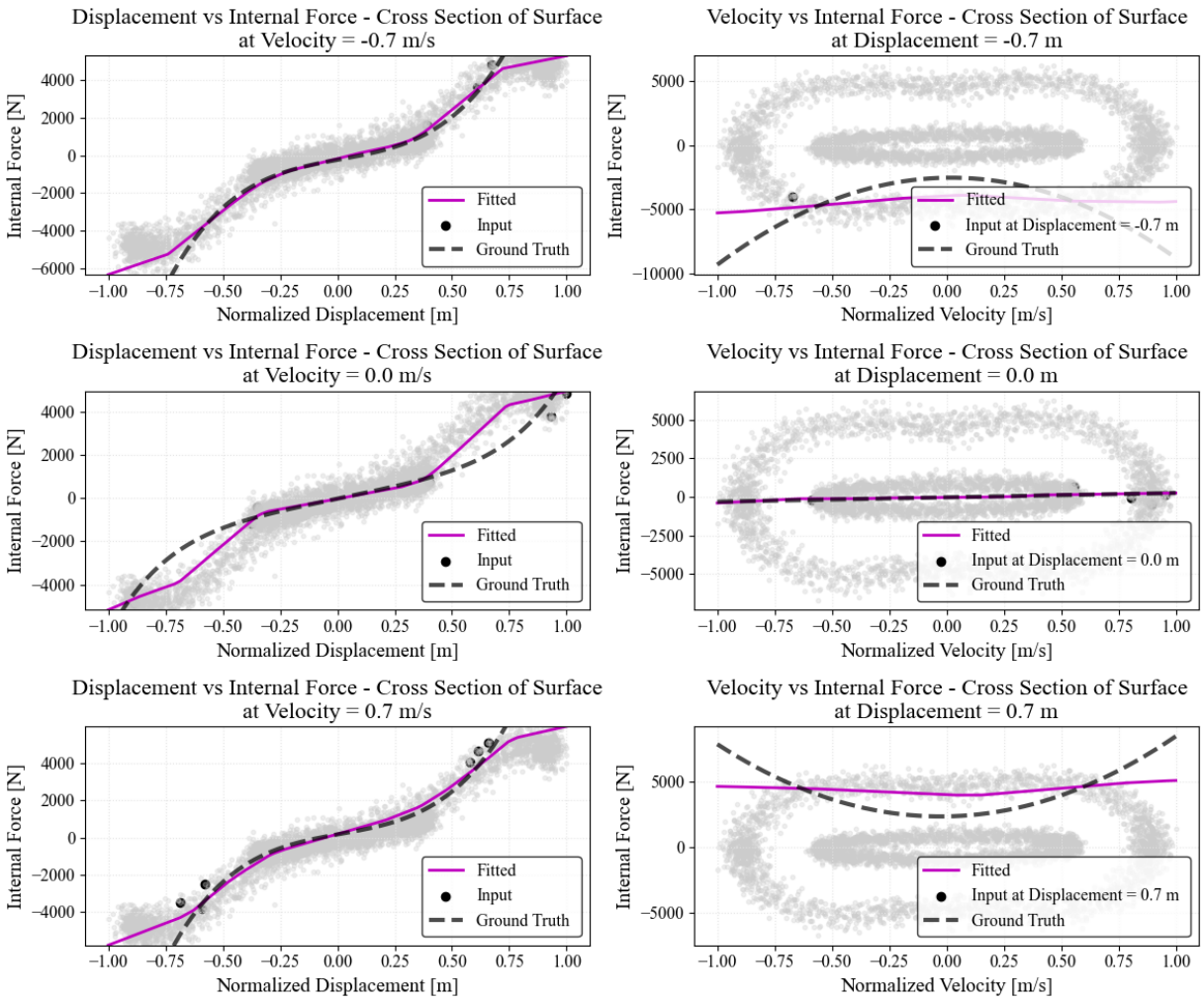


Figure M.5: Case 2.0 : Cross Sections of Surface

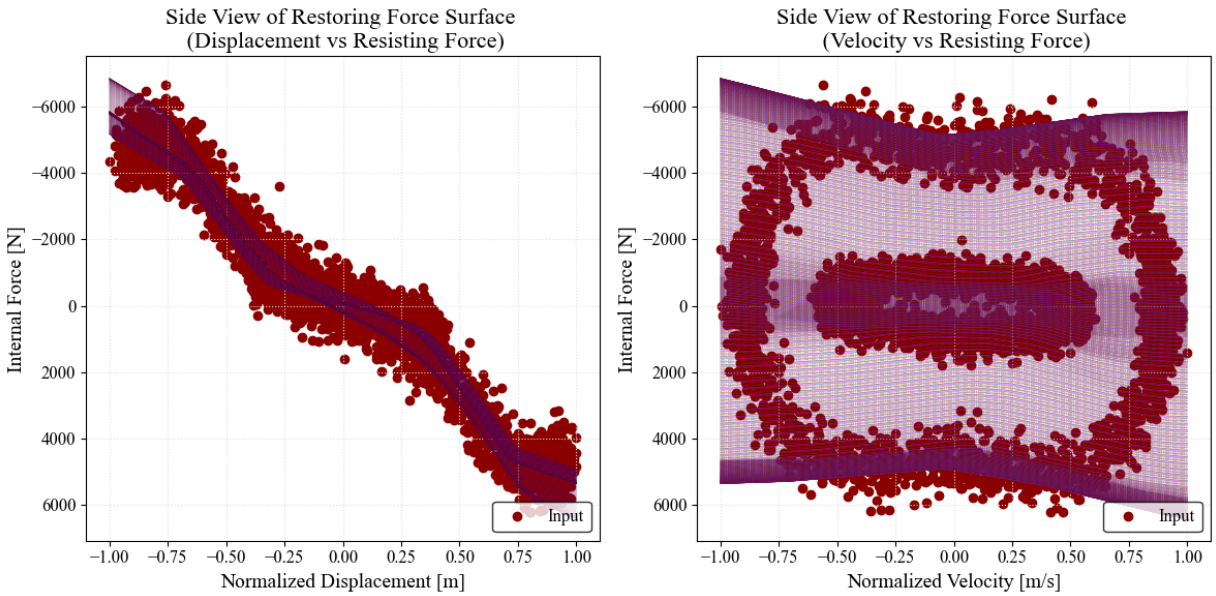


Figure M.6: Case 2.0 : Side Views of Surface

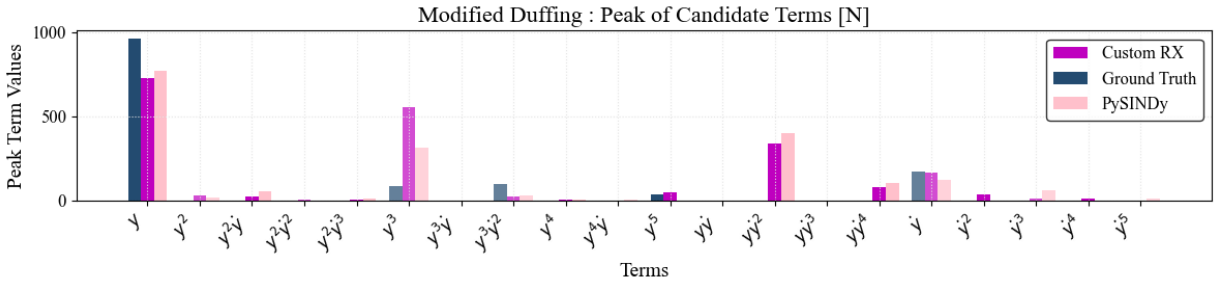


Figure M.7: Case 2.0 : Force Features comparison

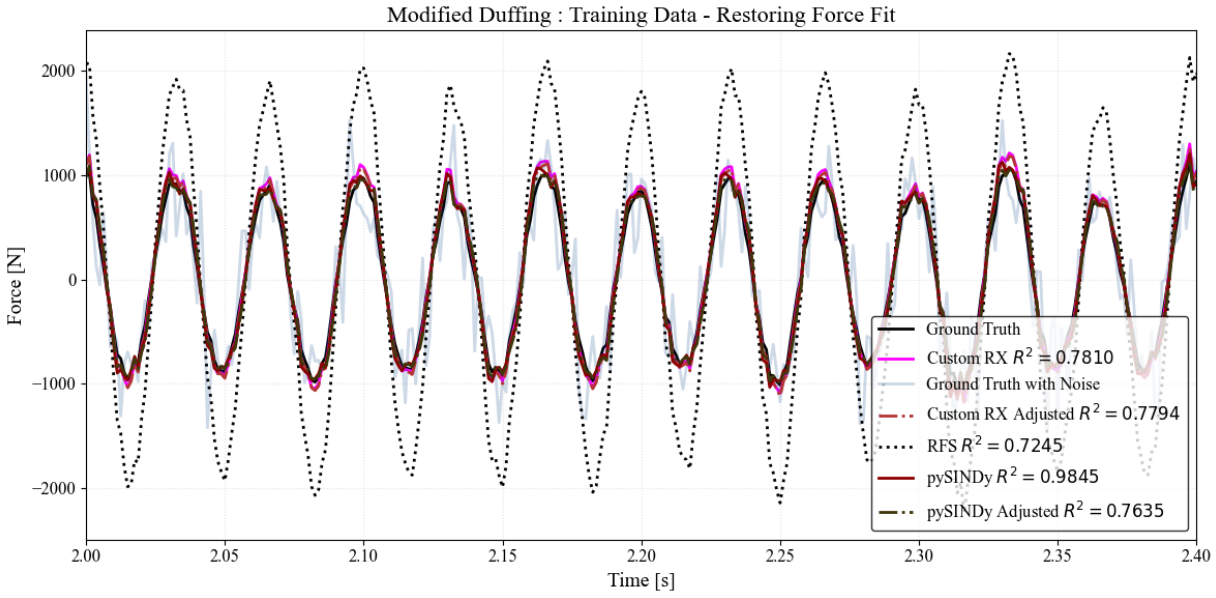


Figure M.8: Case 2.0 : Training Data - Restoring Force Fit

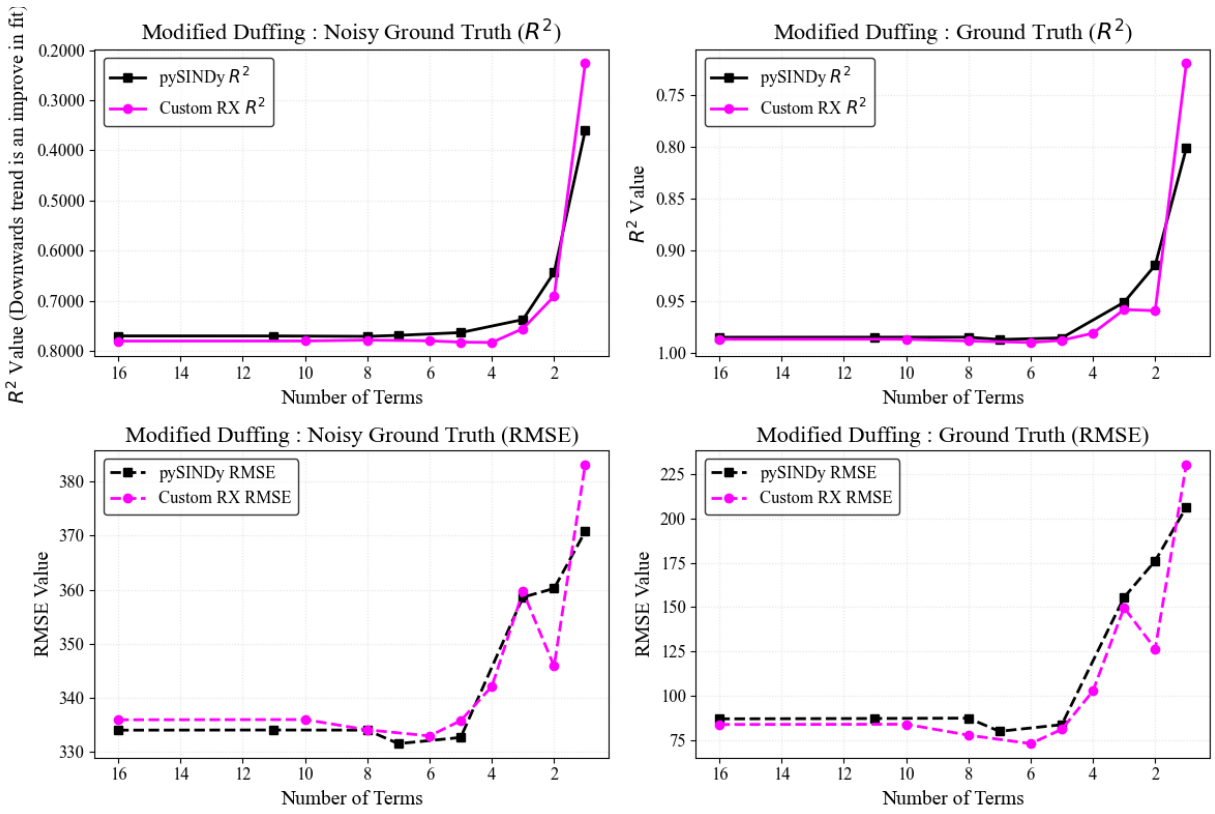


Figure M.9: Case 2.0 : Removing Unnecessary Terms

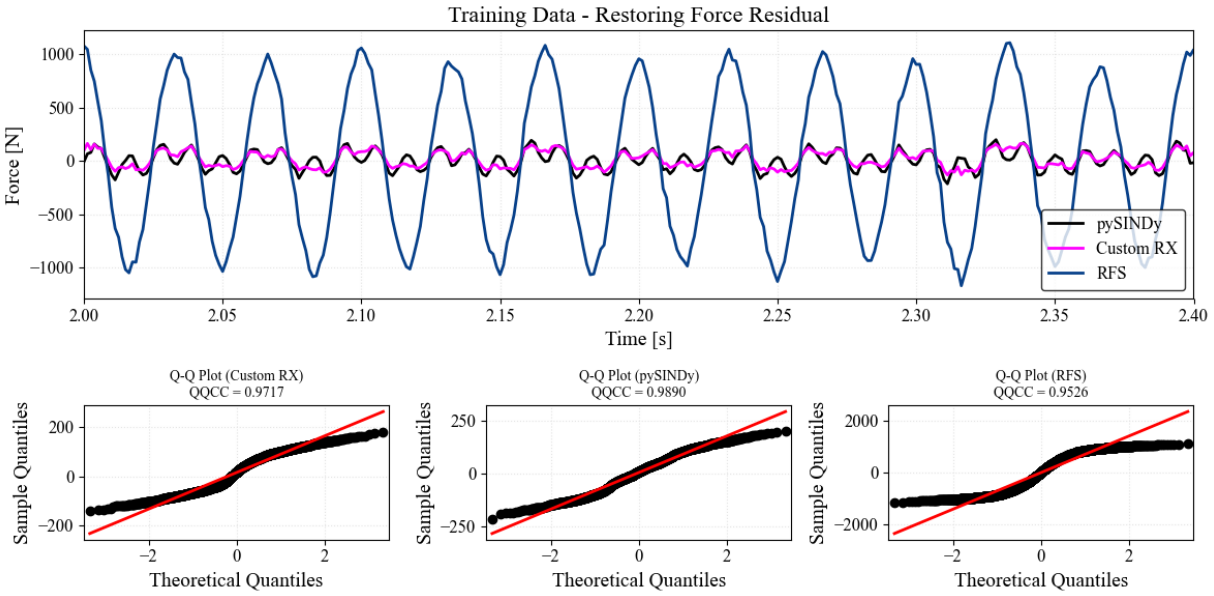
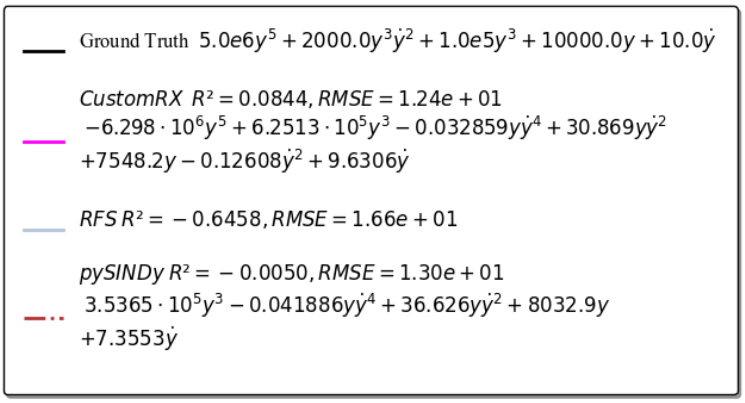


Figure M.10: Case 2.0 : Training Data - Restoring Force Residual

the QQCC value tends to be logarithmic, shape of the QQ plot is a better indicator, all 3 fits seems to have not properly isolated the normally distributed noise.



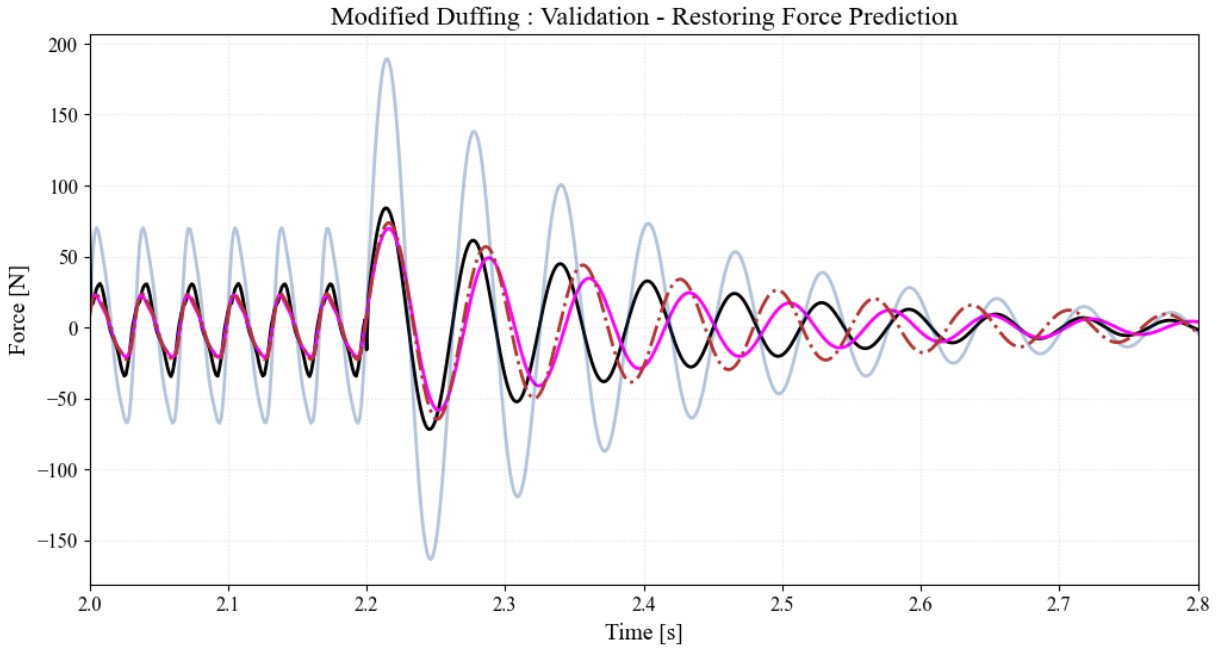


Figure M.11: Case 2.0 : Validation - Restoring Force Prediction

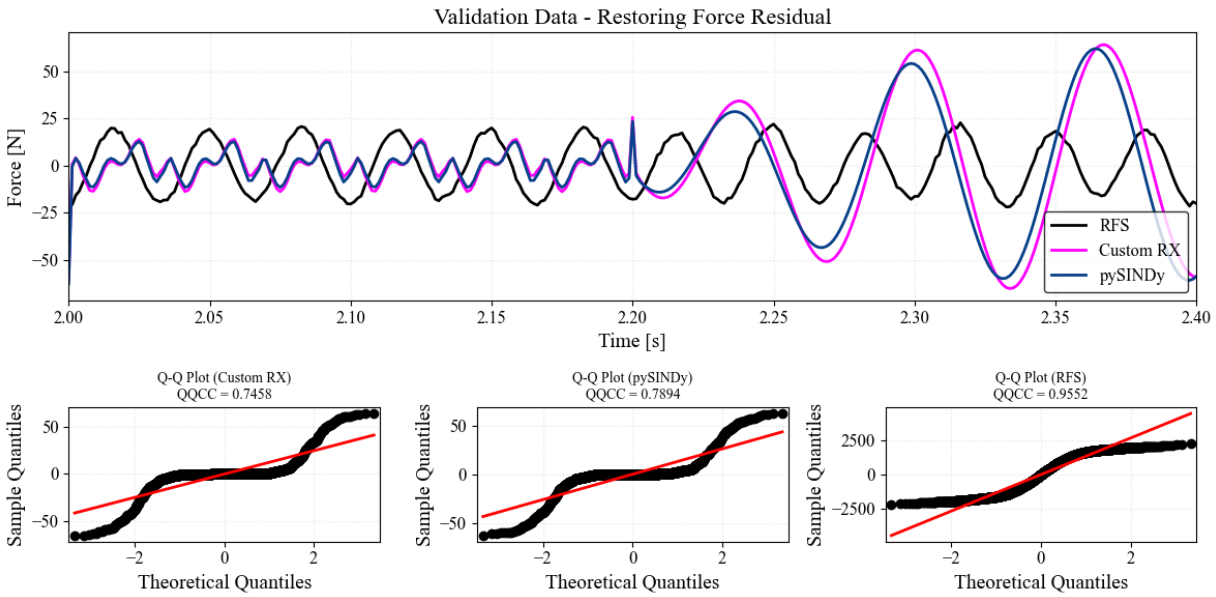


Figure M.12: Case 2.0 : Training Data - Restoring Force Residual

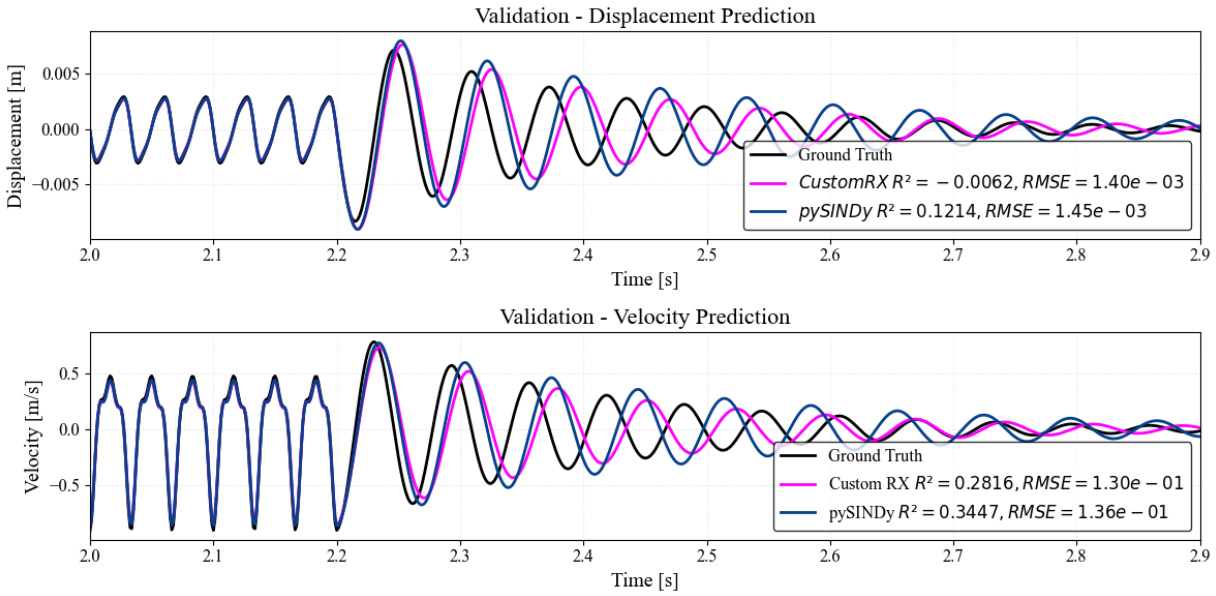


Figure M.13: Case 2.0 : Validation - Displacement & Velocity Prediction

Ground Truth :

$$f(t) - m\ddot{y} = 5.0e6y^5 + 2000.0y^3\dot{y}^2 + 1.0e5y^3 + 1.0e4y + 10.0\dot{y}$$

RFS :

$$f(t) - m\ddot{y} = 2.961y^4\dot{y}^4 + 7.325y^4\dot{y}^3 - 22.31y^4\dot{y}^2 - 42.58y^4\dot{y} + 606.8y^4 - 311.6y^3\dot{y}^4 - 10.97y^3\dot{y}^3 + 454.4y^3\dot{y}^2 + 58.46y^3\dot{y} - 739.4y^3 - 50.79y^2\dot{y}^4 + 1.12y^2\dot{y}^3 + 73.68y^2\dot{y}^2 + 26.46y^2\dot{y} - 477.9y^2 + 568.8y\dot{y}^4 + 26.62y\dot{y}^3 - 929.2y\dot{y}^2 - 43.04y\dot{y} - 5088.0y + 56.34\dot{y}^4 - 127.9\dot{y}^3 - 32.28\dot{y}^2 - 210.4\dot{y} - 58.26$$

Custom RX :

$$f(t) - m\ddot{y} = -6.635 \cdot 10^6 y^5 + 6.346 \cdot 10^5 y^3 + 32.01y\dot{y}^2 + 7557.0y + 8.502\dot{y}$$

pySINDy :

$$f(t) - m\ddot{y} = 3.501 \cdot 10^5 y^3 + 38.29y\dot{y}^2 + 8019.0y + 4.86\dot{y}$$

Plots for Case 2.1

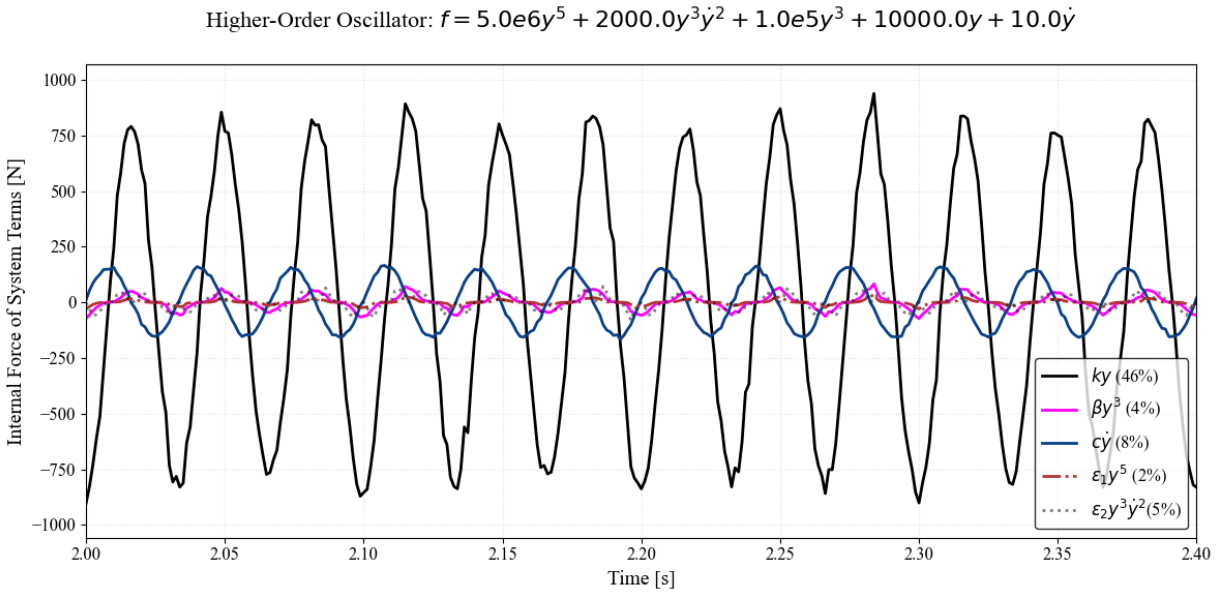


Figure M.14: Case 2.1 Components

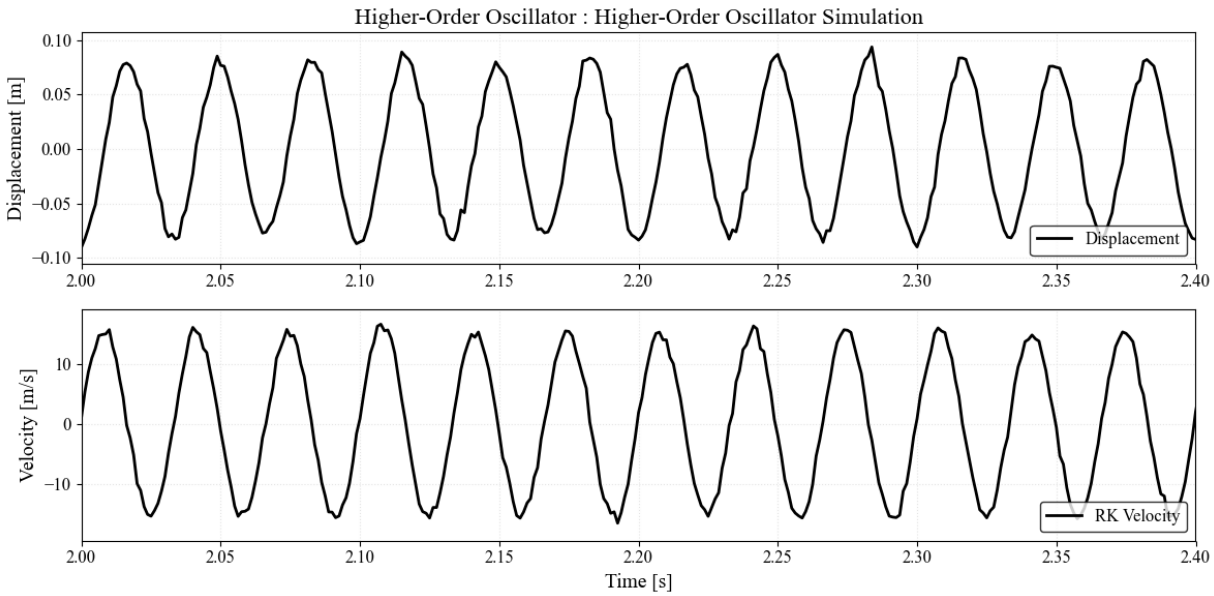


Figure M.15: Case 2.1 : Case 2.1 Simulation

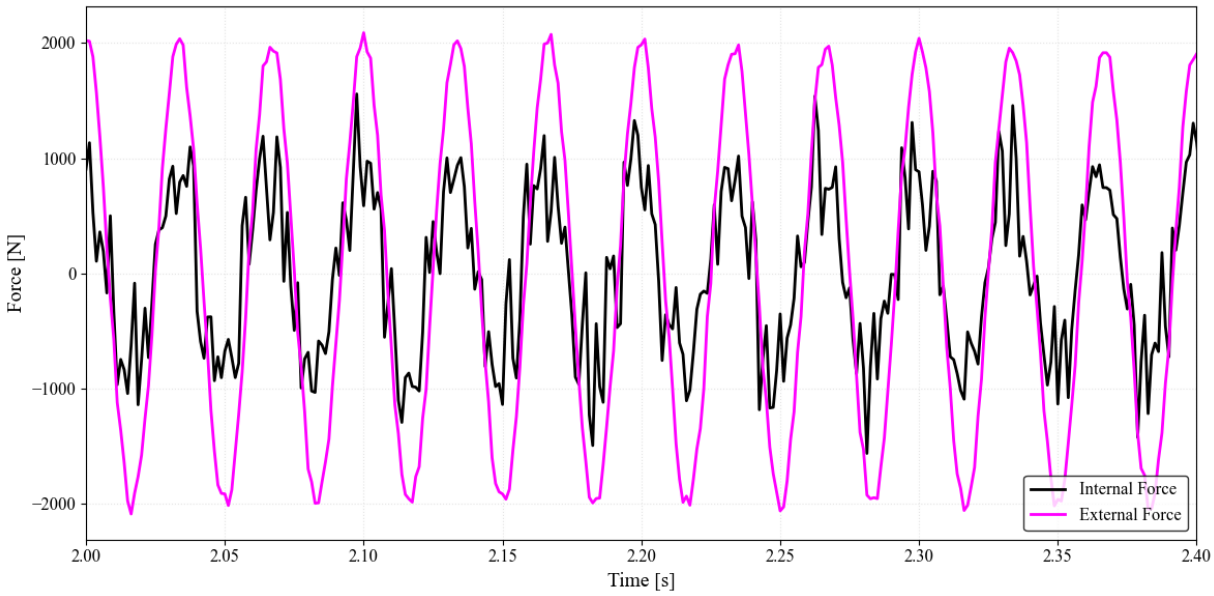


Figure M.16: Case 2.1 : Target Restoring Force

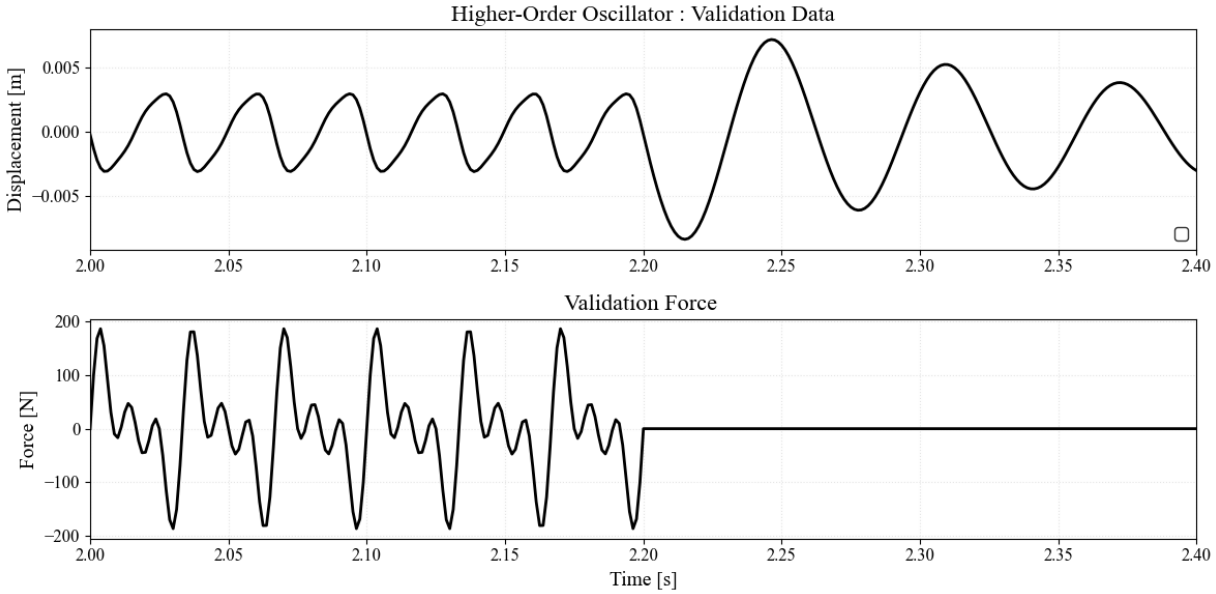


Figure M.17: Case 2.1 : Validation Data

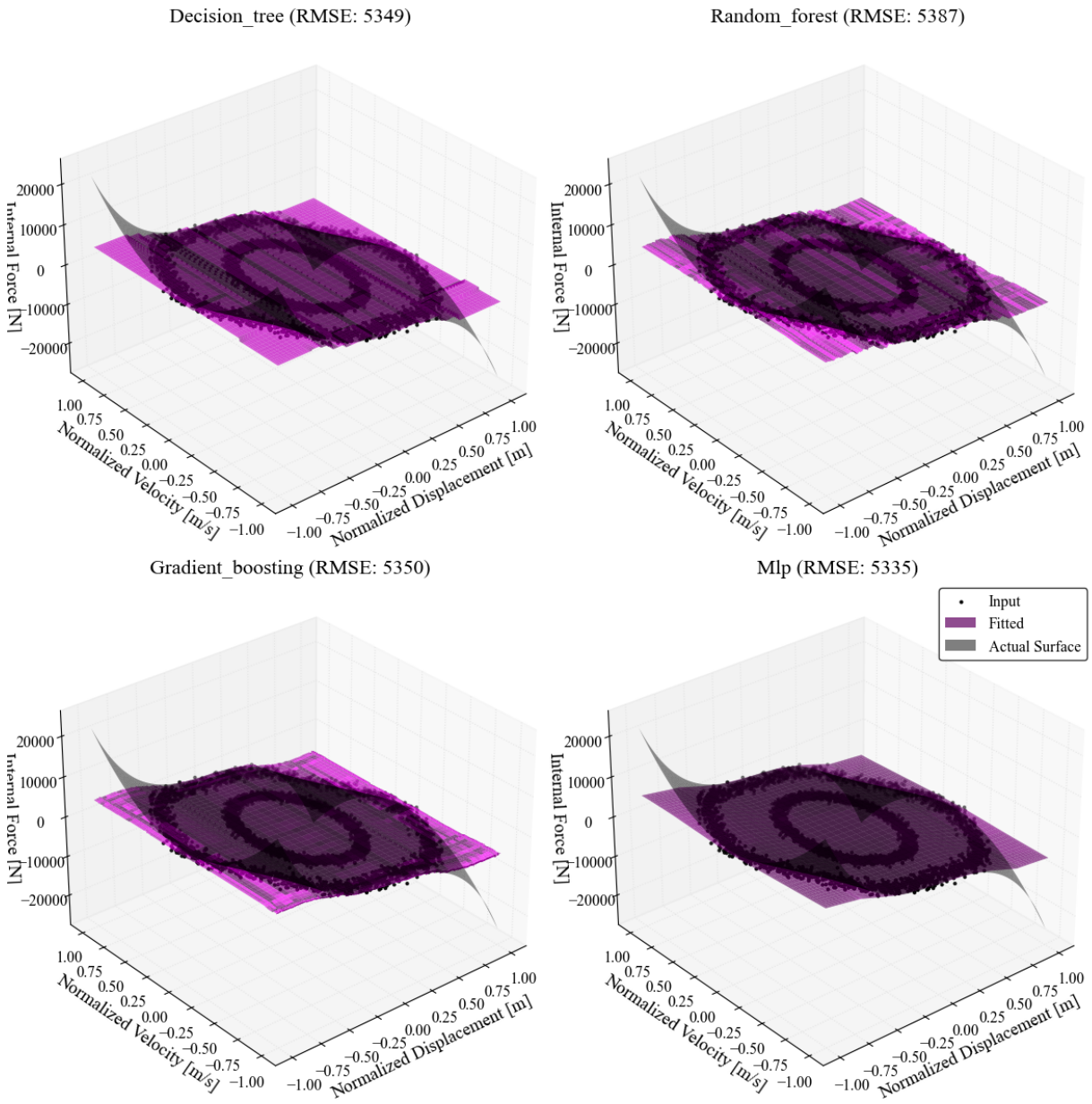
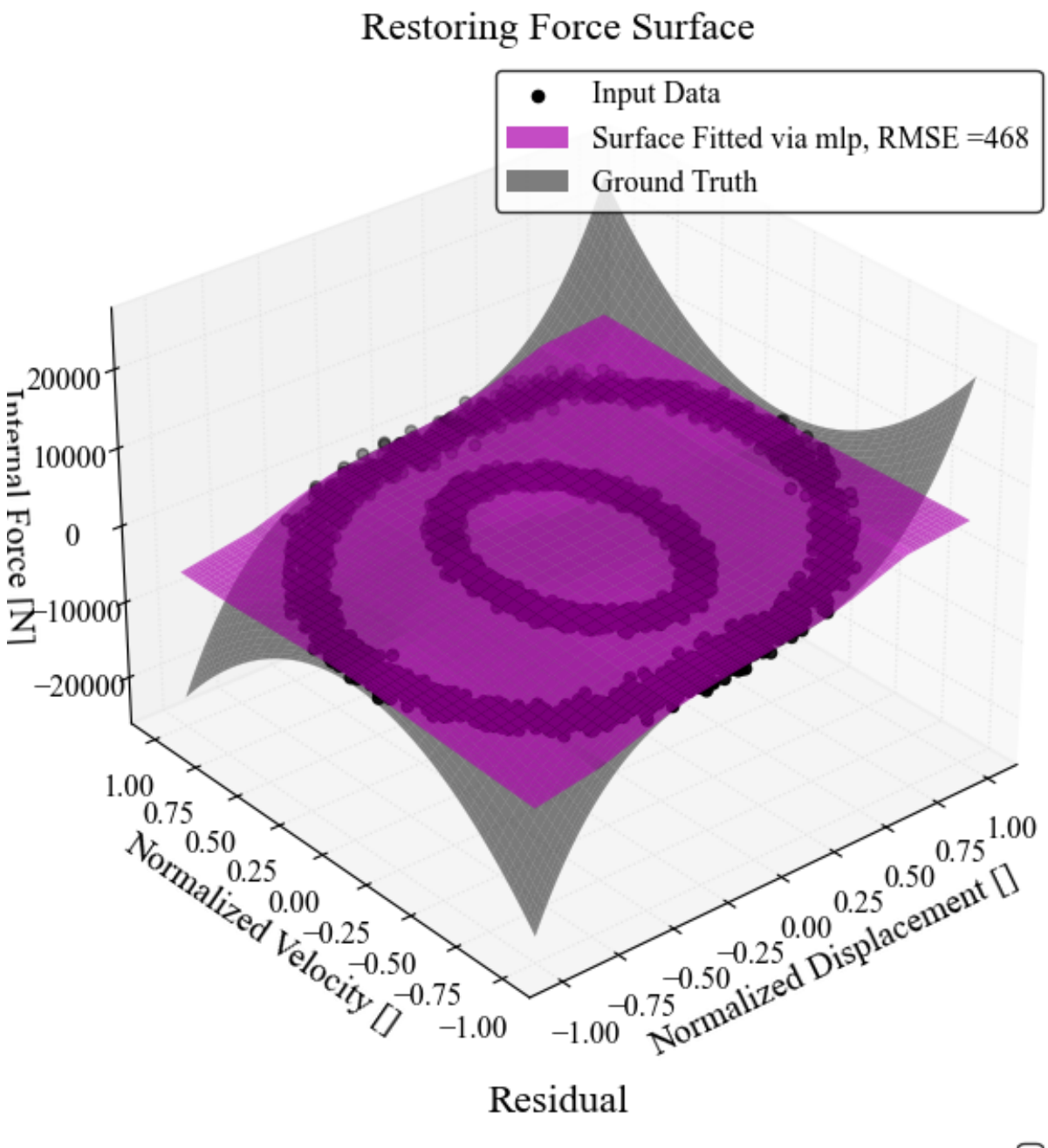
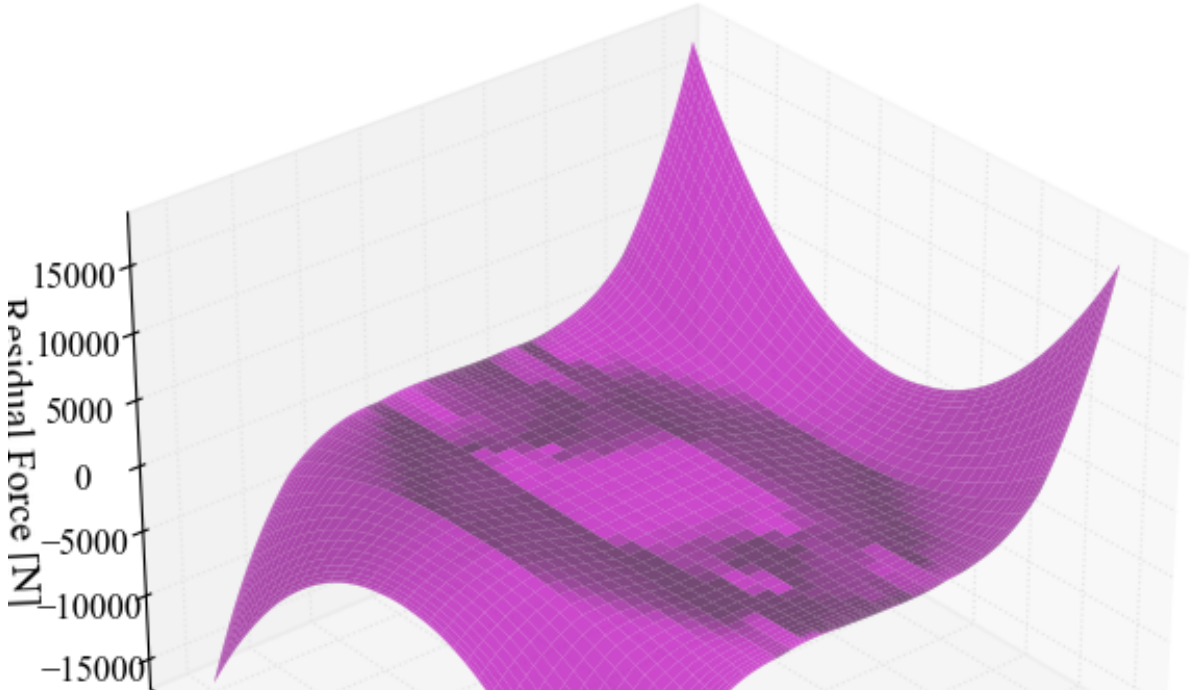


Figure M.18: Case 2.1: Machine Learning fitted 3D Surfaces


Restoring Force Surface
Input Data
Surface Fitted via mlp, RMSE =468
Ground Truth
Internal Force [N]
20000
10000
0
−10000
−20000
Normalized Velocity []
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Normalized Displacement []
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Residual


Residual Force [N]
15000
10000
5000
0
−5000
−10000
−15000

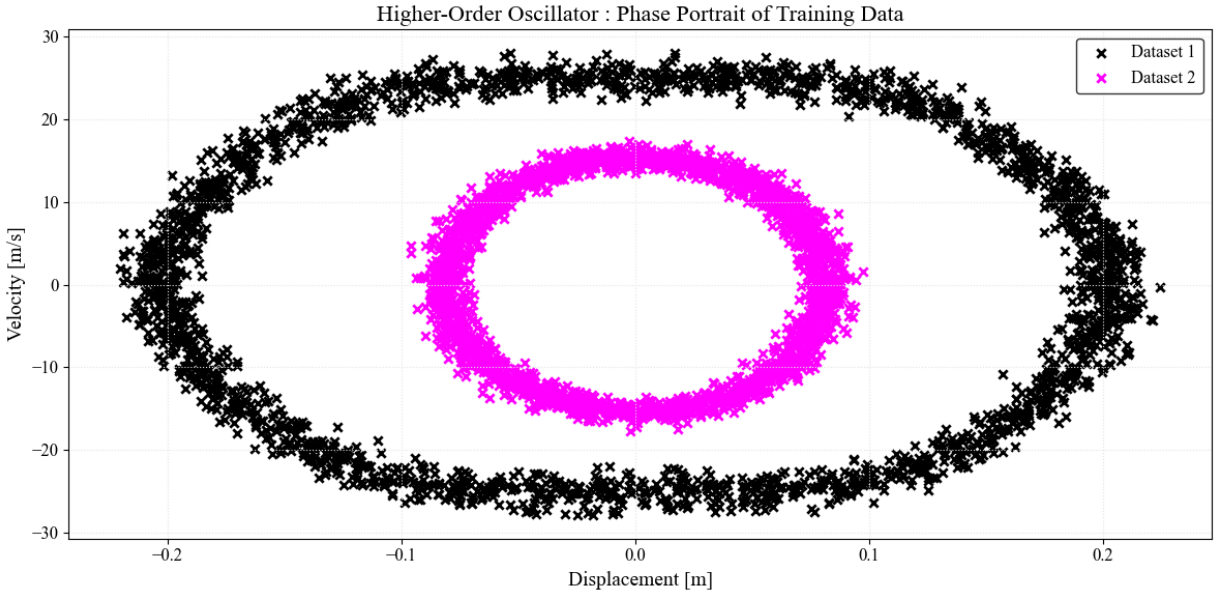


Figure M.20: Case 2.1 : Phase Portrait of Training Data

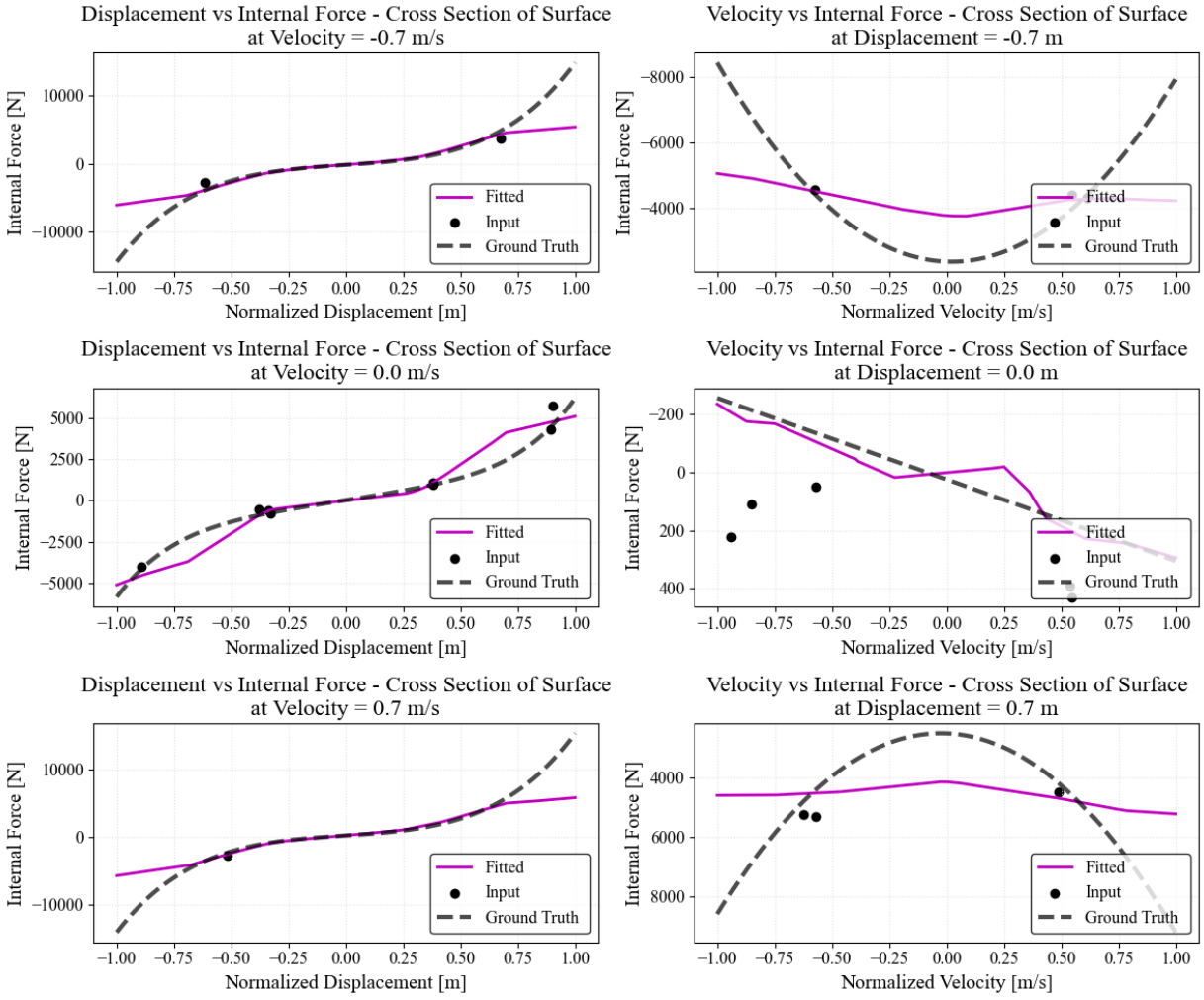


Figure M.21: Case 2.1 : Cross Sections of Surface

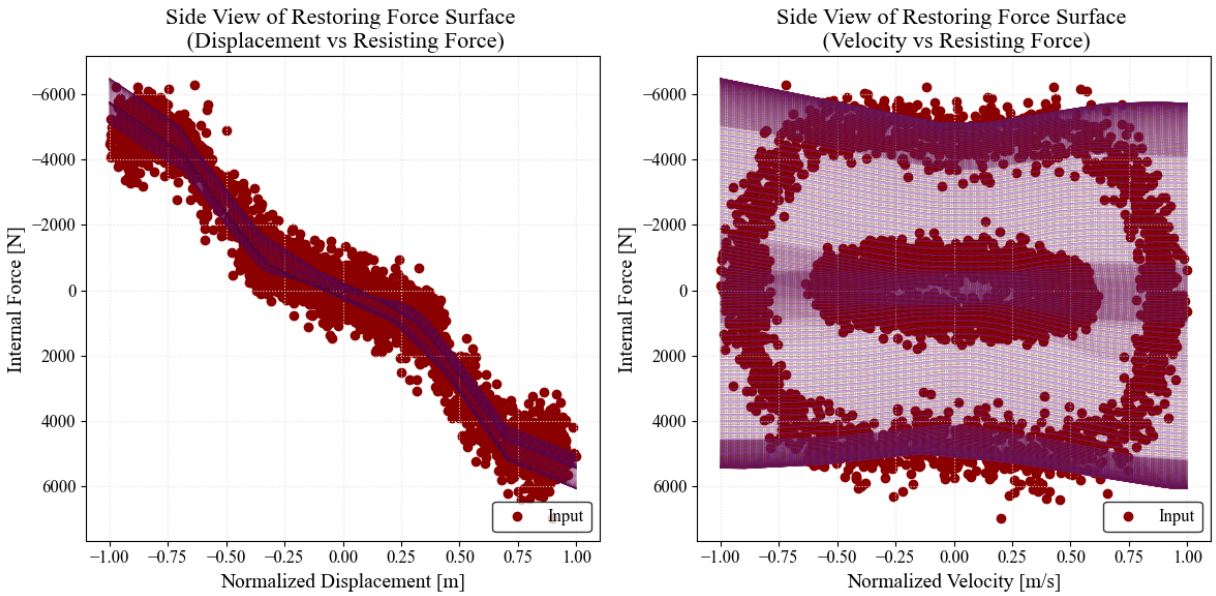


Figure M.22: Case 2.1 : Side Views of Surface

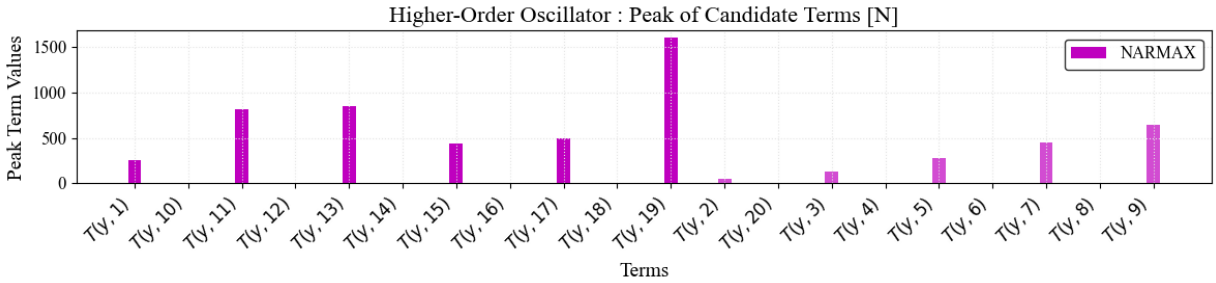


Figure M.23: Case 2.1 : Force Features comparison

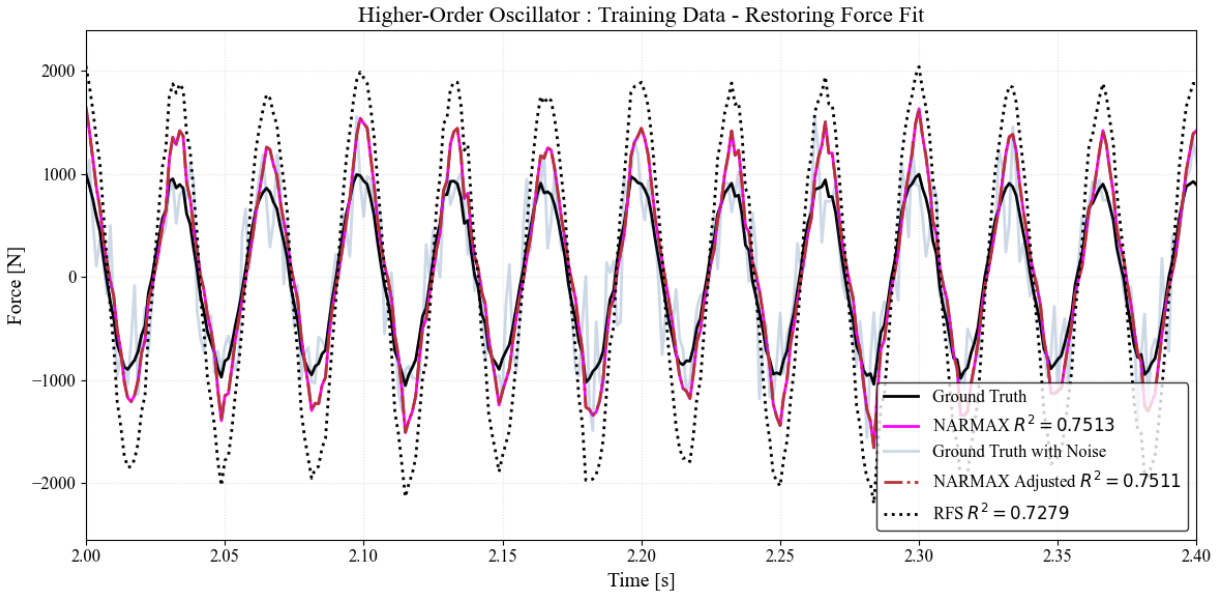


Figure M.24: Case 2.1 : Training Data - Restoring Force Fit

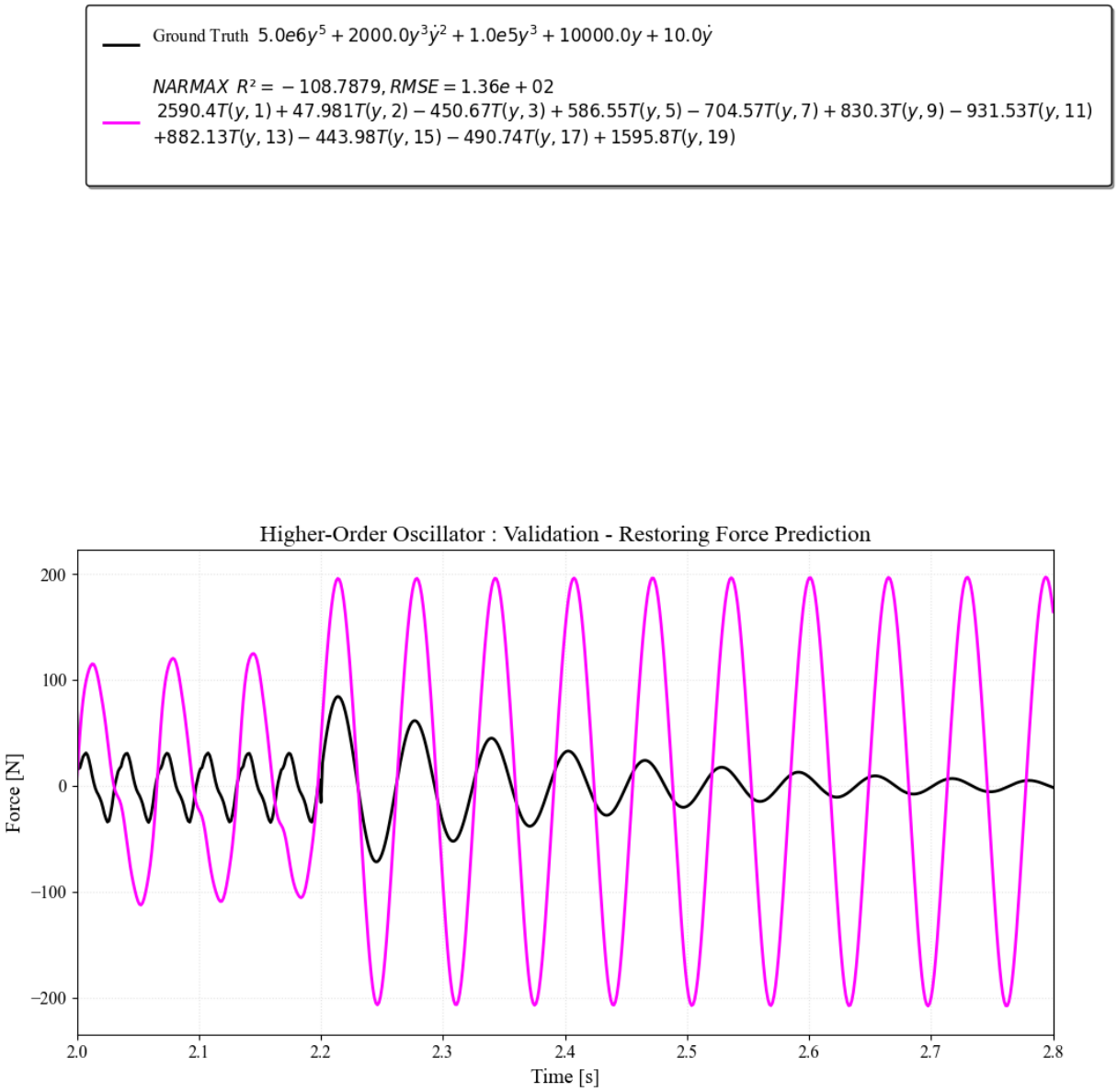


Figure M.25: Case 2.1 : cutoff percentage

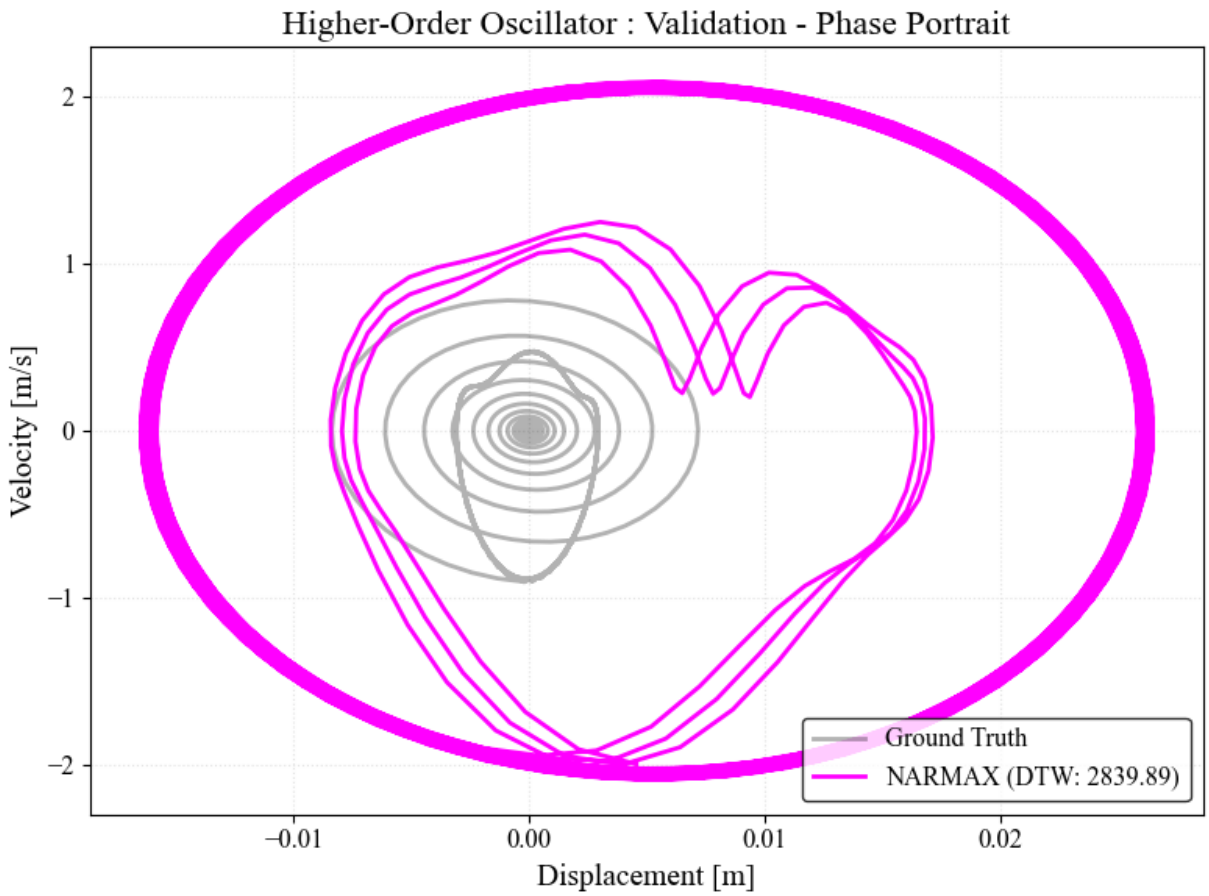


Figure M.26: Case 2.1 : Phase portrait

$Higher - OrderOscillator$

Ground Truth :

$$f(t) - m\ddot{y} = 5.0e6y^5 + 2000.0y^3\dot{y}^2 + 1.0e5y^3 + 1.0e4y + 10.0\dot{y}$$

Custom RX :

$$f(t) - m\ddot{y} = 2590.0T(y,1) + 47.98T(y,2) - 450.7T(y,3) + 586.5T(y,5) - 704.6T(y,7) + 830.3T(y,9) - 931.5 T(y,11) + 882.1T(y,13) - 444.0T(y,15) - 490.7T(y,17) + 1596.0T(y,19)$$

Plots for Case 3.0

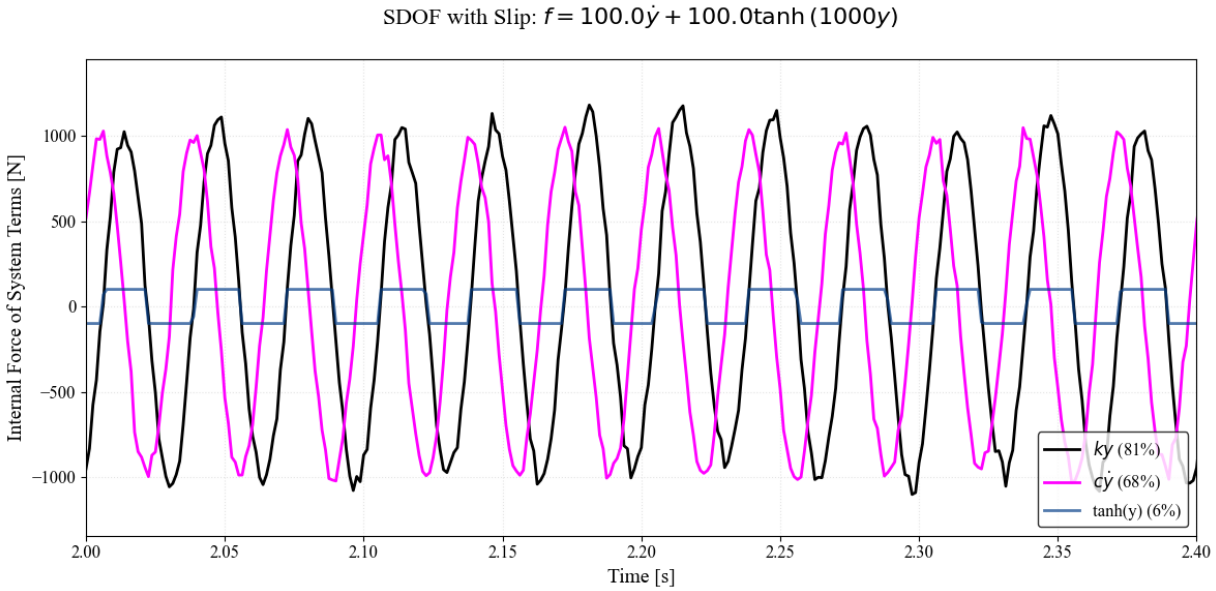


Figure M.27: Case 3.0 Components

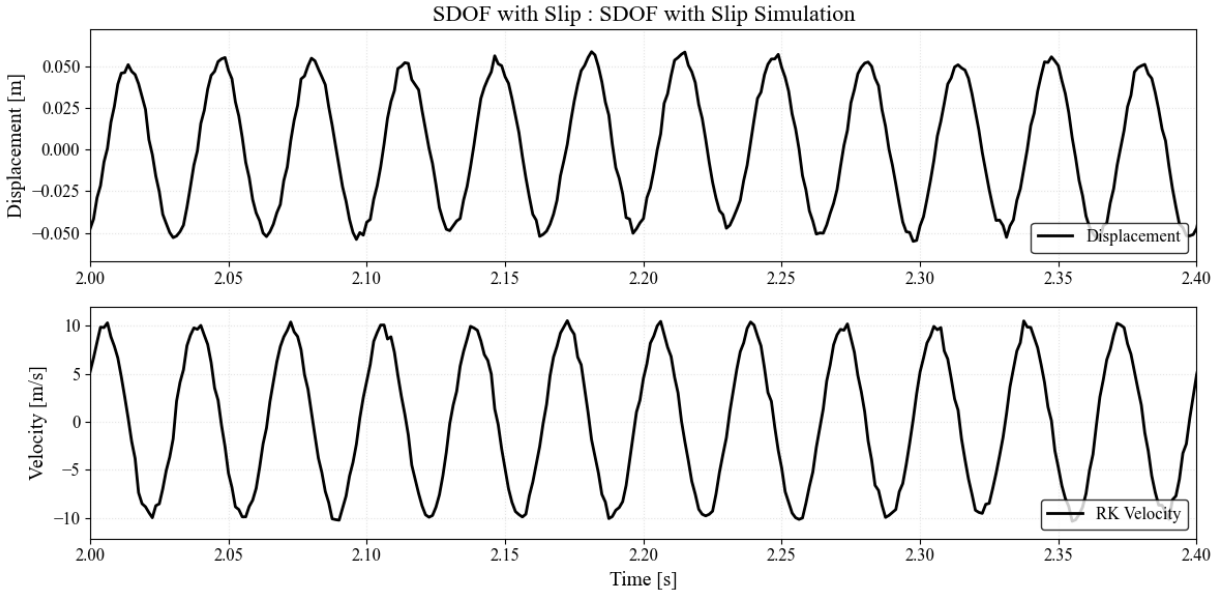


Figure M.28: Case 3.0 : Case 3.0 Simulation

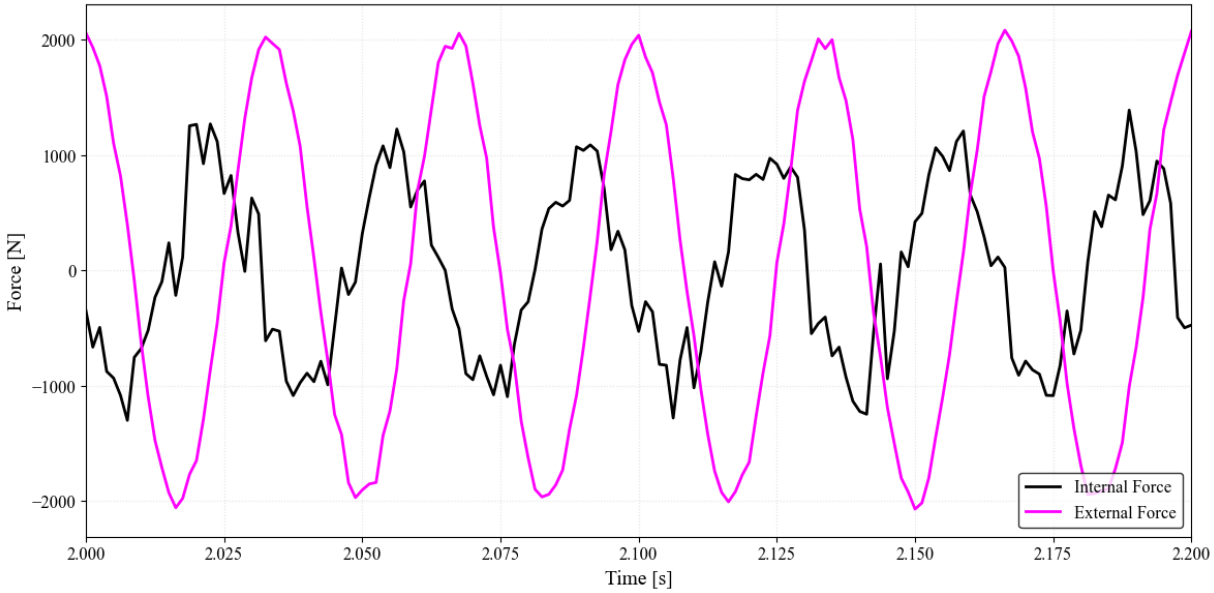


Figure M.29: Case 3.0 : Target Restoring Force

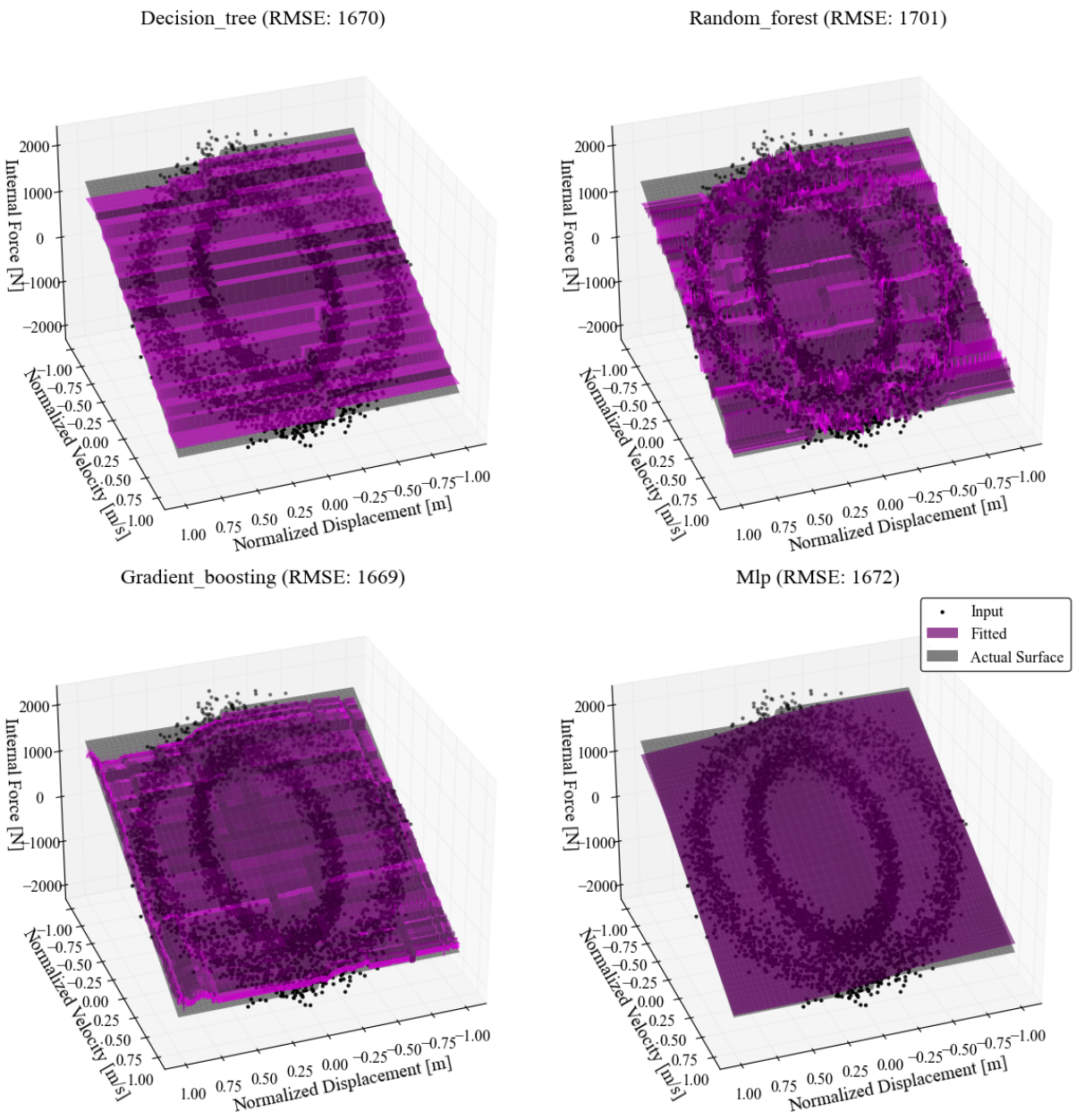


Figure M.30: Case 3.0: Machine Learning fitted 3D Surfaces

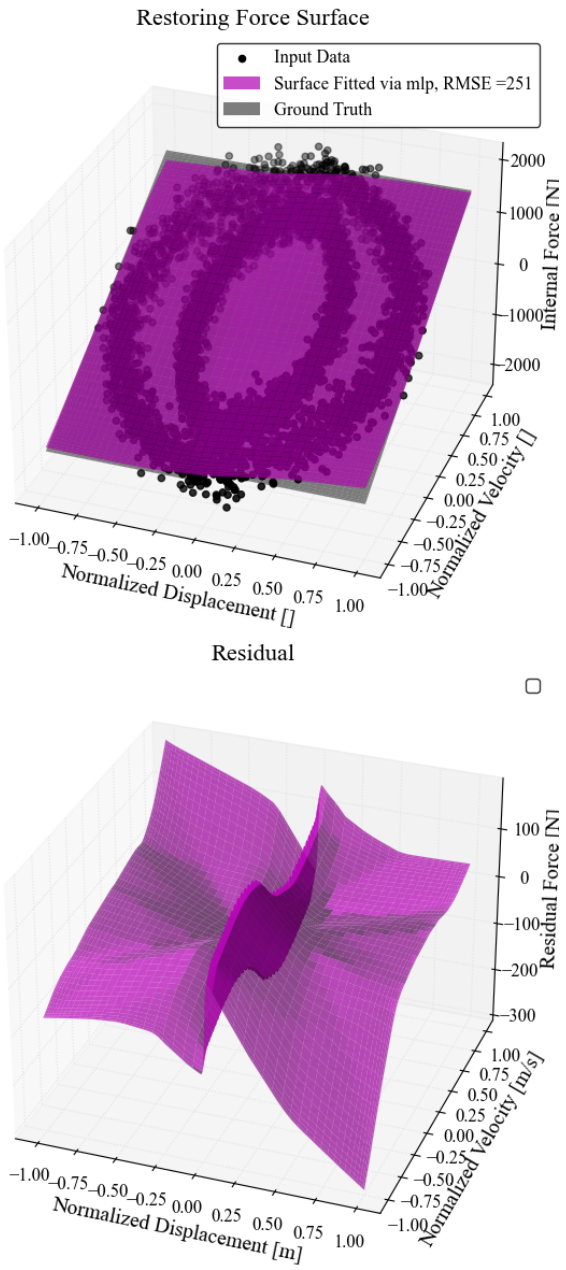


Figure M.31: Case 3.0: Selected Restoring Force Surface

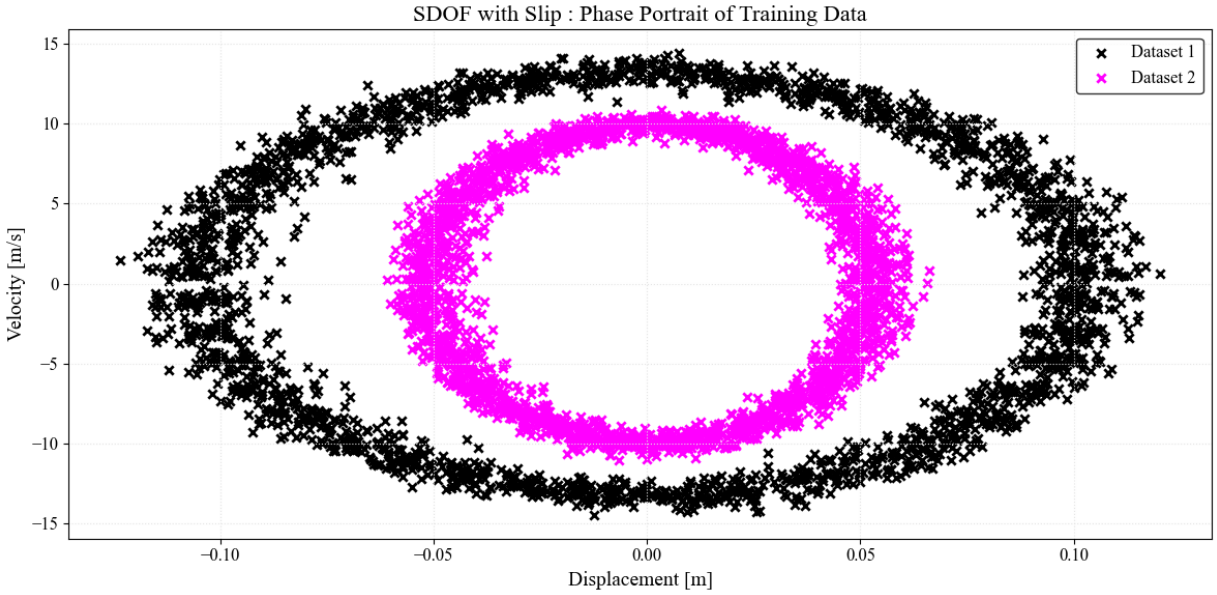


Figure M.32: Case 3.0 : Phase Portrait of Training Data

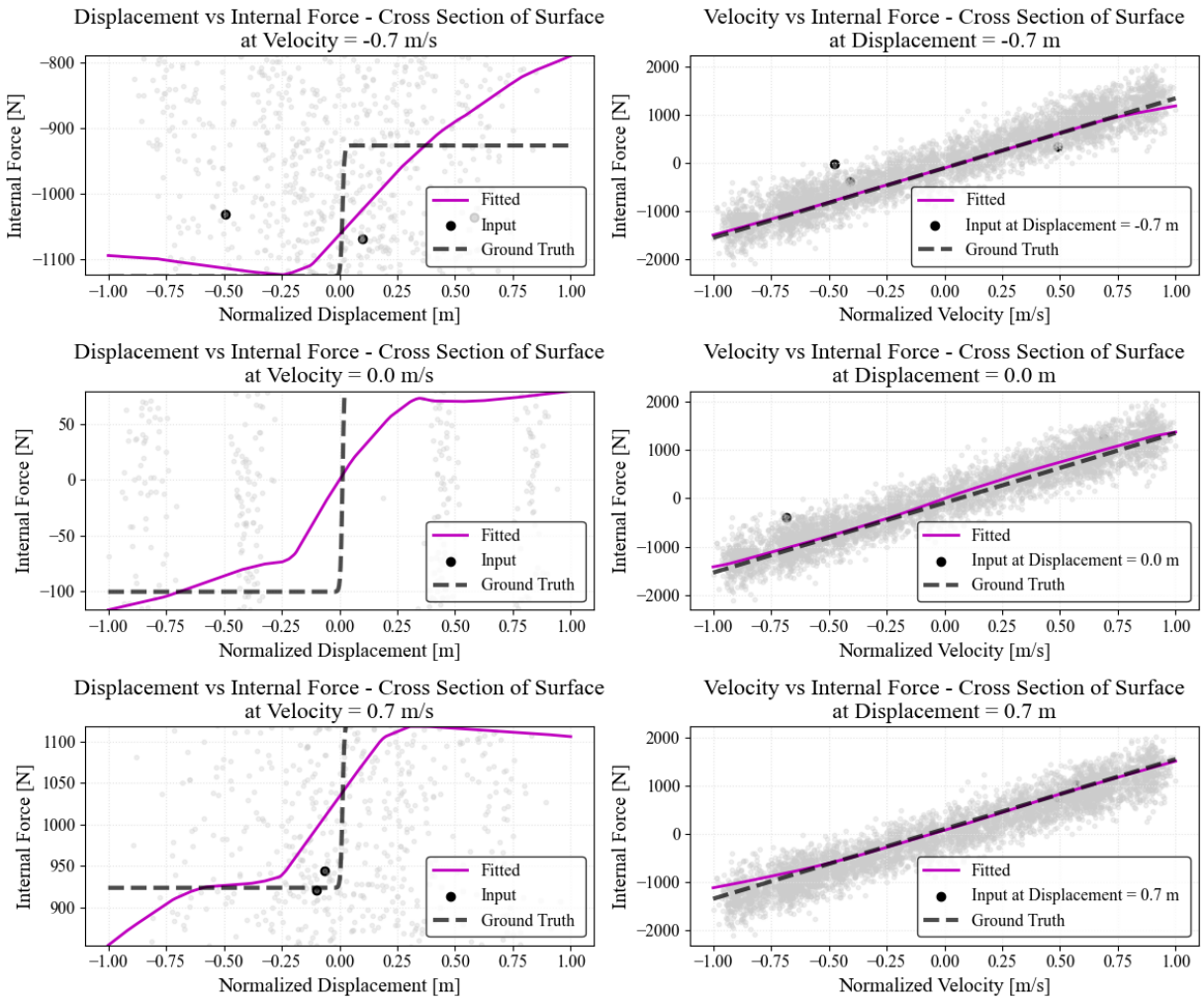


Figure M.33: Case 3.0 : Cross Sections of Surface

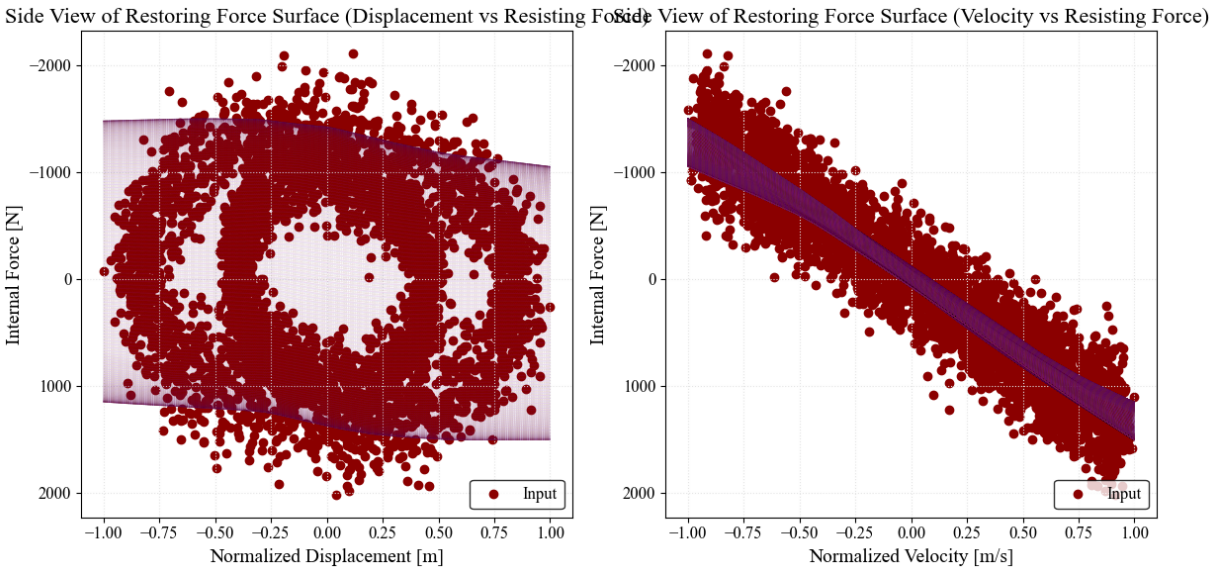


Figure M.34: Case 3.0 : Side Views of Surface

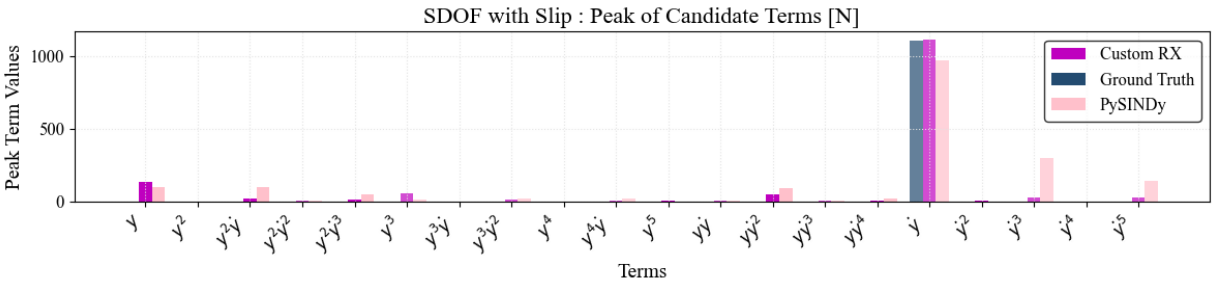


Figure M.35: Case 3.0 : Force Features comparison

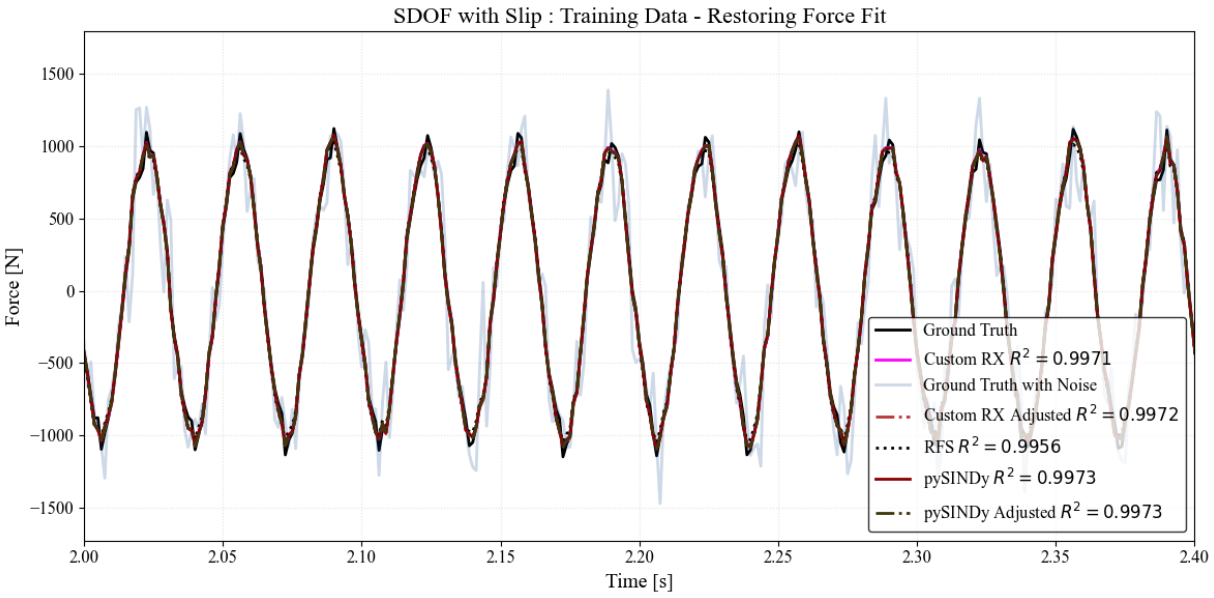


Figure M.36: Case 3.0 : Training Data - Restoring Force Fit

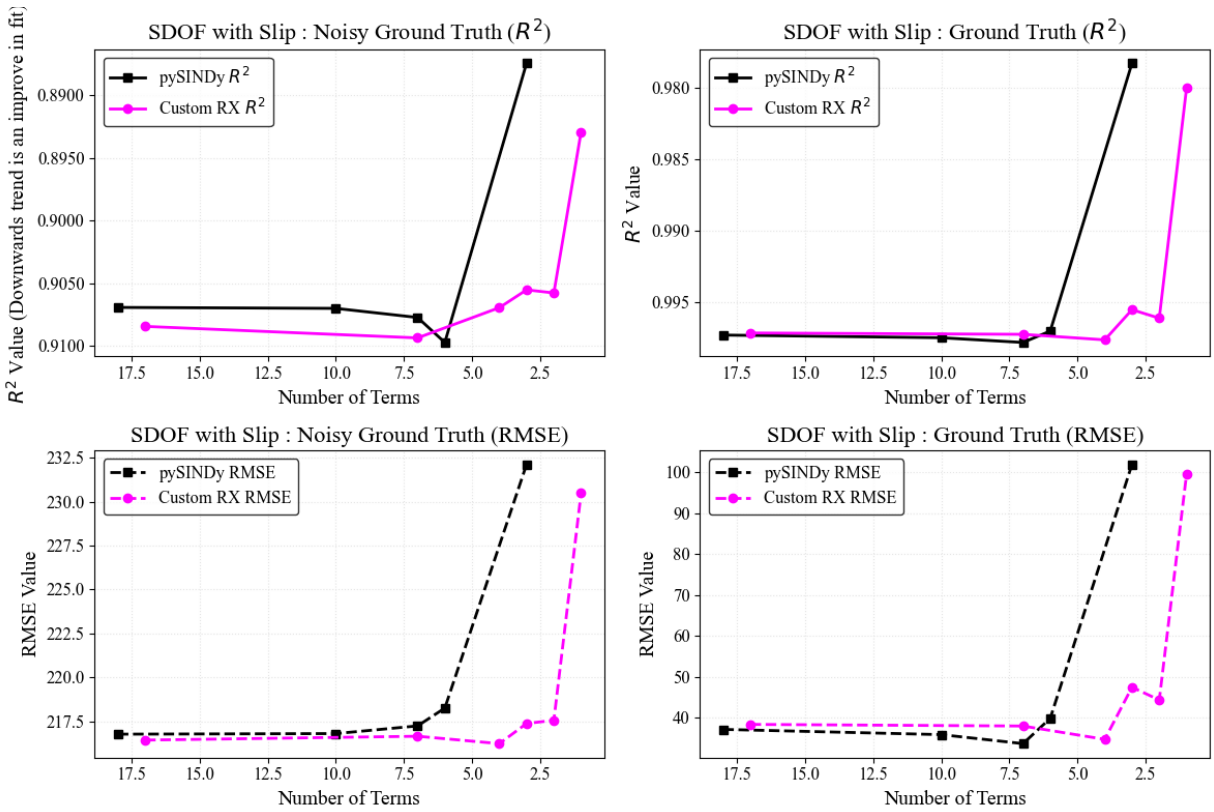


Figure M.37: Case 3.0 : Removing Unnecessary Terms

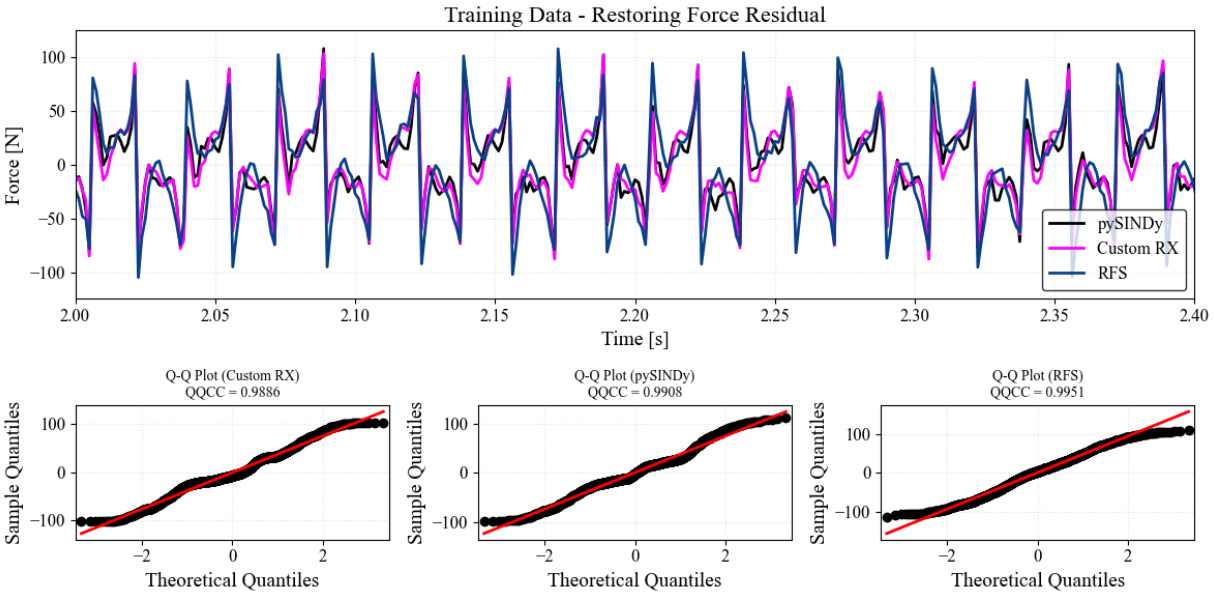


Figure M.38: Case 3.0 : Training Data - Restoring Force Residual

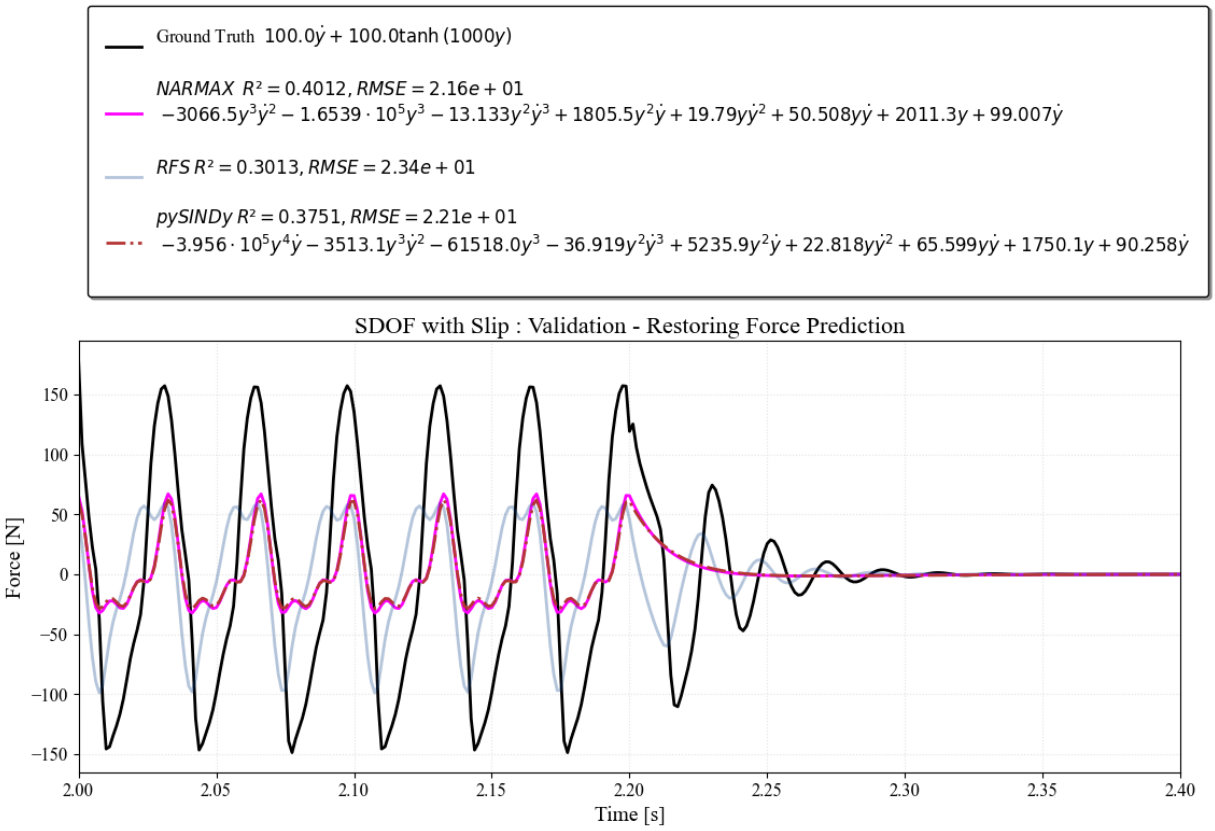


Figure M.39: Case 3.0 : Validation - Restoring Force Prediction

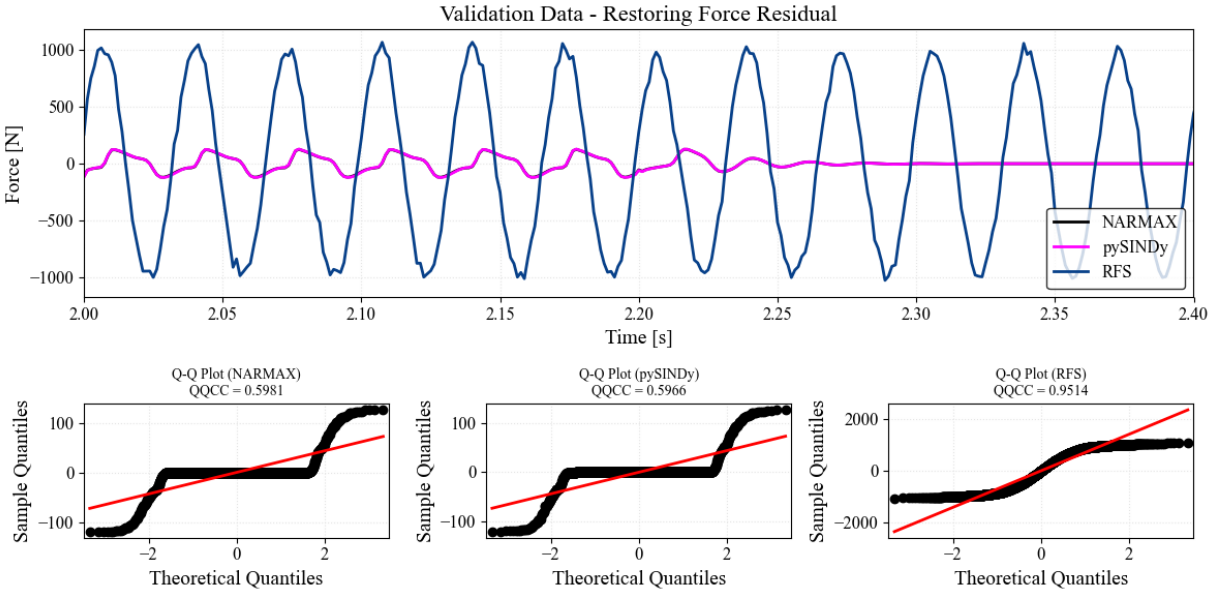


Figure M.40: Case 3.0 : Training Data - Restoring Force Residual

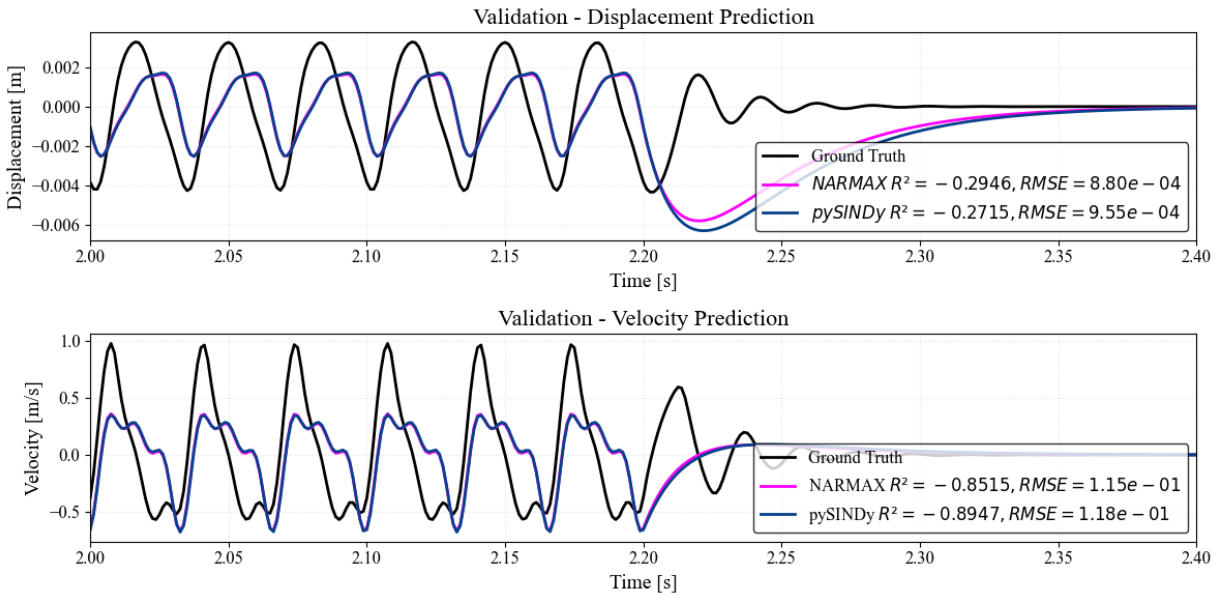


Figure M.41: Case 3.0 : Validation - Displacement & Velocity Prediction

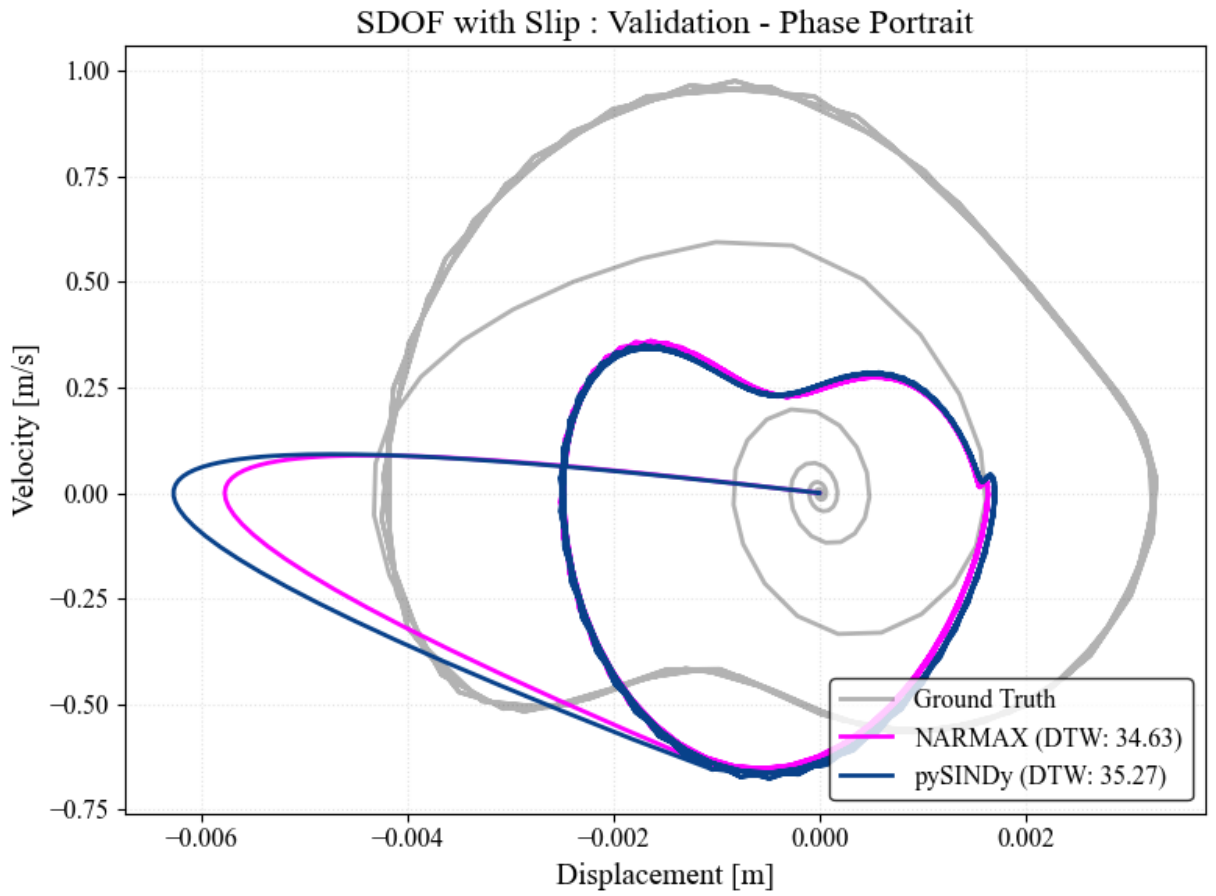


Figure M.42: Case 3.0 : Validation - Phase Portrait

Plots for Case 3.2

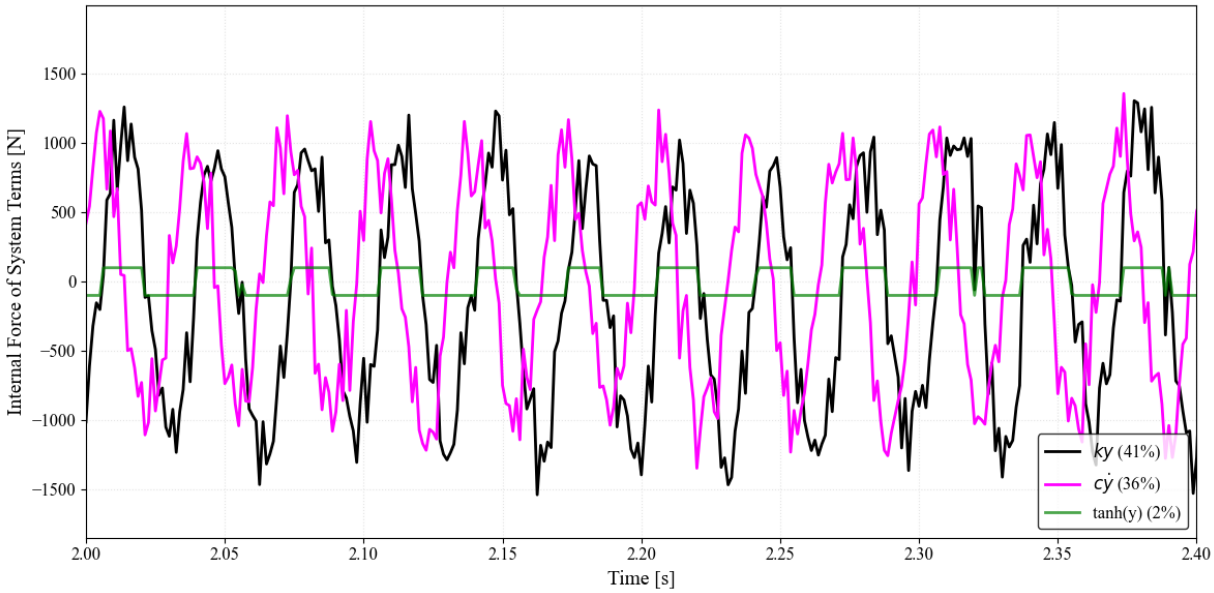


Figure M.43: Case 3.2 Components

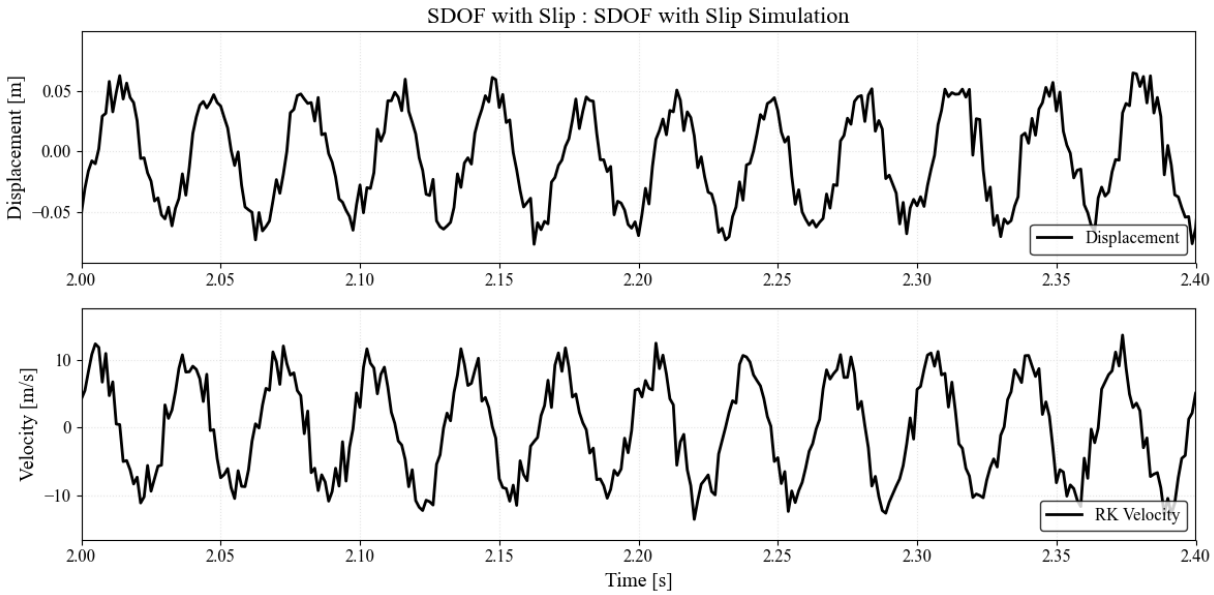


Figure M.44: Case 3.2 : Case 3.2 Simulation

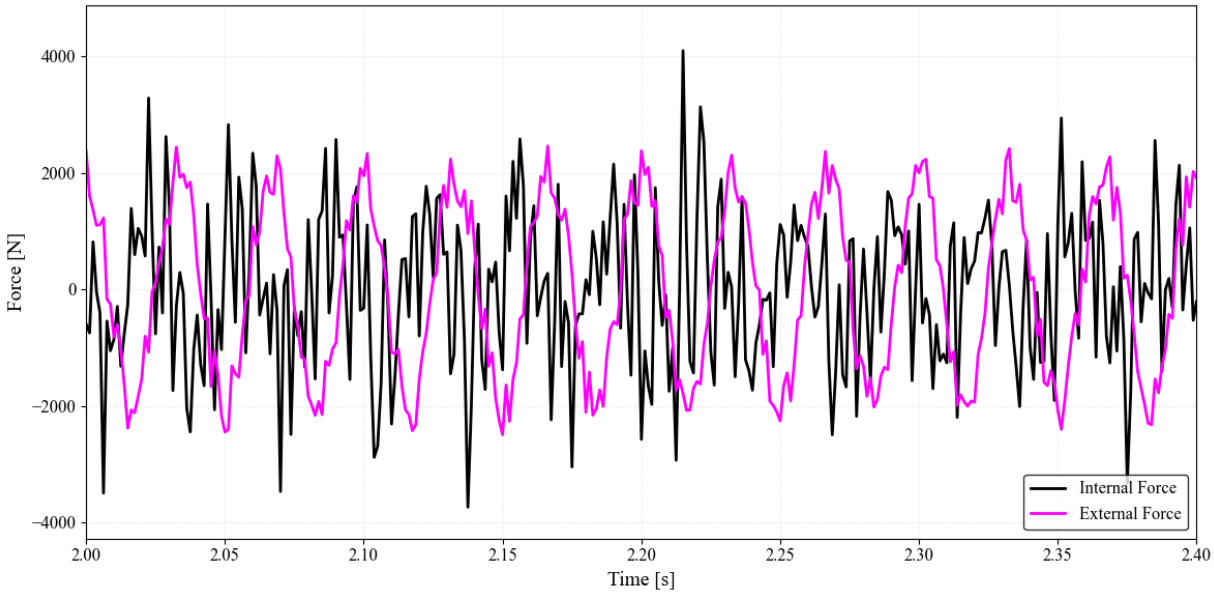


Figure M.45: Case 3.2 : Target Restoring Force

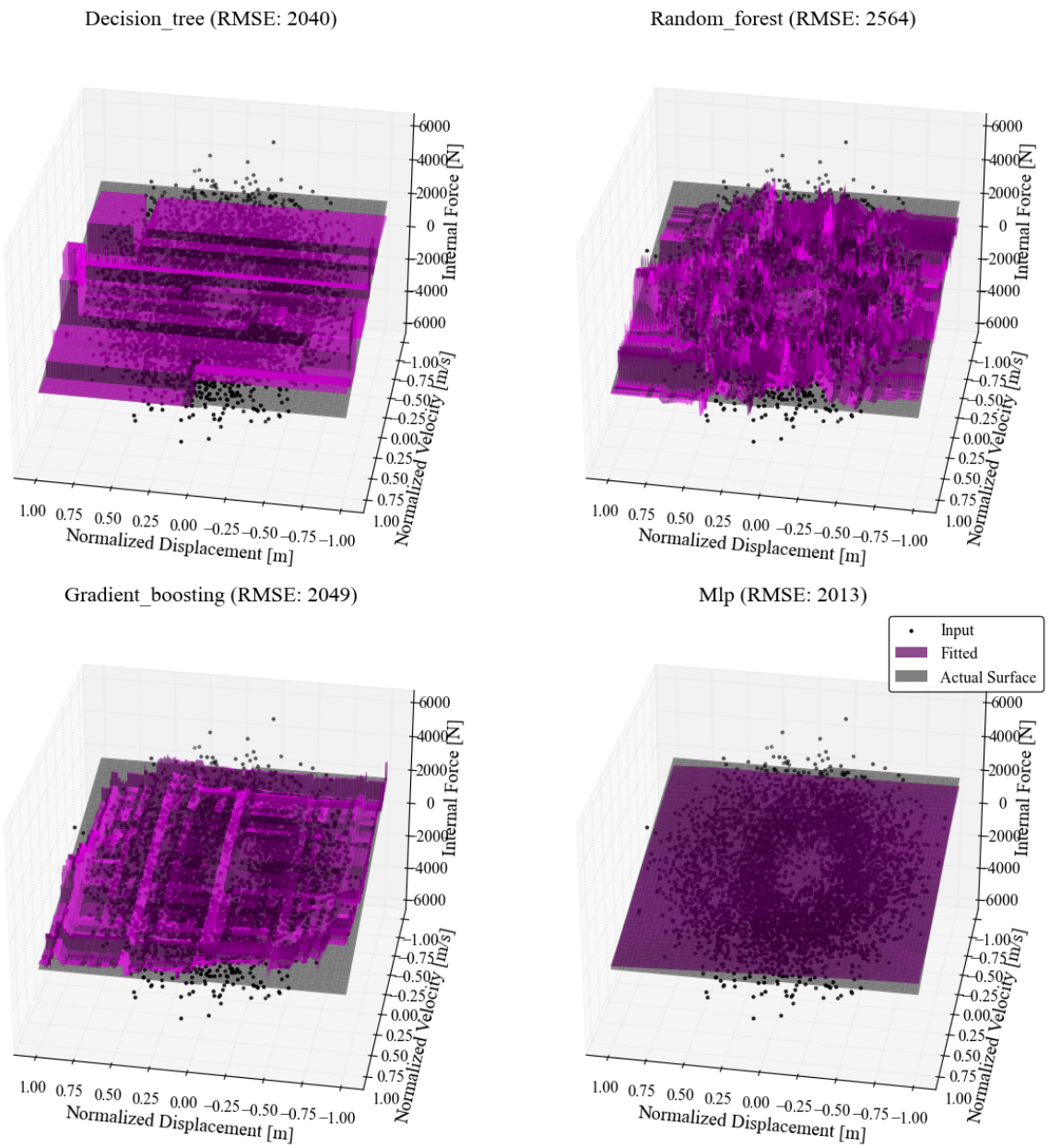


Figure M.46: Case 3.2: Machine Learning fitted 3D Surfaces

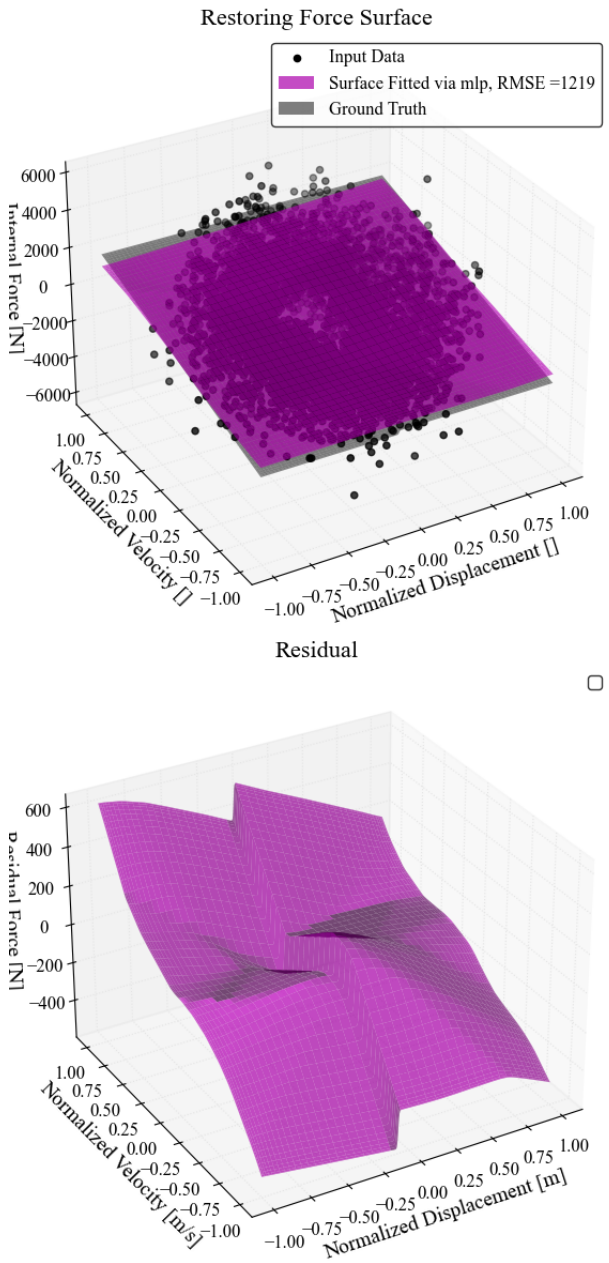


Figure M.47: Case 3.2: Selected Restoring Force Surface

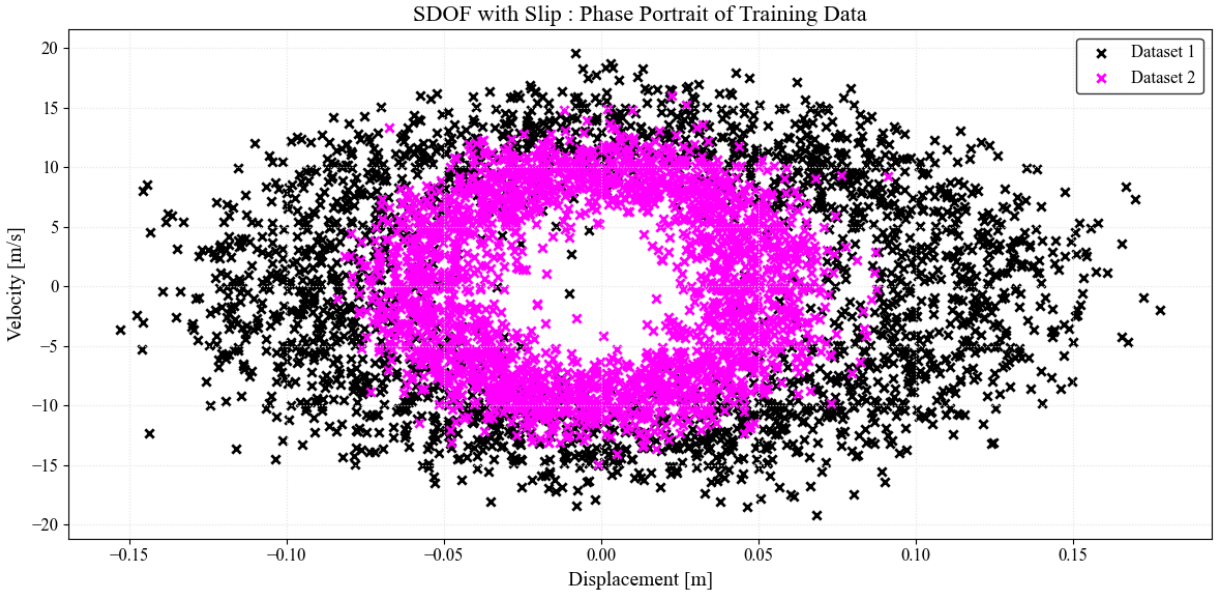


Figure M.48: Case 3.2 : Phase Portrait of Training Data

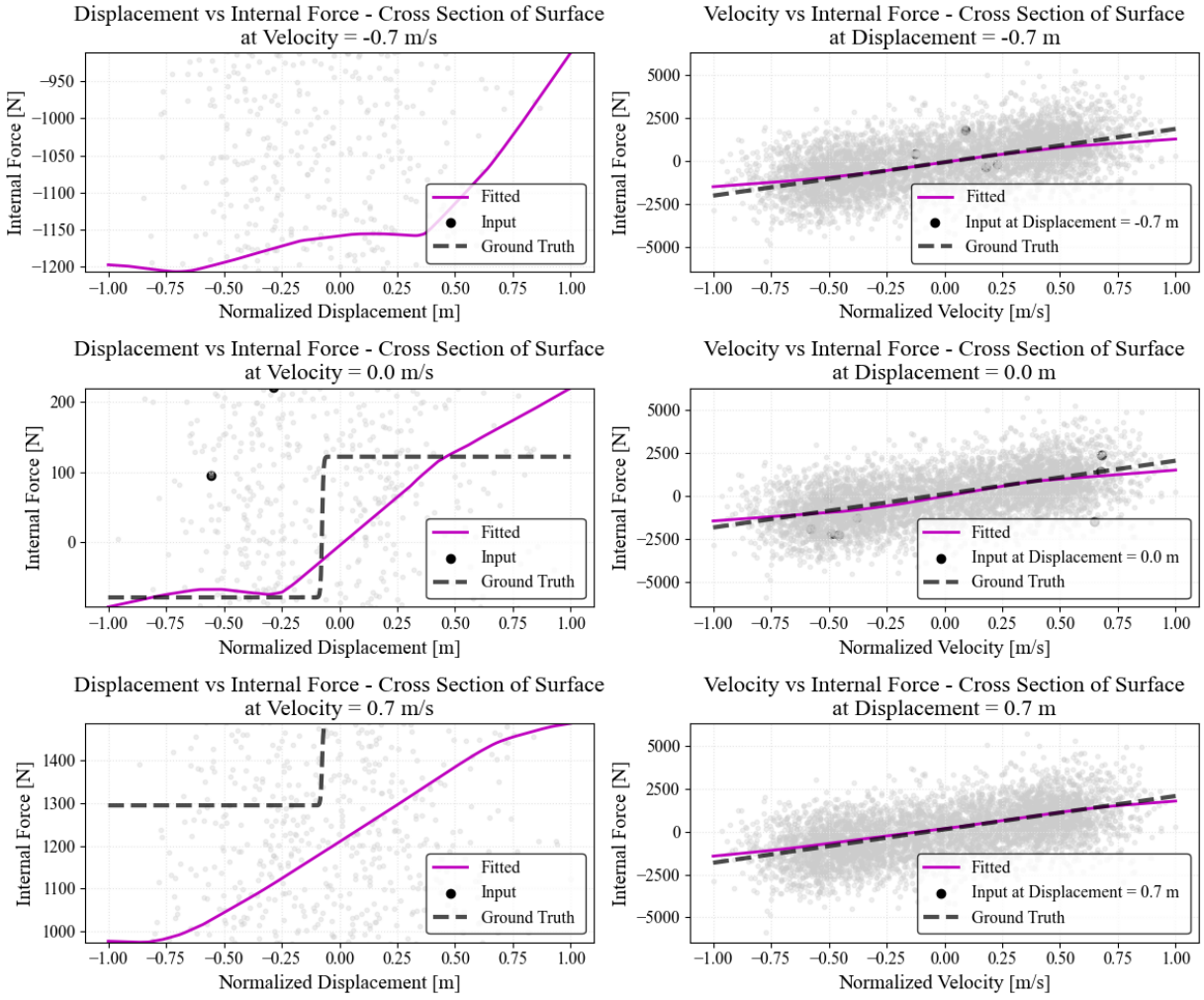


Figure M.49: Case 3.2 : Cross Sections of Surface

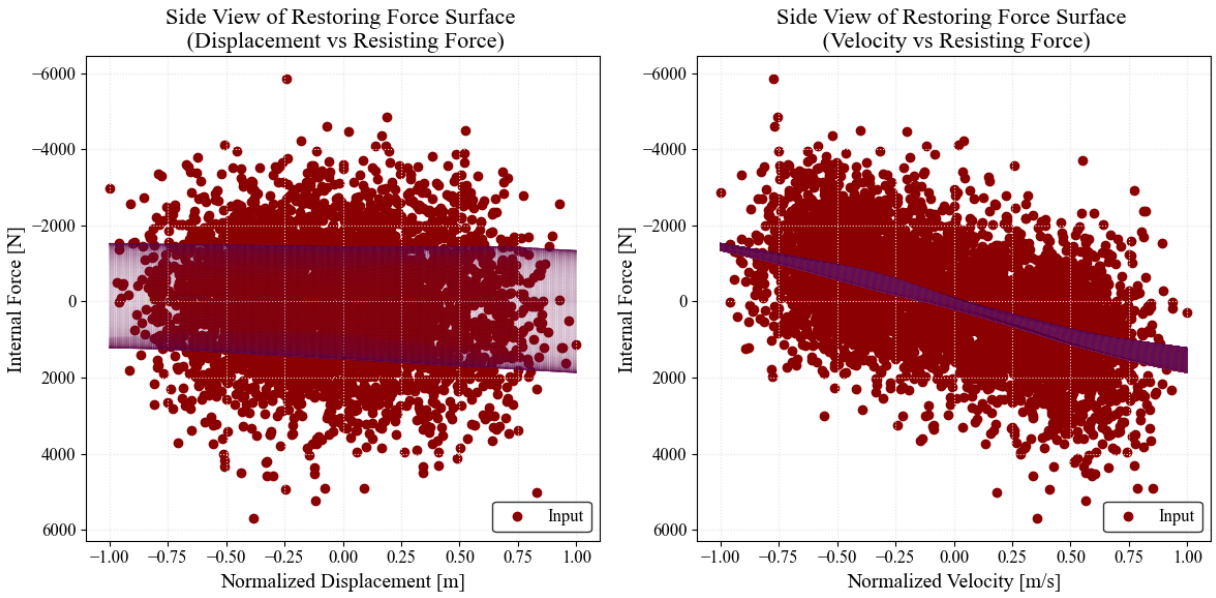


Figure M.50: Case 3.2 : Side Views of Surface

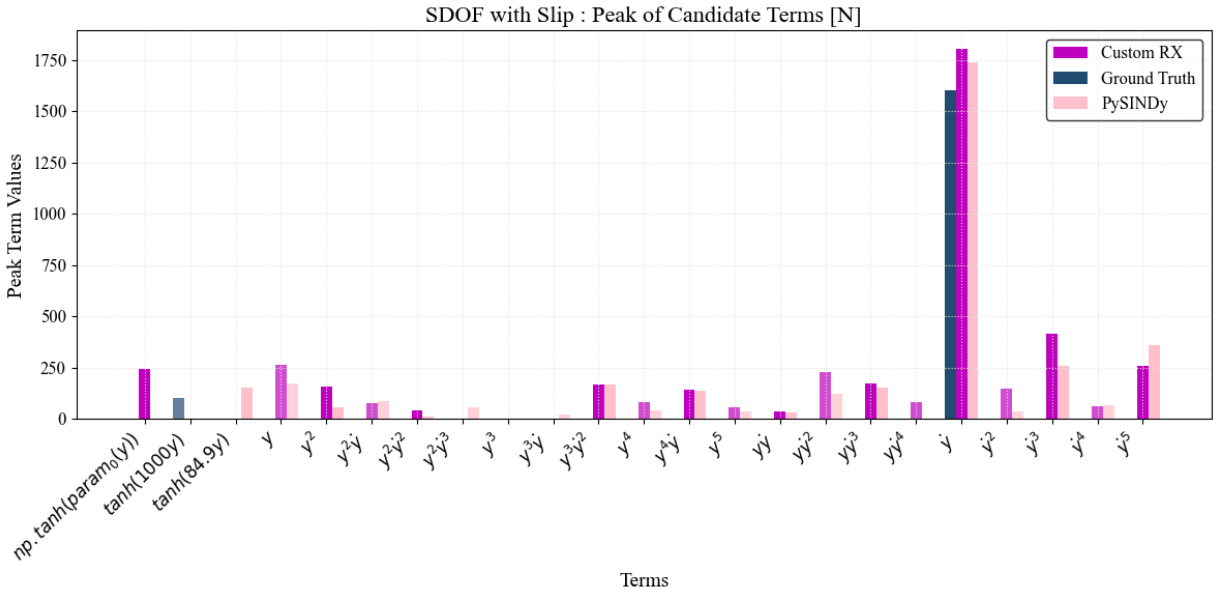


Figure M.51: Case 3.2 : Force Features comparison

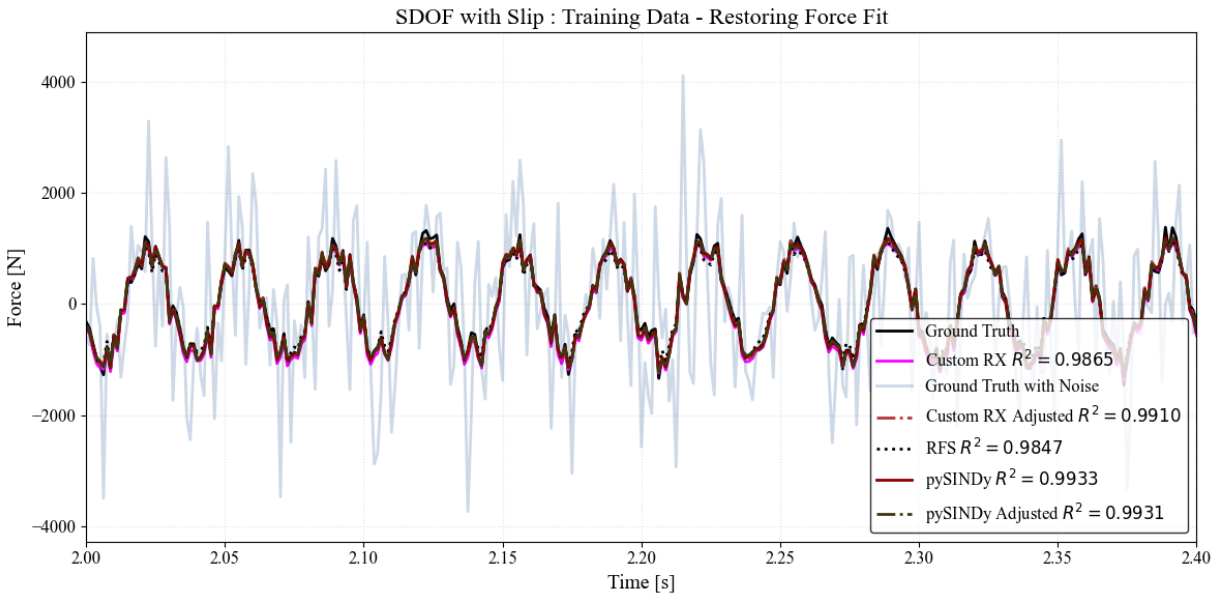


Figure M.52: Case 3.2 : Training Data - Restoring Force Fit

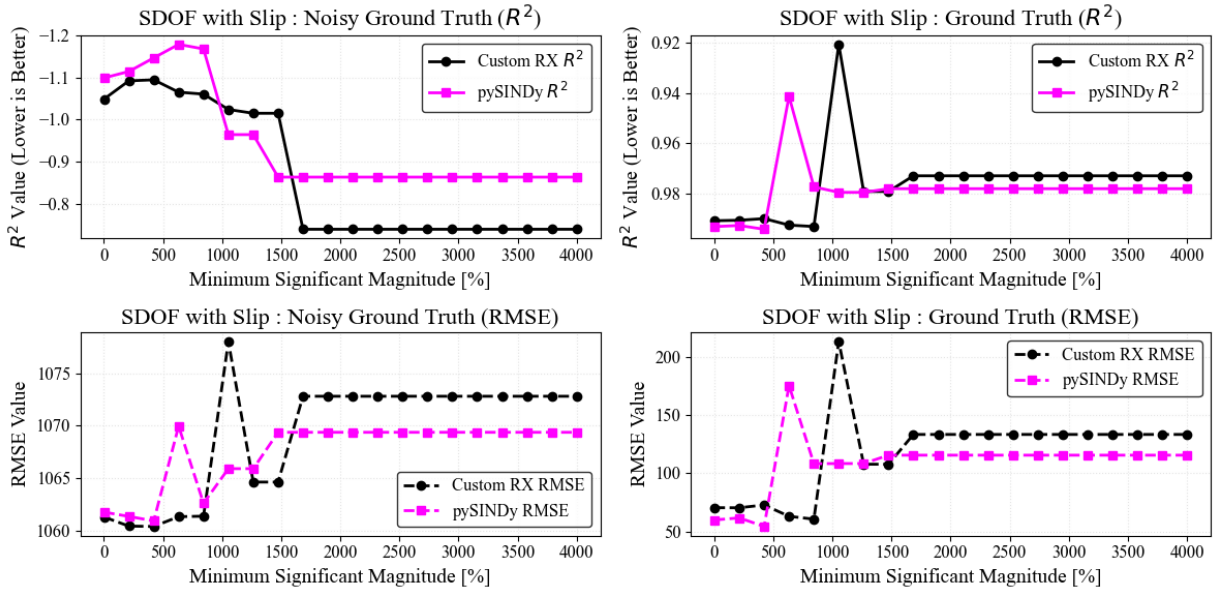


Figure M.53: Case 3.2 : cutoff percentage

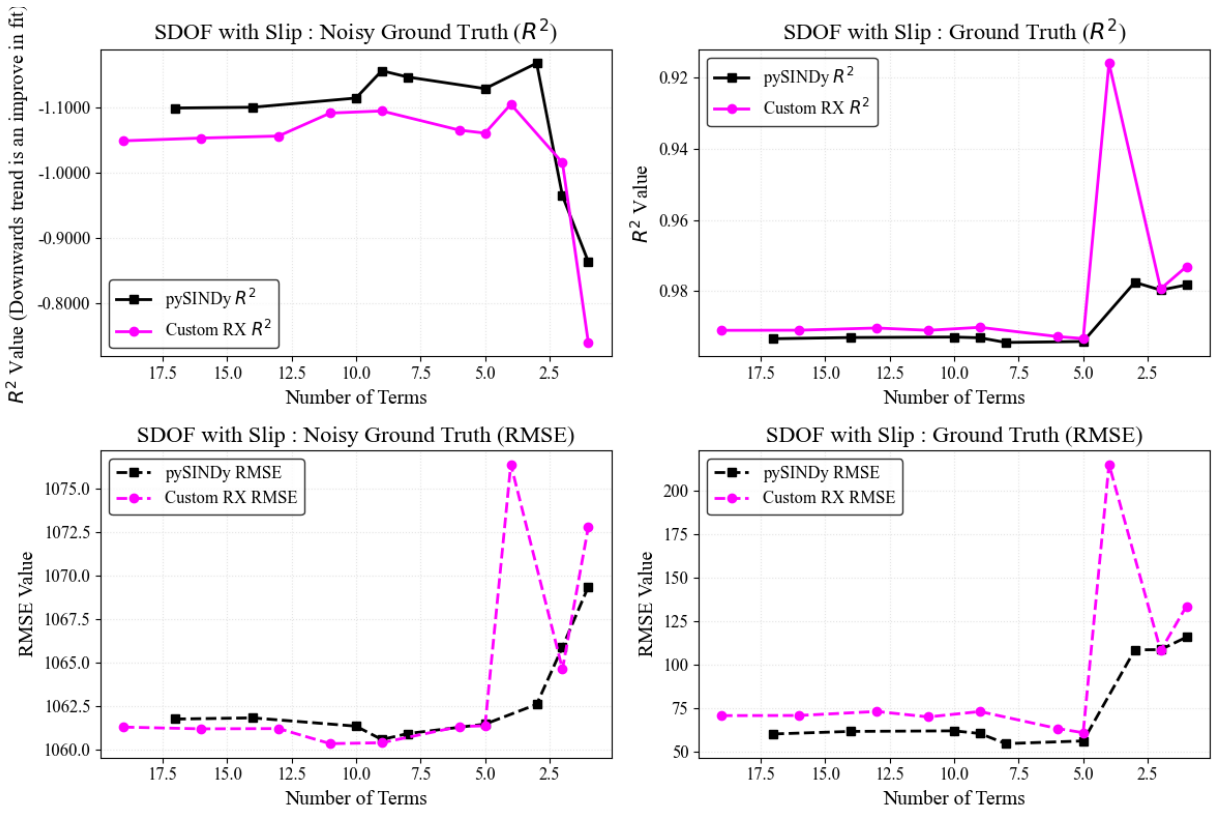


Figure M.54: Case 3.2 : Removing Unnecessary Terms

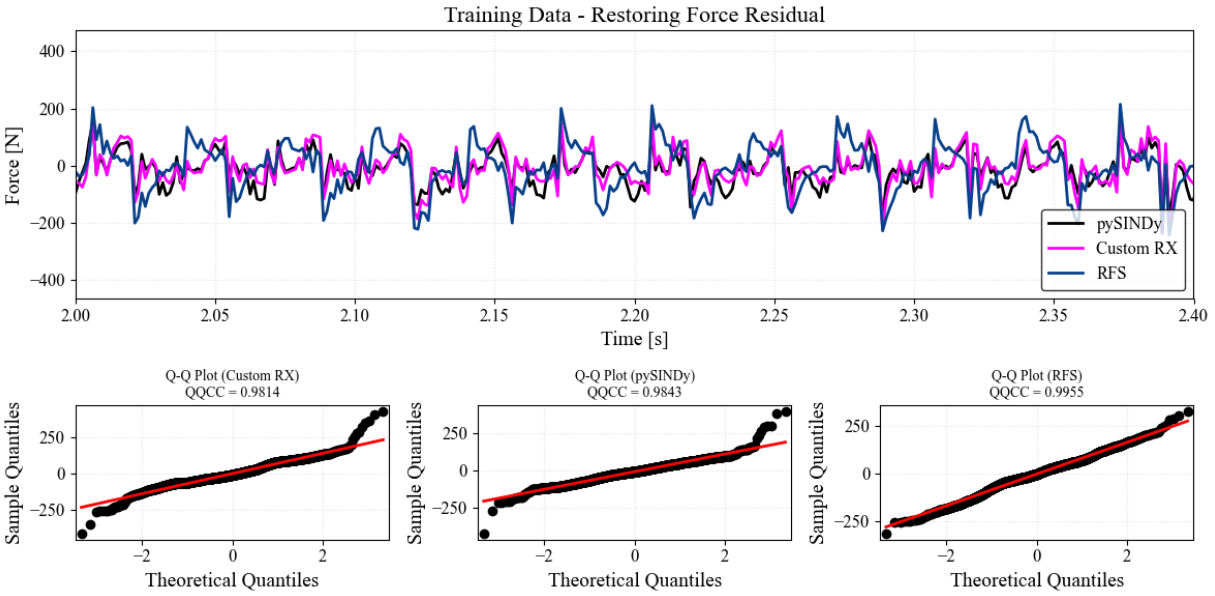
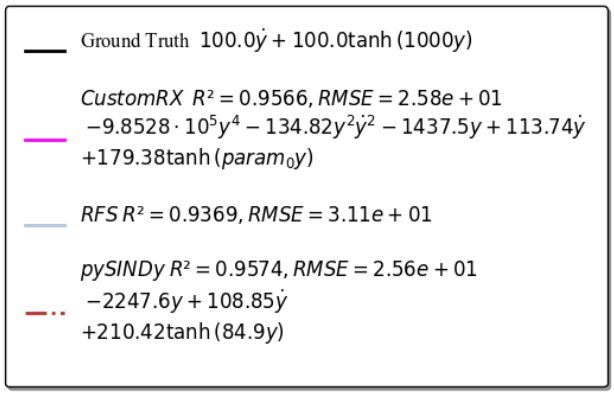


Figure M.55: Case 3.2 : Training Data - Restoring Force Residual



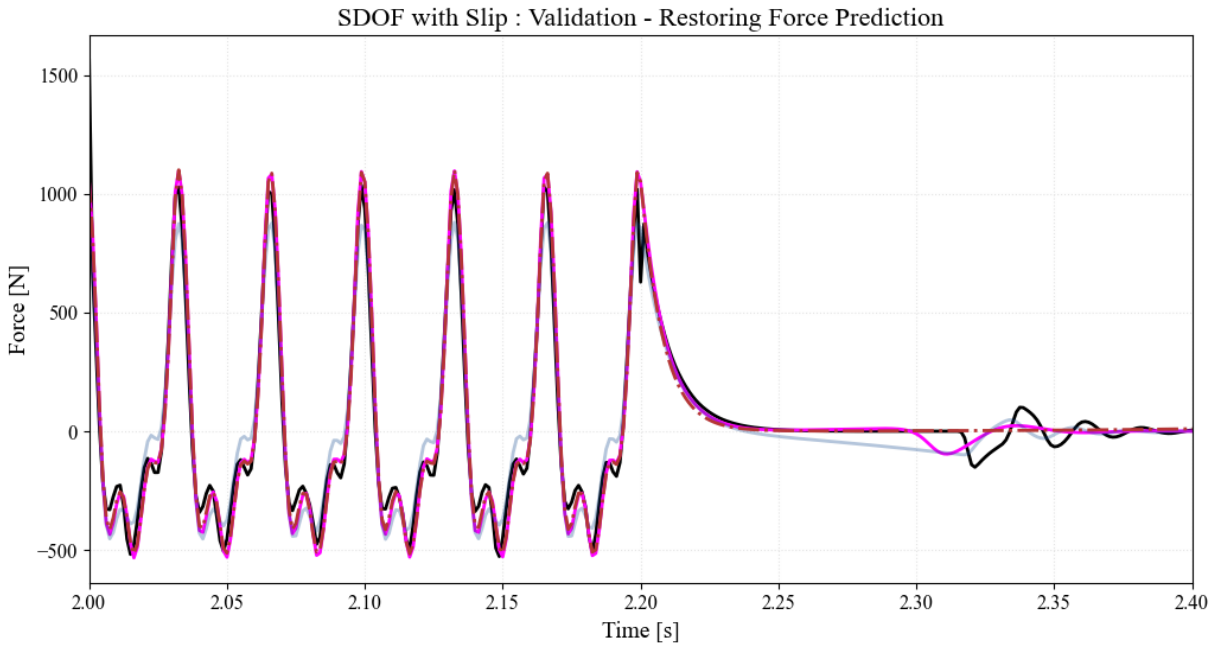


Figure M.56: Case 3.2 : Validation - Restoring Force Prediction

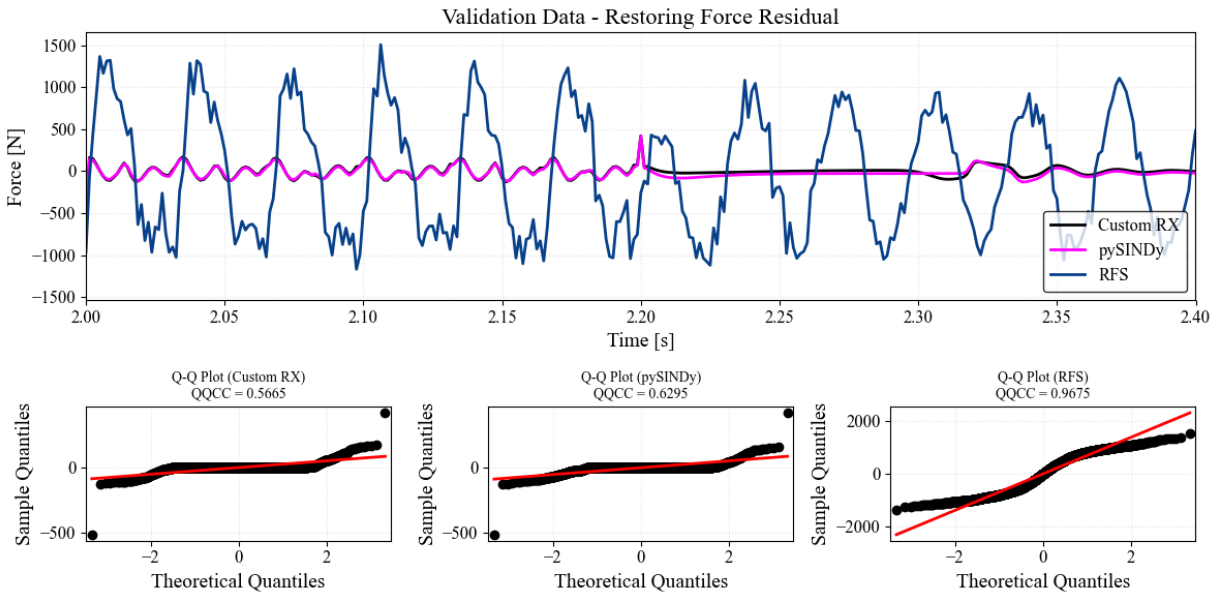


Figure M.57: Case 3.2 : Training Data - Restoring Force Residual

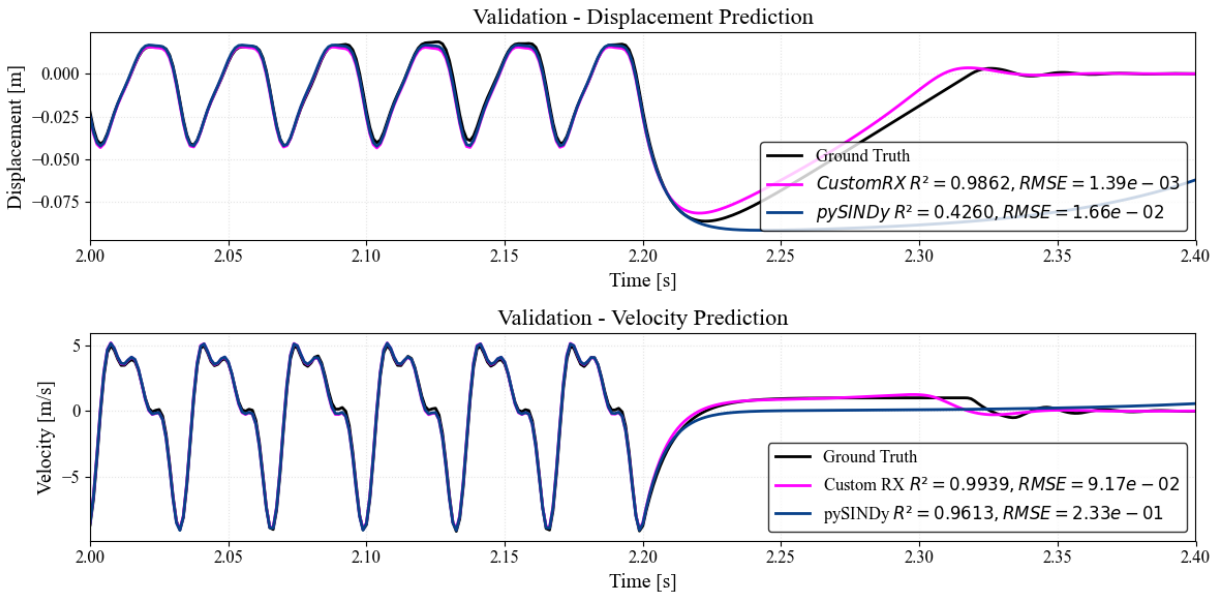


Figure M.58: Case 3.2 : Validation - Displacement & Velocity Prediction

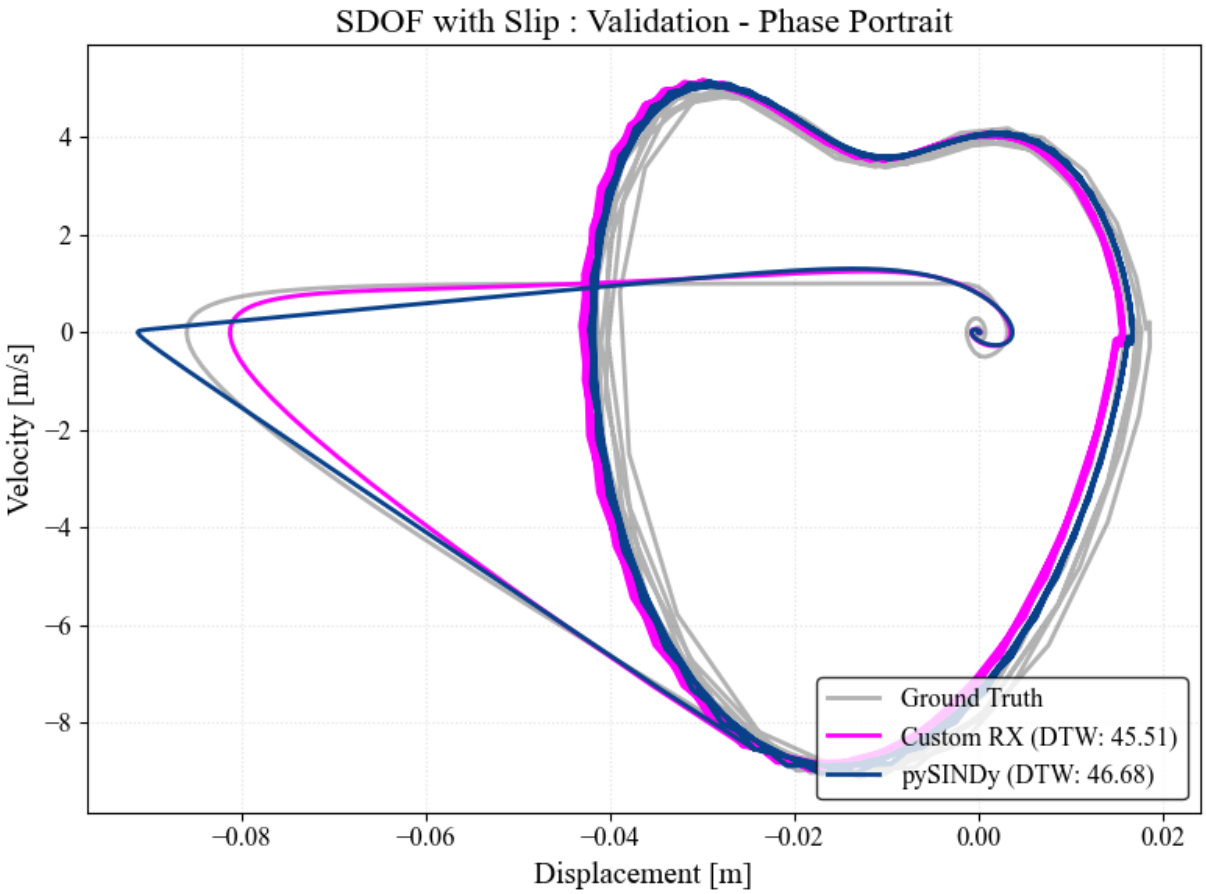


Figure M.59: Case 3.2 : Validation - Phase Portrait

N

Extended Results and Intermediate Steps in Identifying Pile Forces

Results for Shaft Force from Test 1

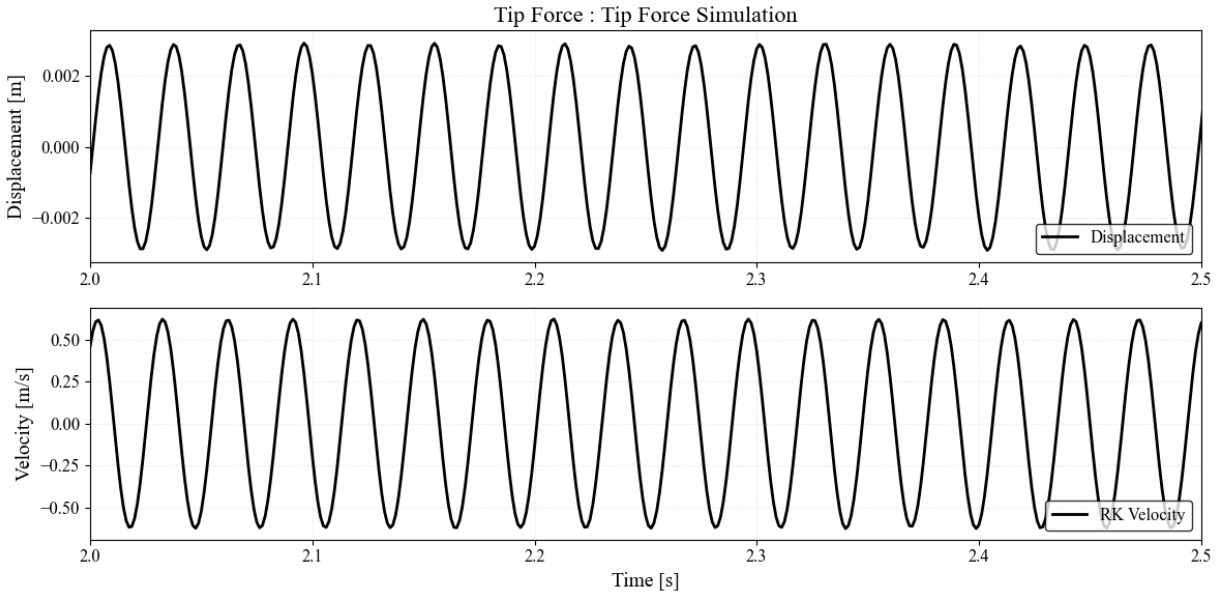


Figure N.1: Shaft Force from Test 1 - Filtered Measurements

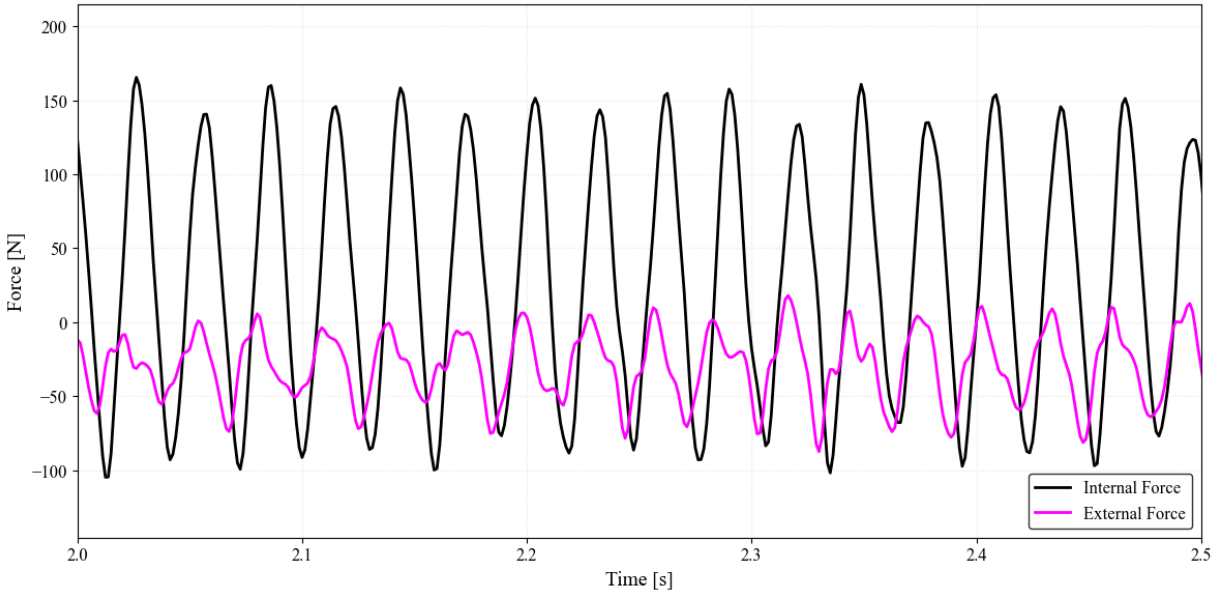


Figure N.2: Shaft Force from Test 1 External forcing & Measured Tip Force

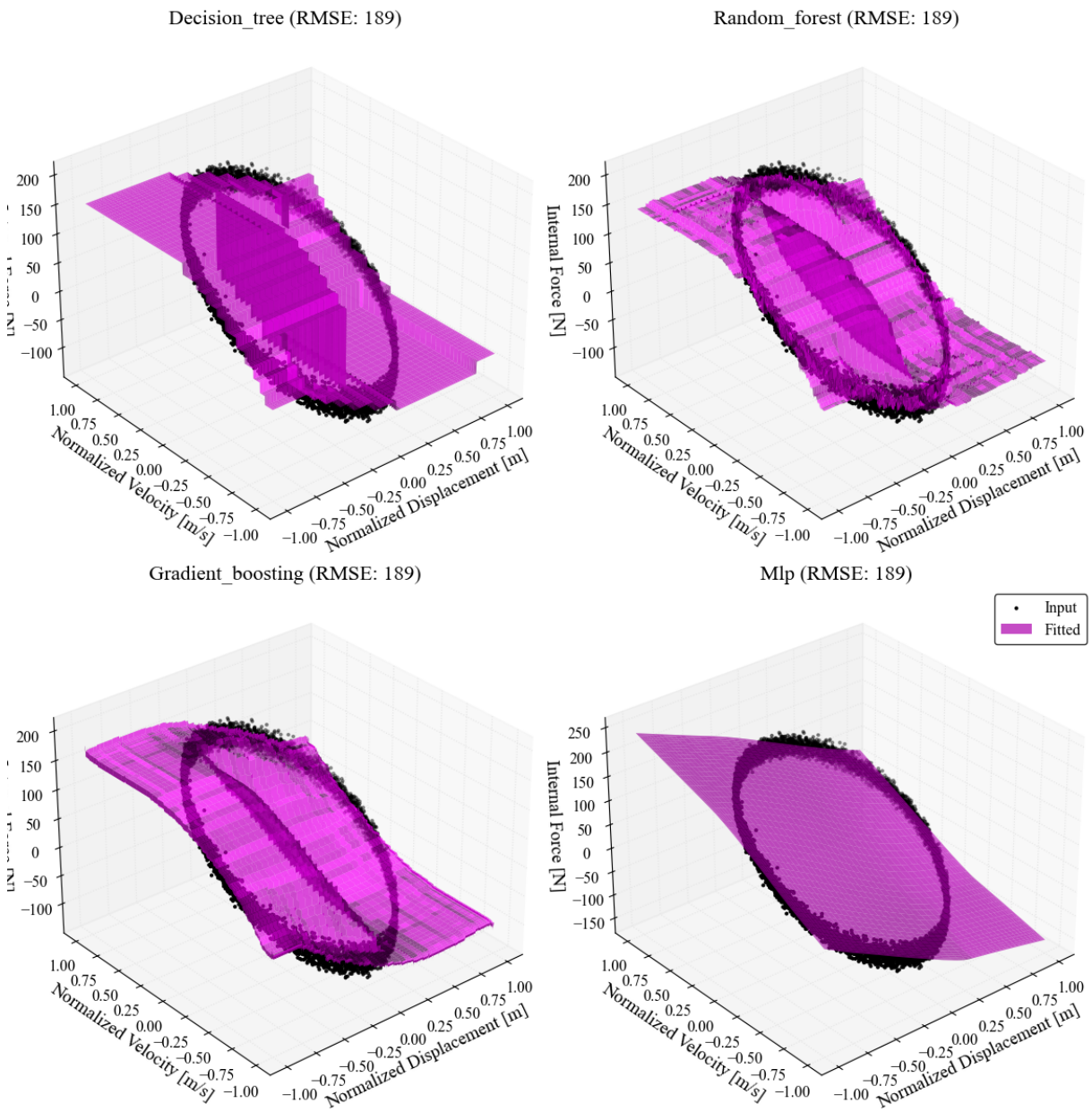


Figure N.3: Shaft Force from Test 1 : Machine Learning fitted 3D Surfaces

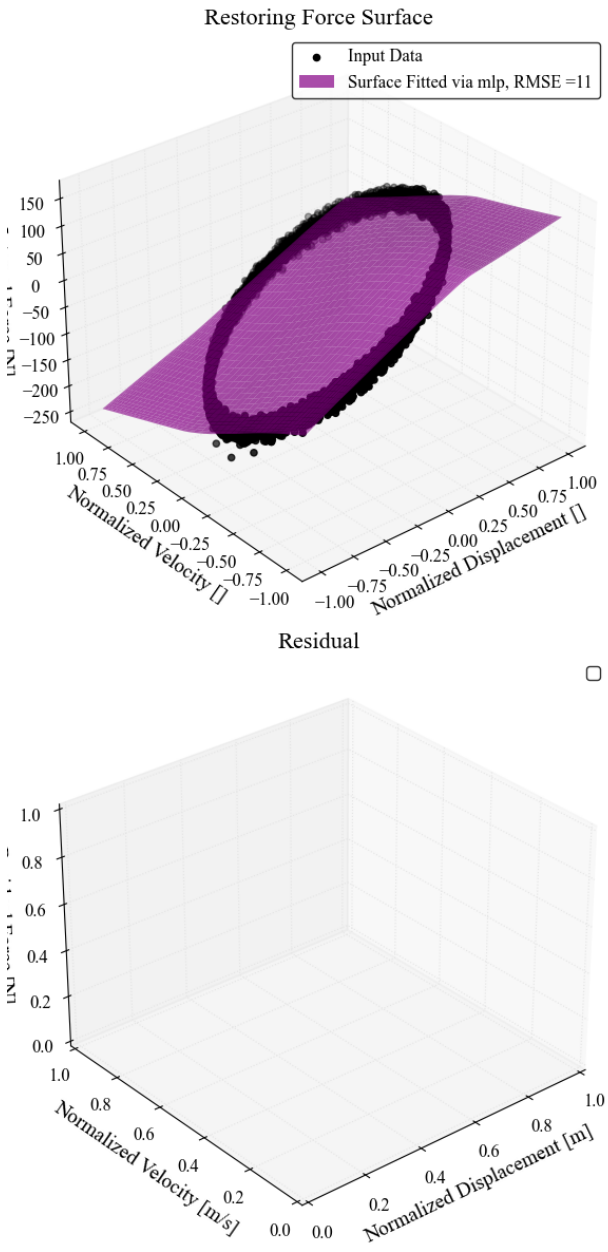


Figure N.4: Shaft Force from Test 1: Selected Restoring Force Surface

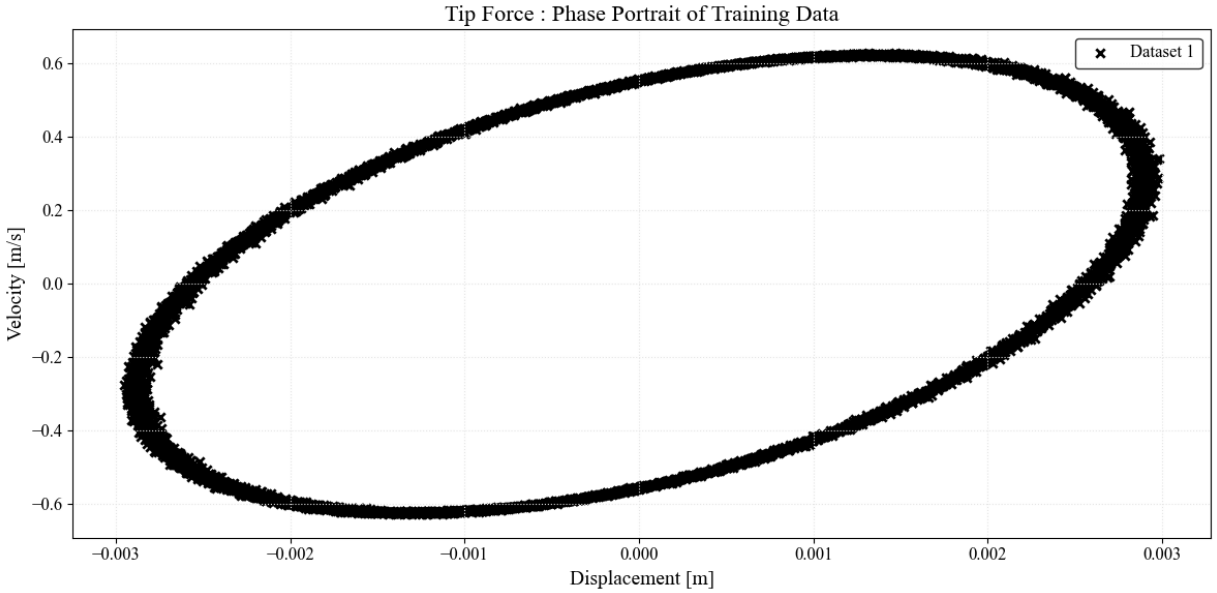


Figure N.5: Shaft Force from Test 1: Phase Portrait of Training Data

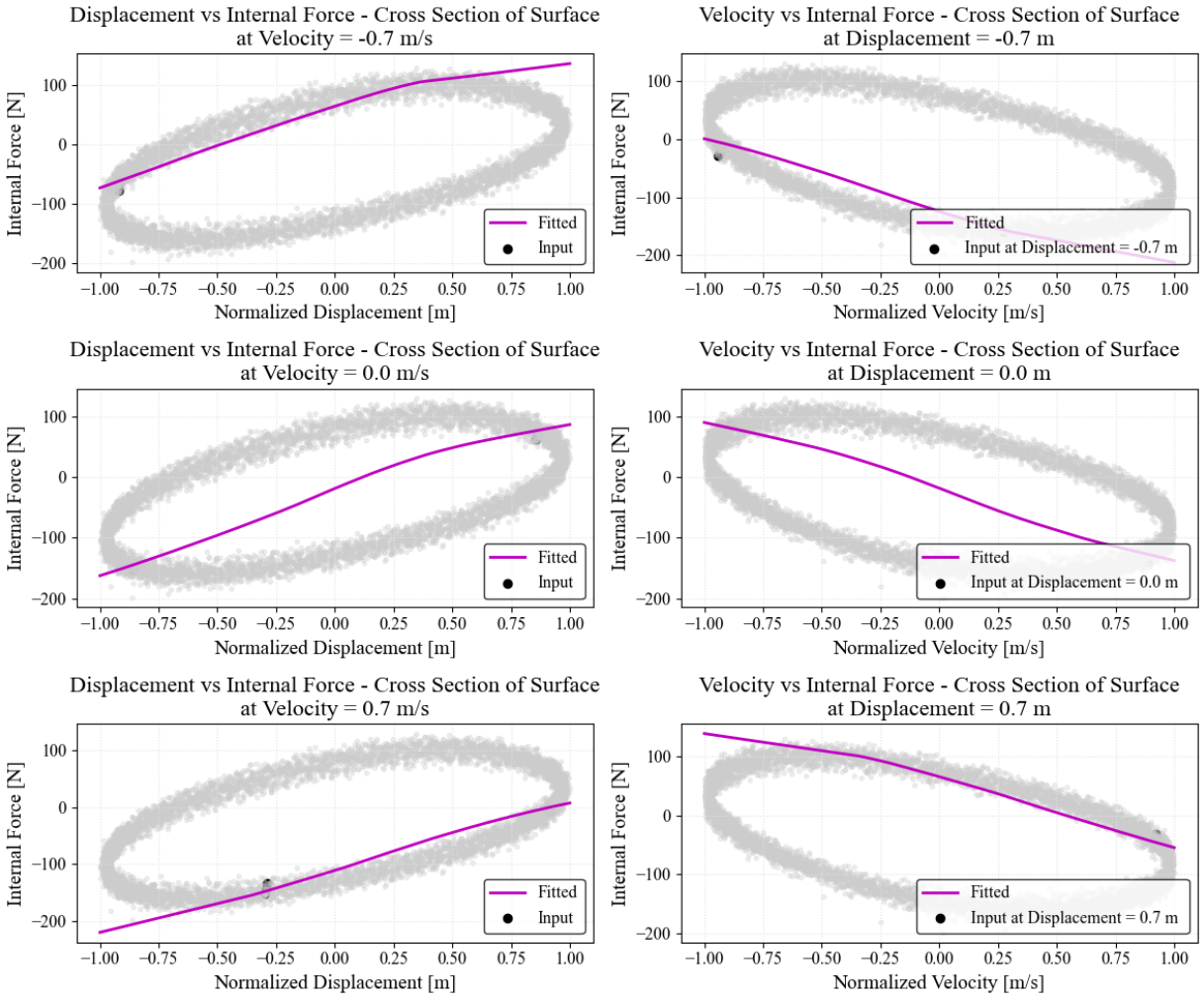


Figure N.6: Shaft Force from Test 1 : Cross Sections of Surface

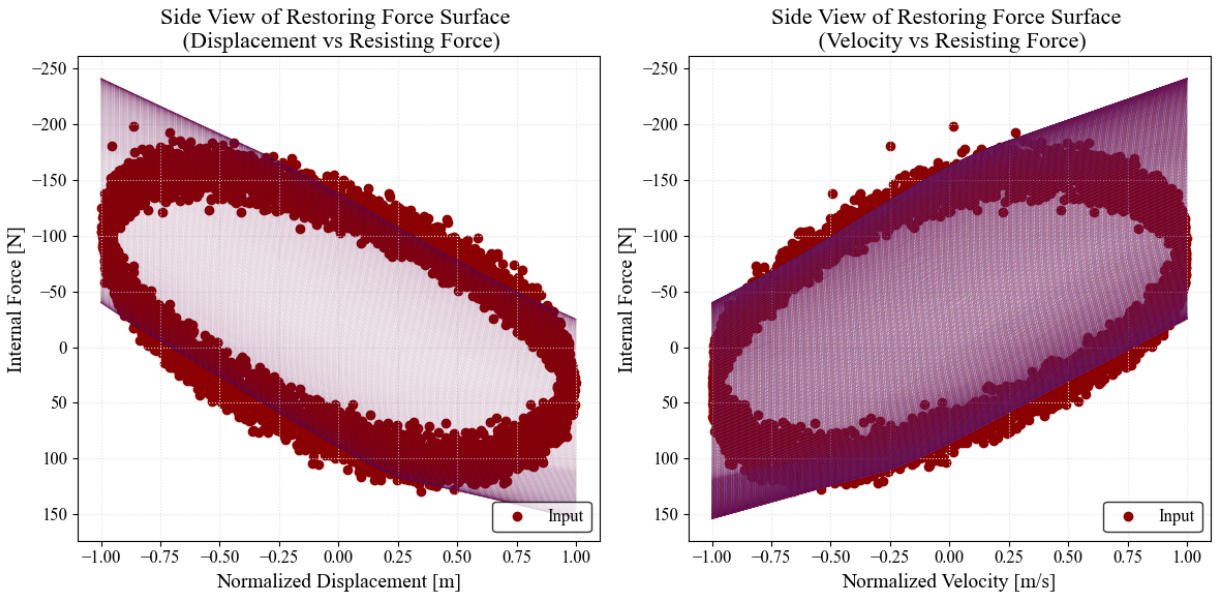


Figure N.7: Shaft Force from Test 1 : Side Views of Surface

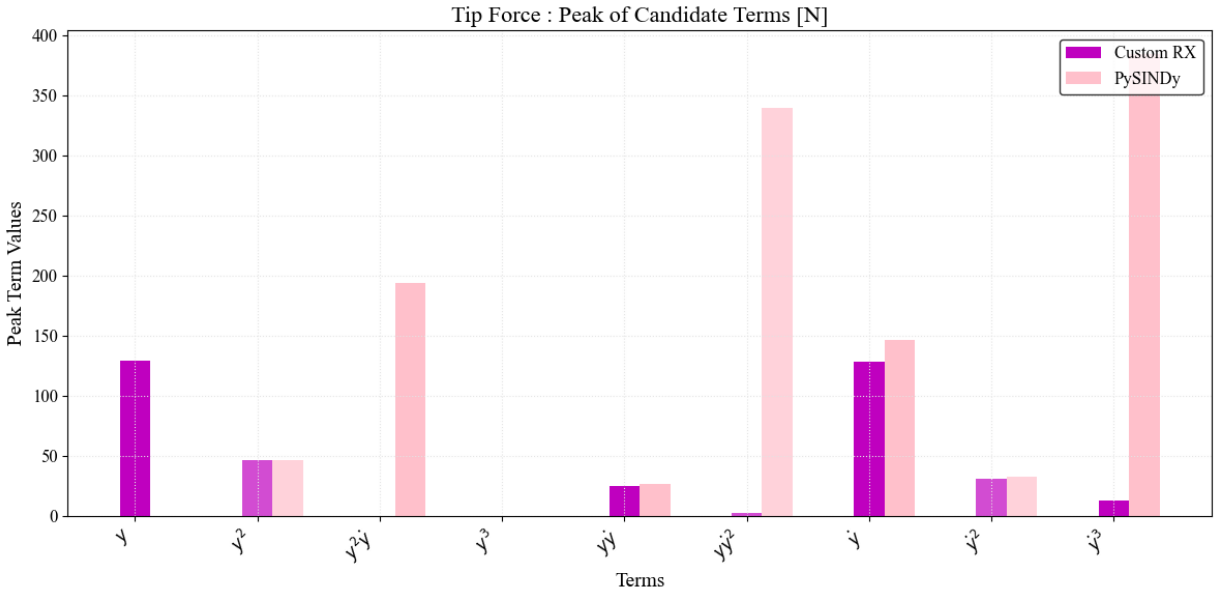


Figure N.8: Shaft Force from Test 1 : Force Features comparison

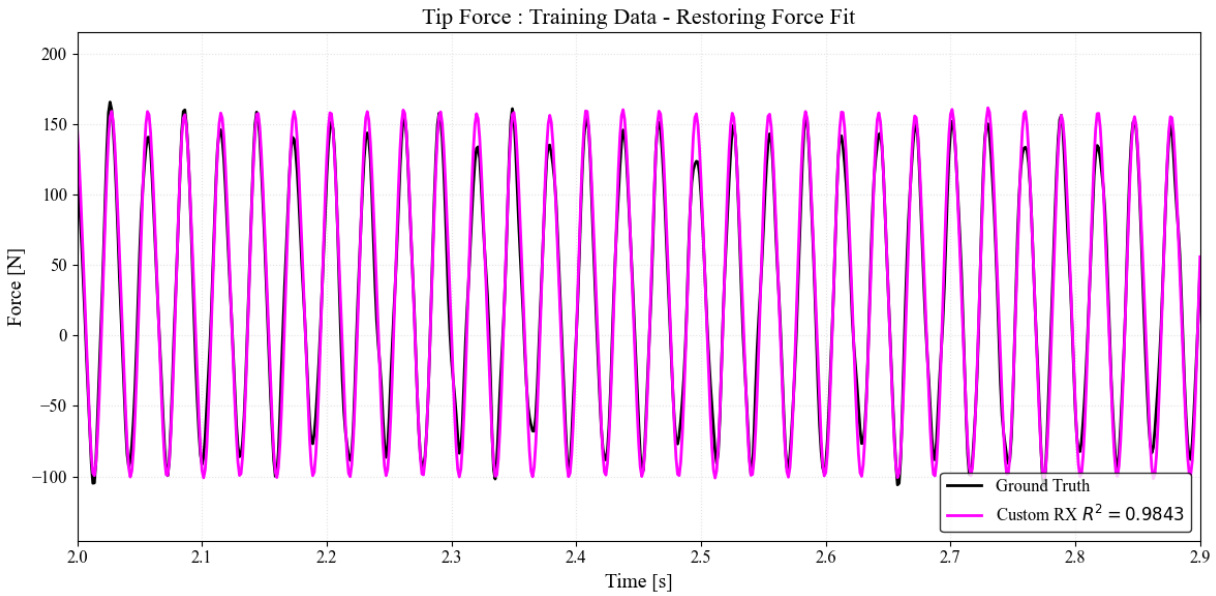


Figure N.9: Shaft Force from Test 1 : Training Data - Restoring Force Fit

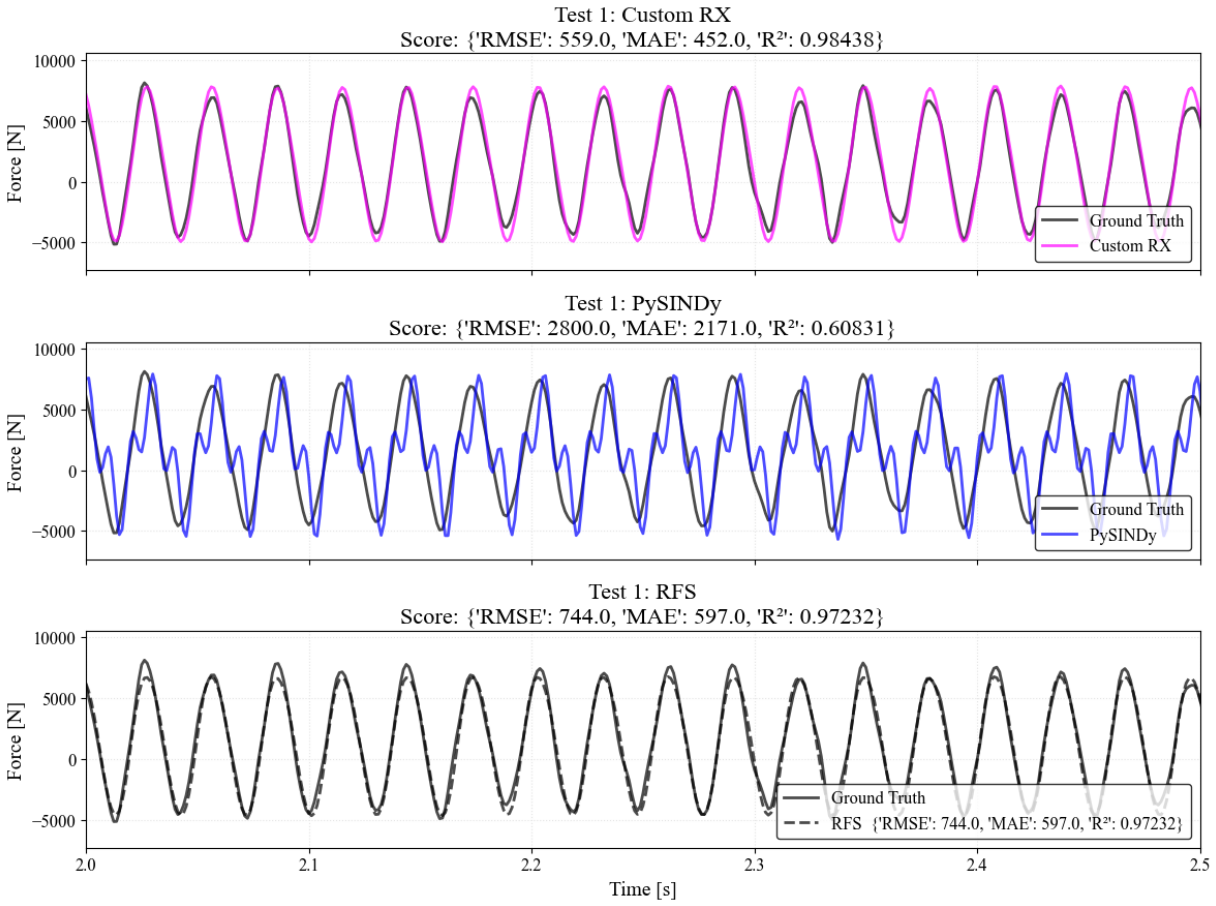


Figure N.10: Shaft Force from Test 1 : Test 1 Data - Fit for t =2-10s

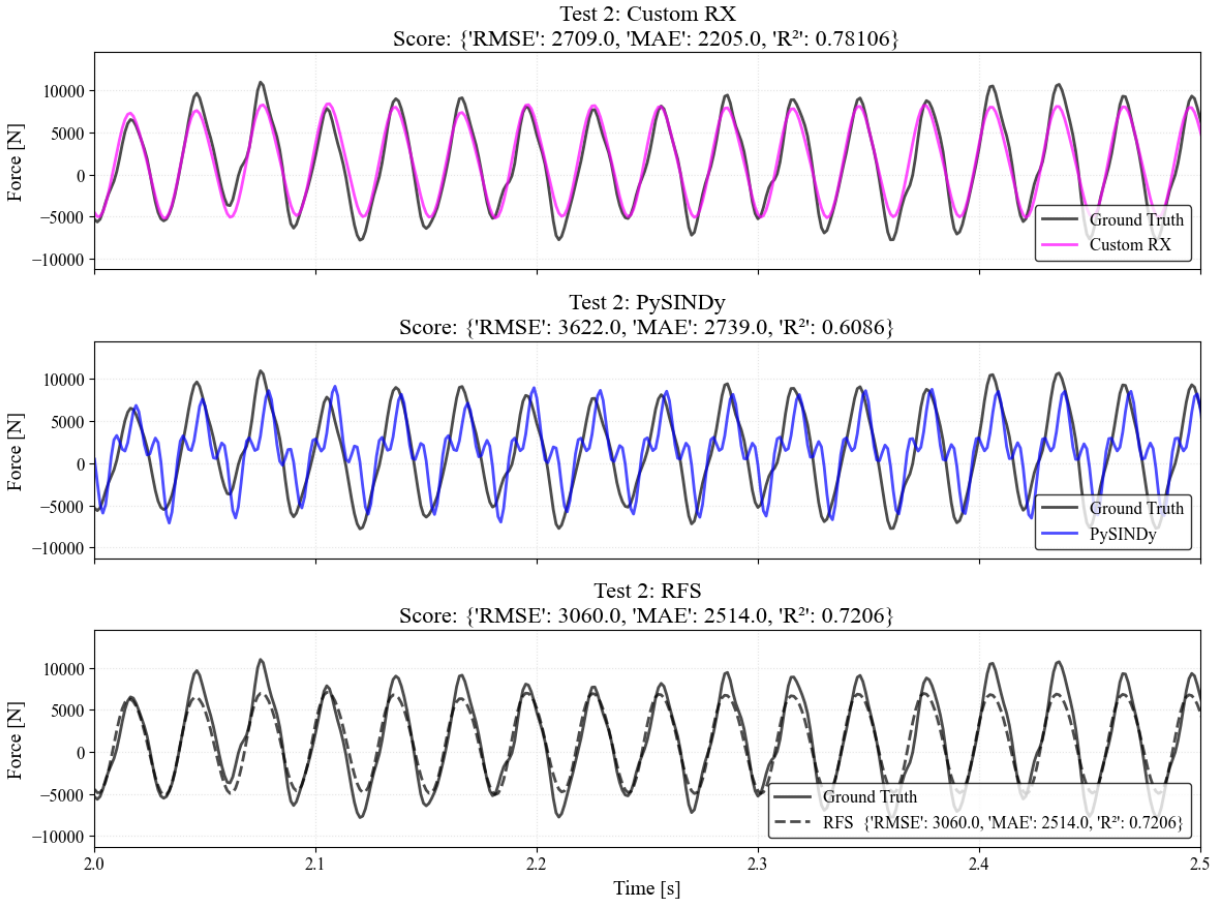


Figure N.11: Shaft Force from Test 1 : Test 2 Data - Fit for t =2-10s

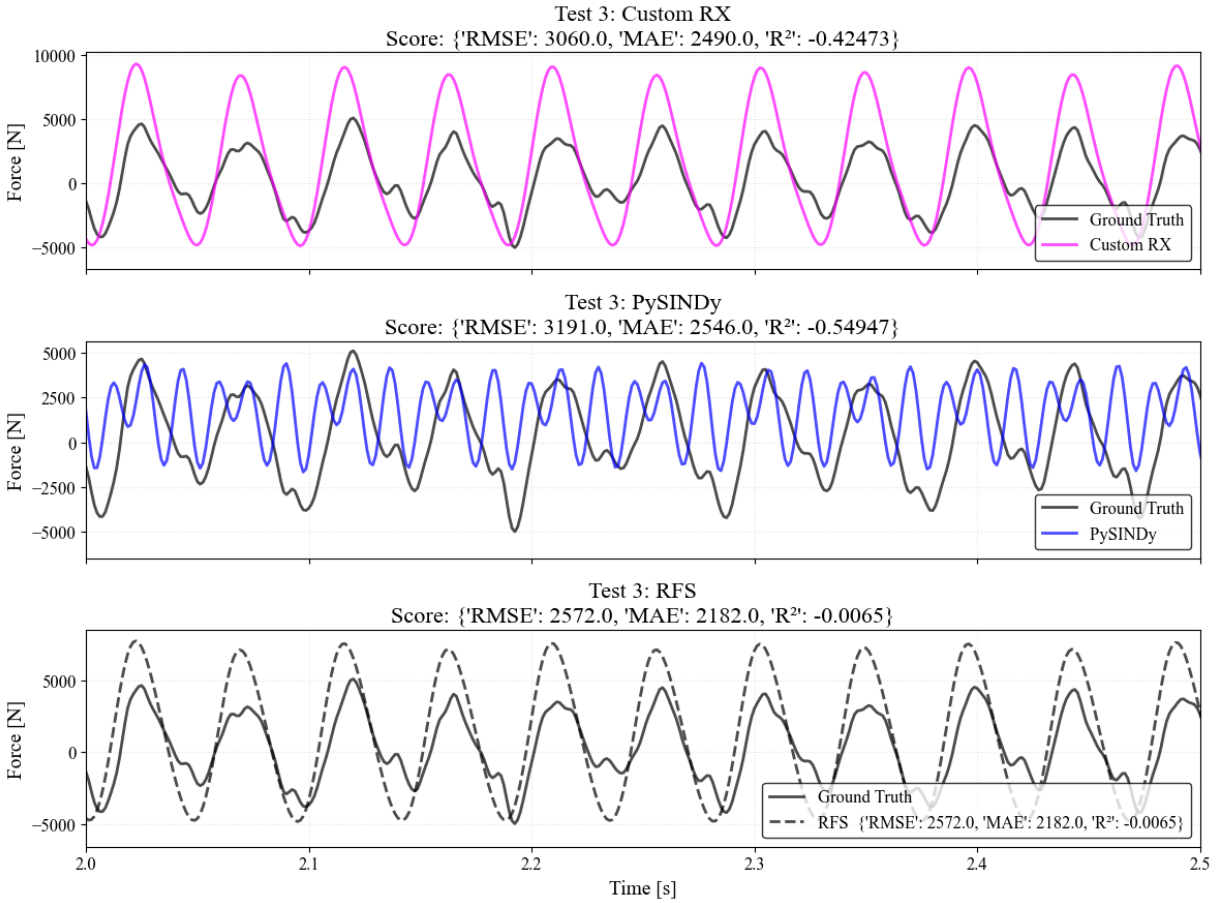


Figure N.12: Shaft Force from Test 1 : Test 3 Data - Fit for t =2-10s

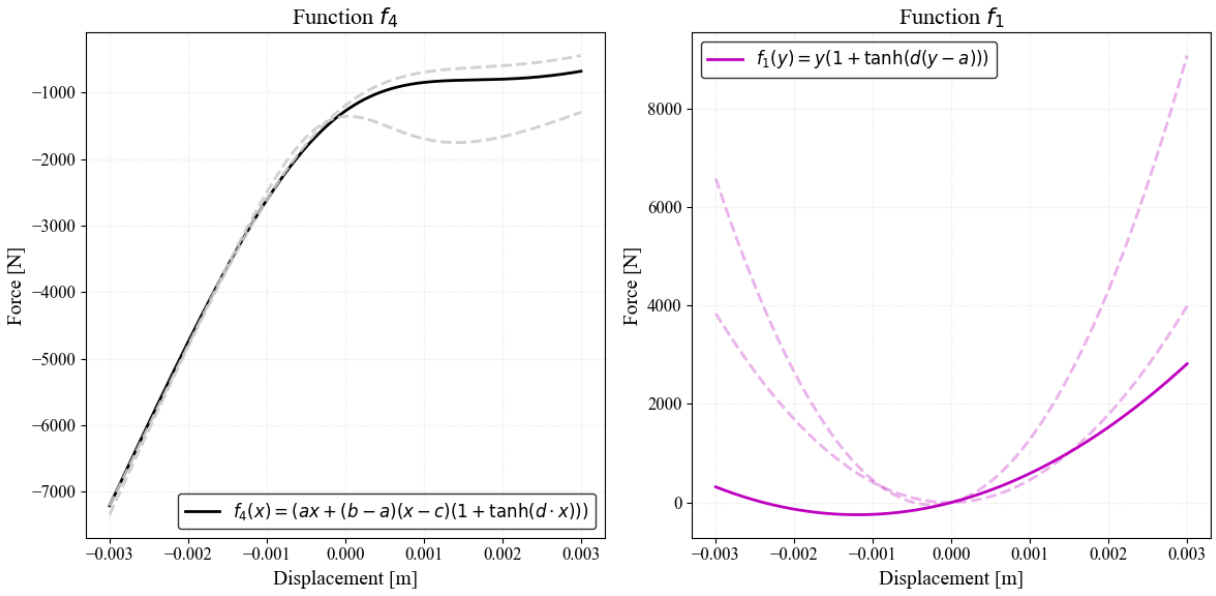


Figure N.13: Shaft Force from Test 1 : Test 4 Data - Fit for t =2-10s

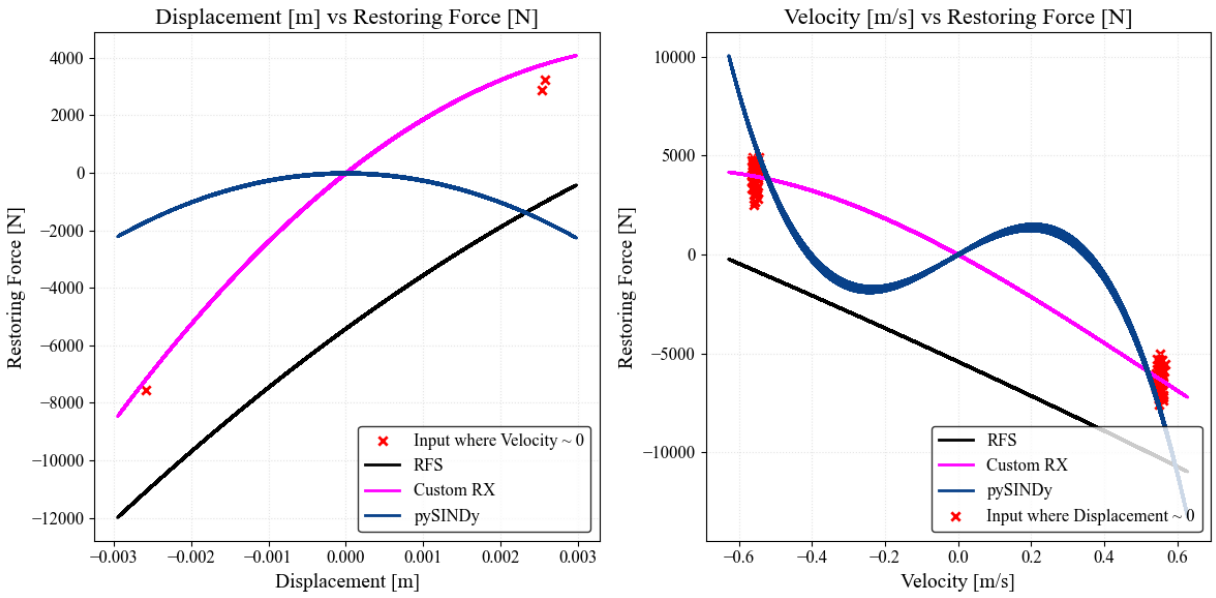


Figure N.14: Shaft Force from Test 1 : Test 1 Data - Fit for t =2-40s

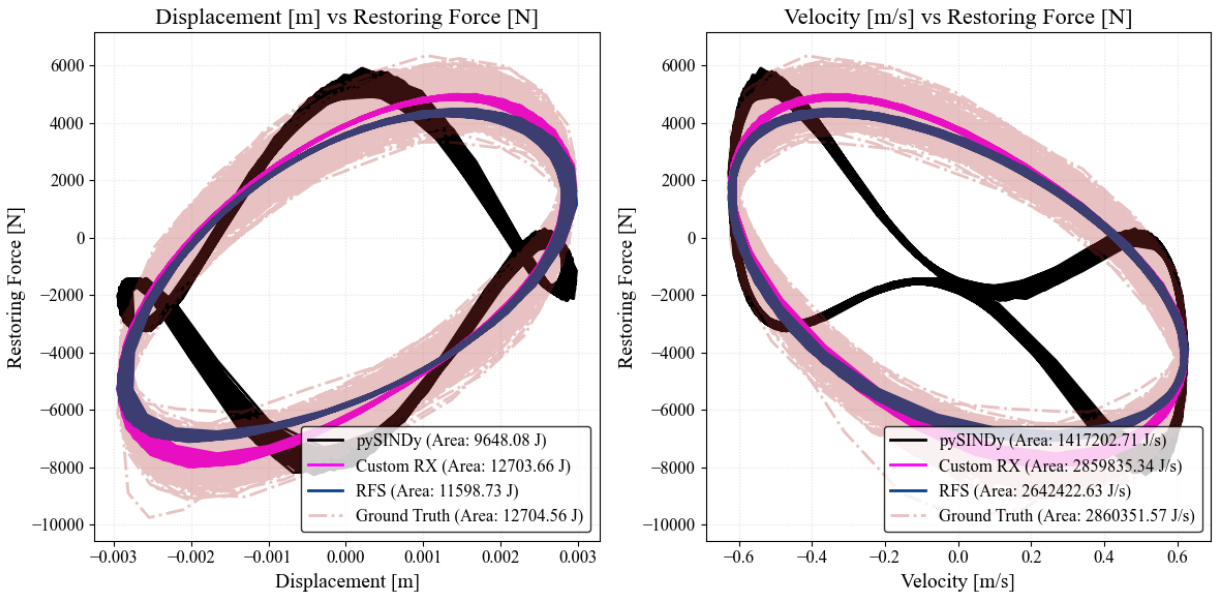


Figure N.15: Shaft Force from Test 1 : Test 2 Data - Fit for t =2-40s

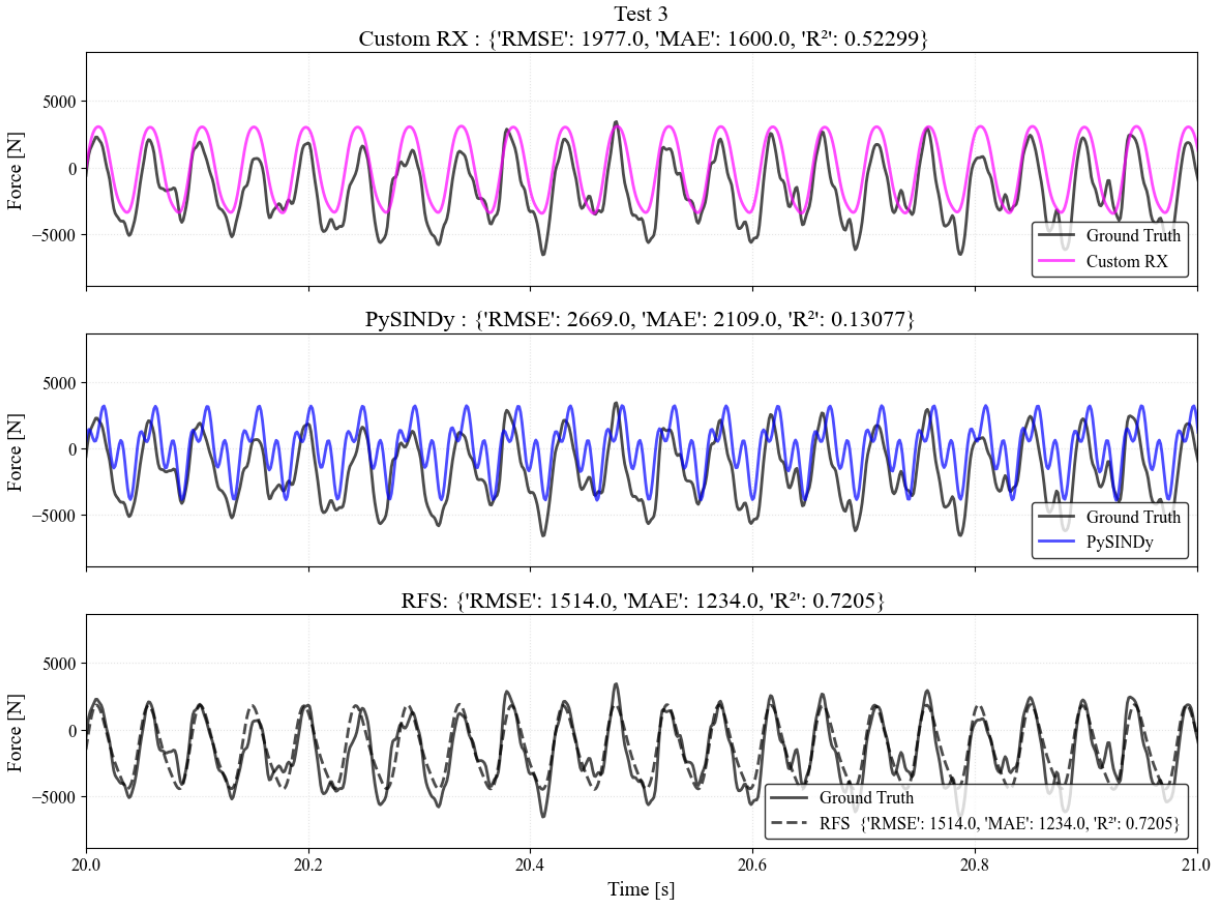


Figure N.16: Shaft Force from Test 1 : Test 3 Data - Fit for t =2-40s

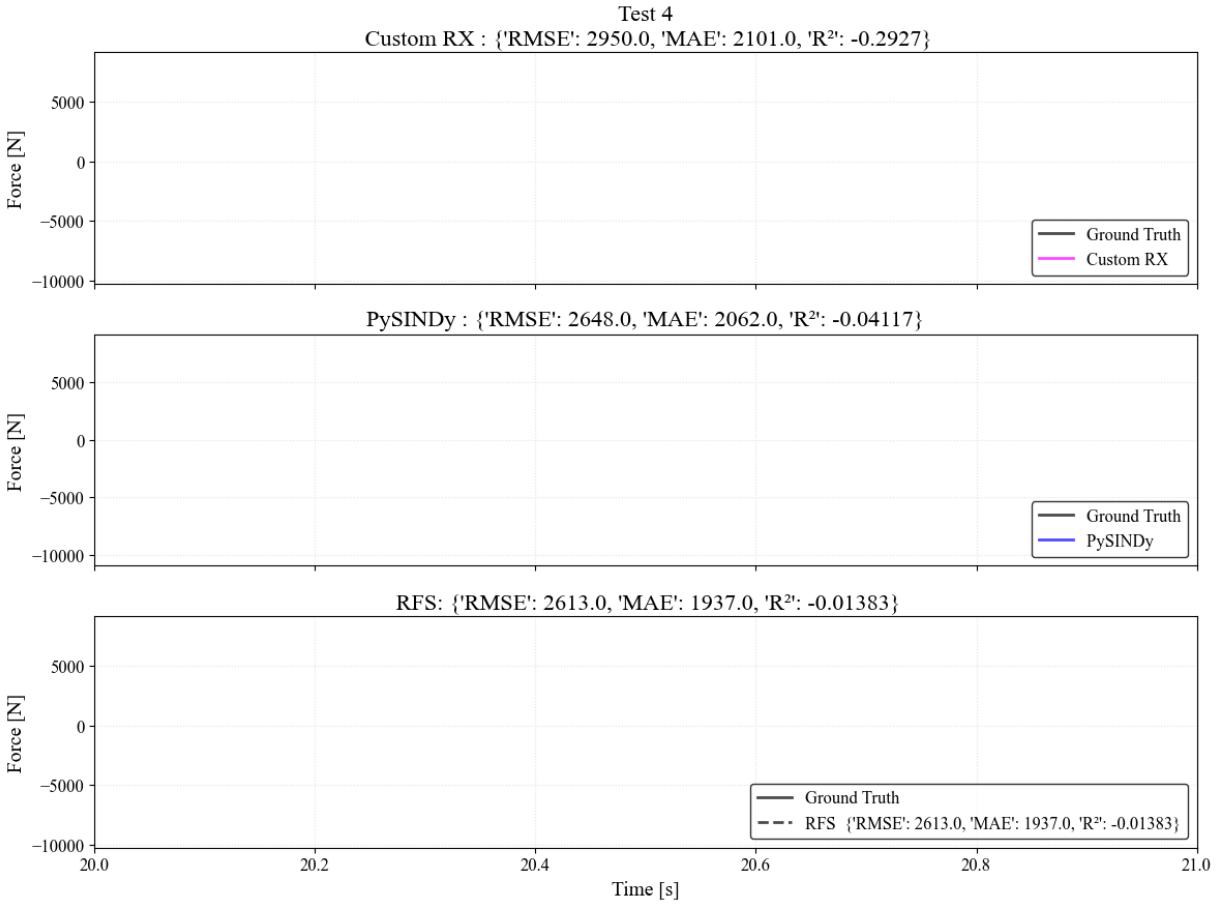


Figure N.17: Shaft Force from Test 1 : Test 4 Data - Fit for t =2-40s

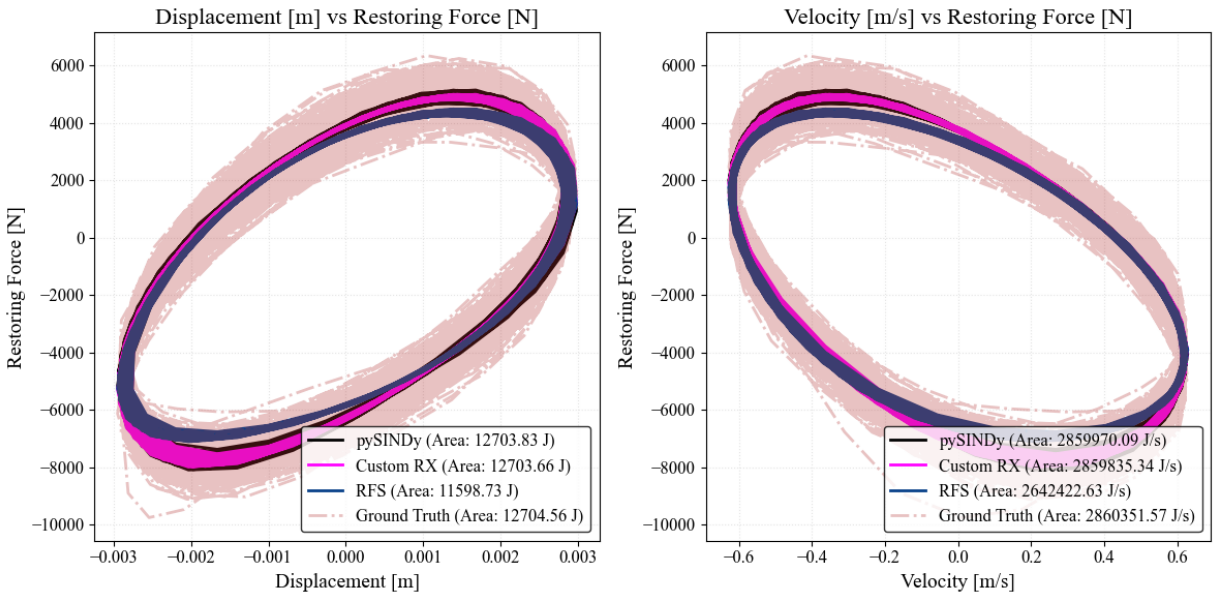


Figure N.18: Shaft Force from Test 1: Fitted loading Cycles

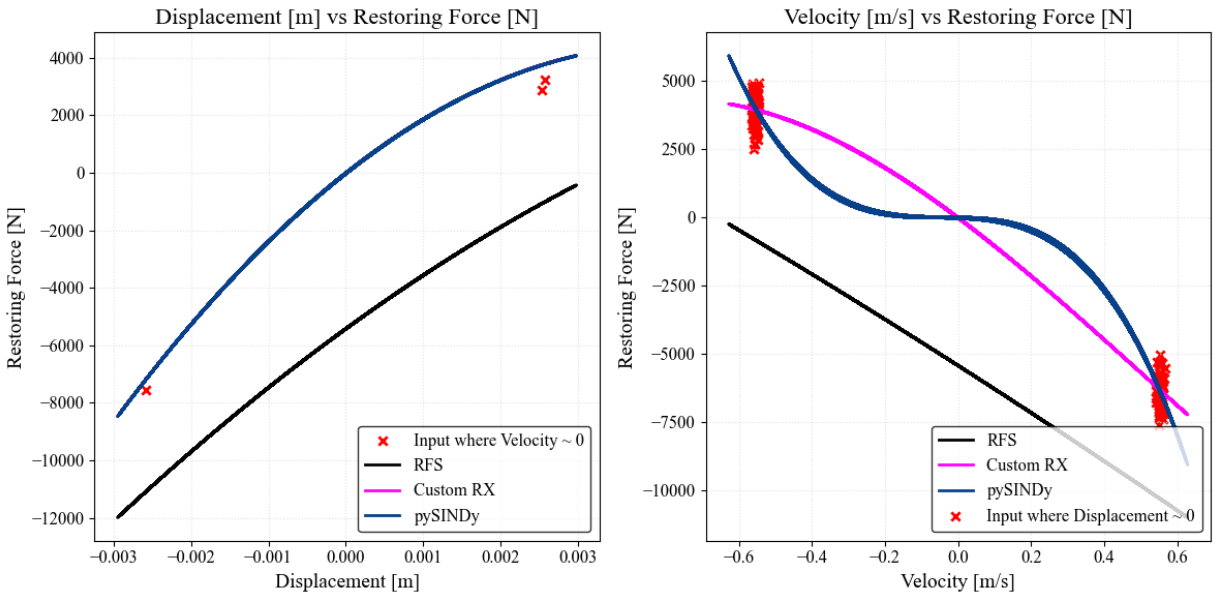


Figure N.19: Shaft Force from Test 1: Fitted loading Cycles - Section

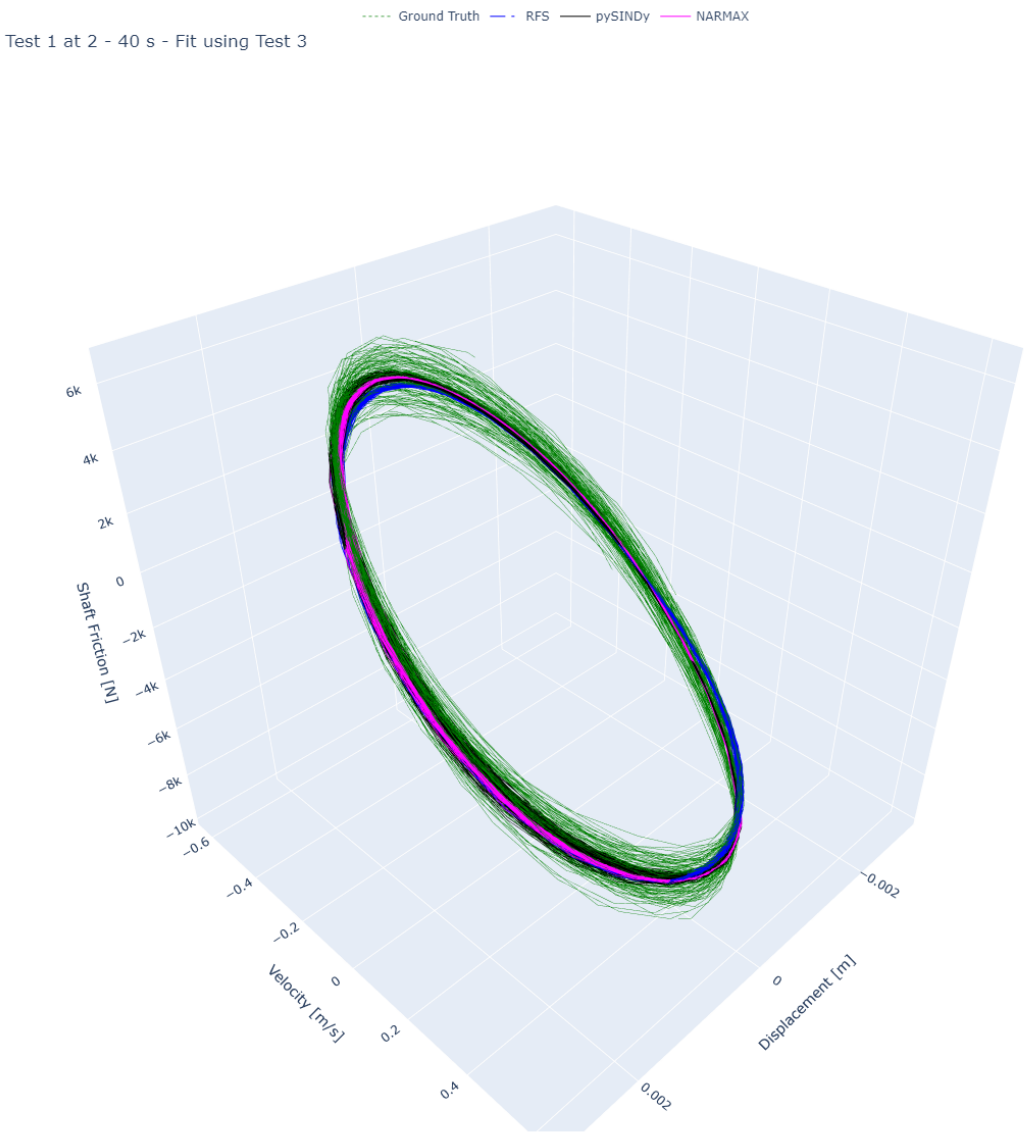


Figure N.20: 3D View-1 of shaft force vs displacement and velocity for shaft force from Test 1

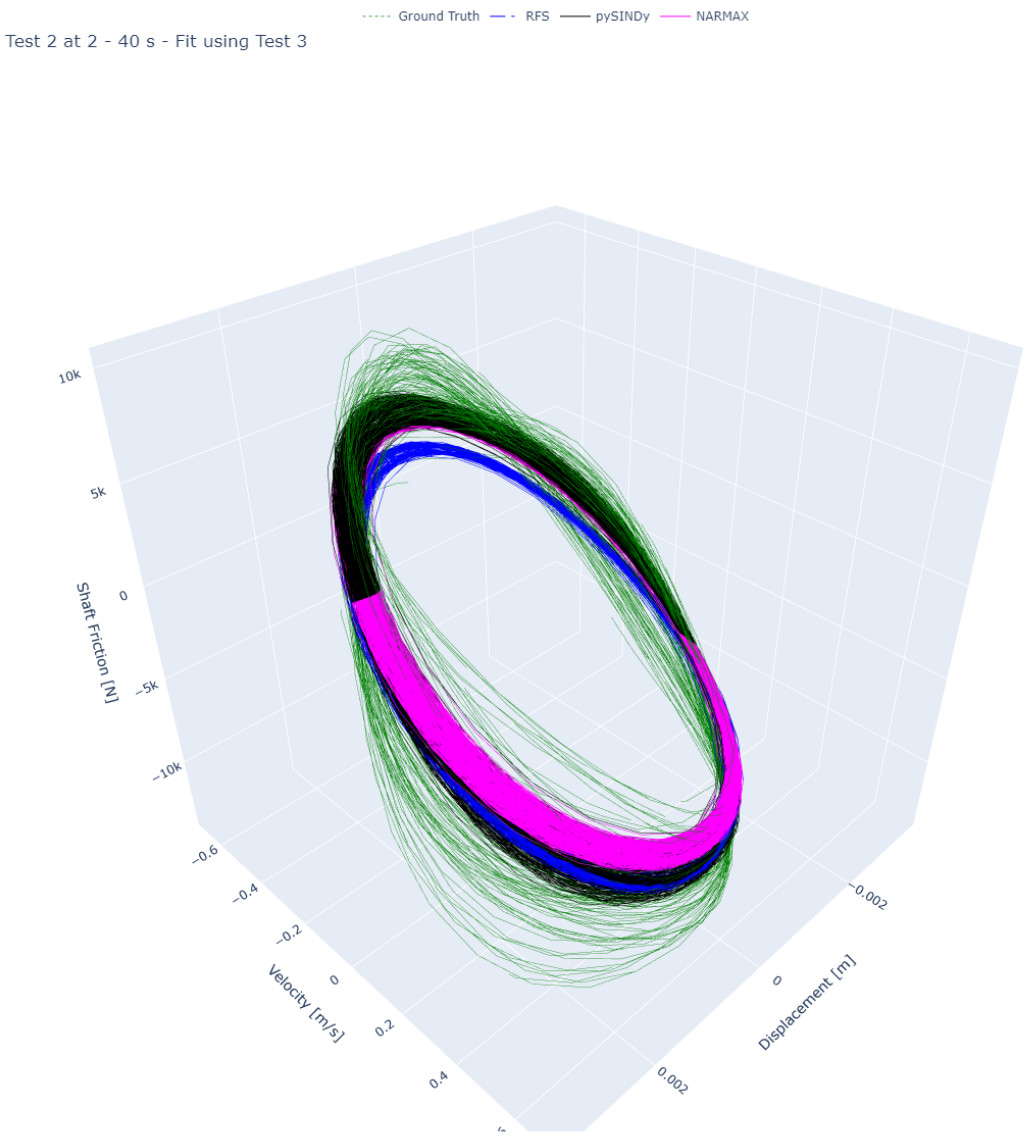


Figure N.21: 3D View-2 of shaft force vs displacement and velocity for shaft force from Test 1

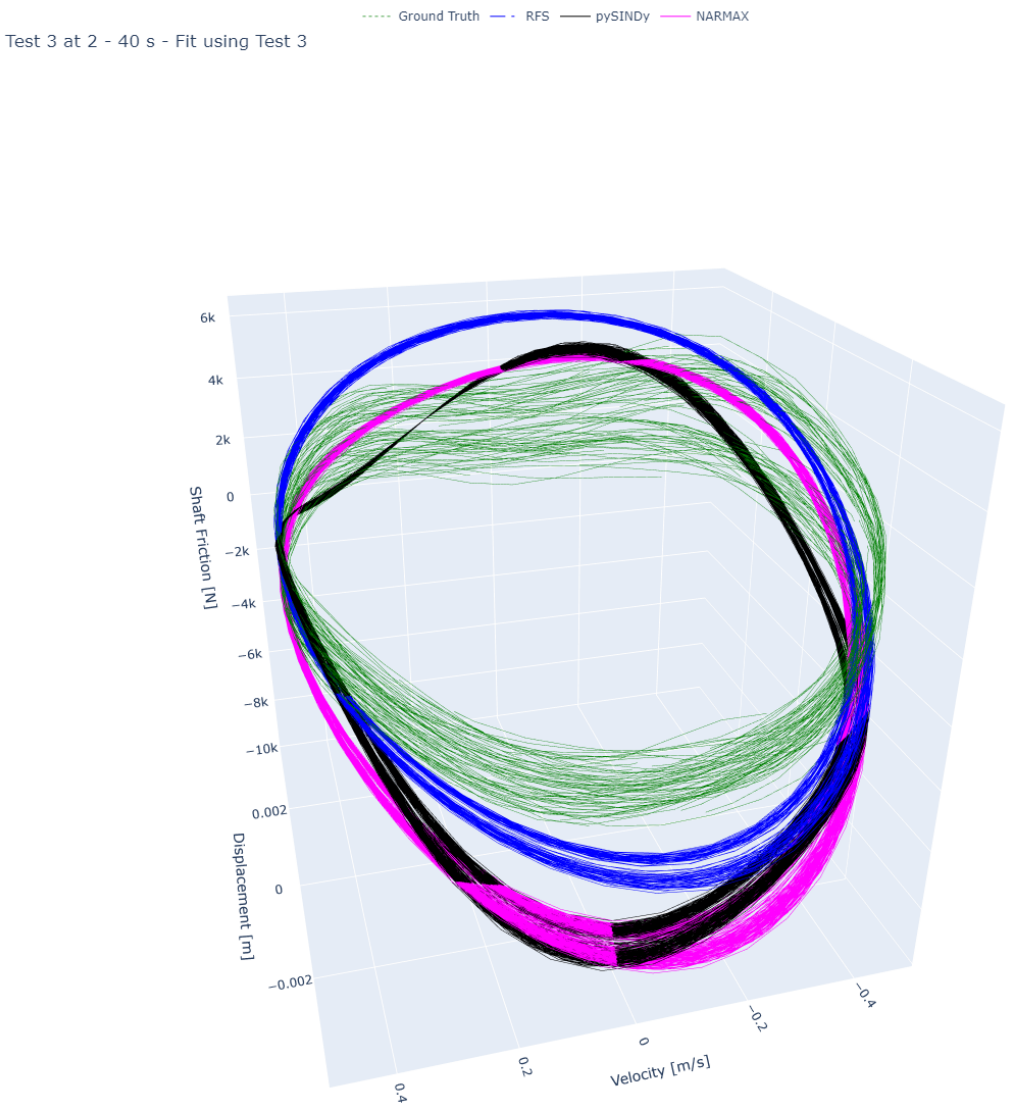


Figure N.22: 3D View-3 of shaft force vs displacement and velocity for shaft force from Test 1

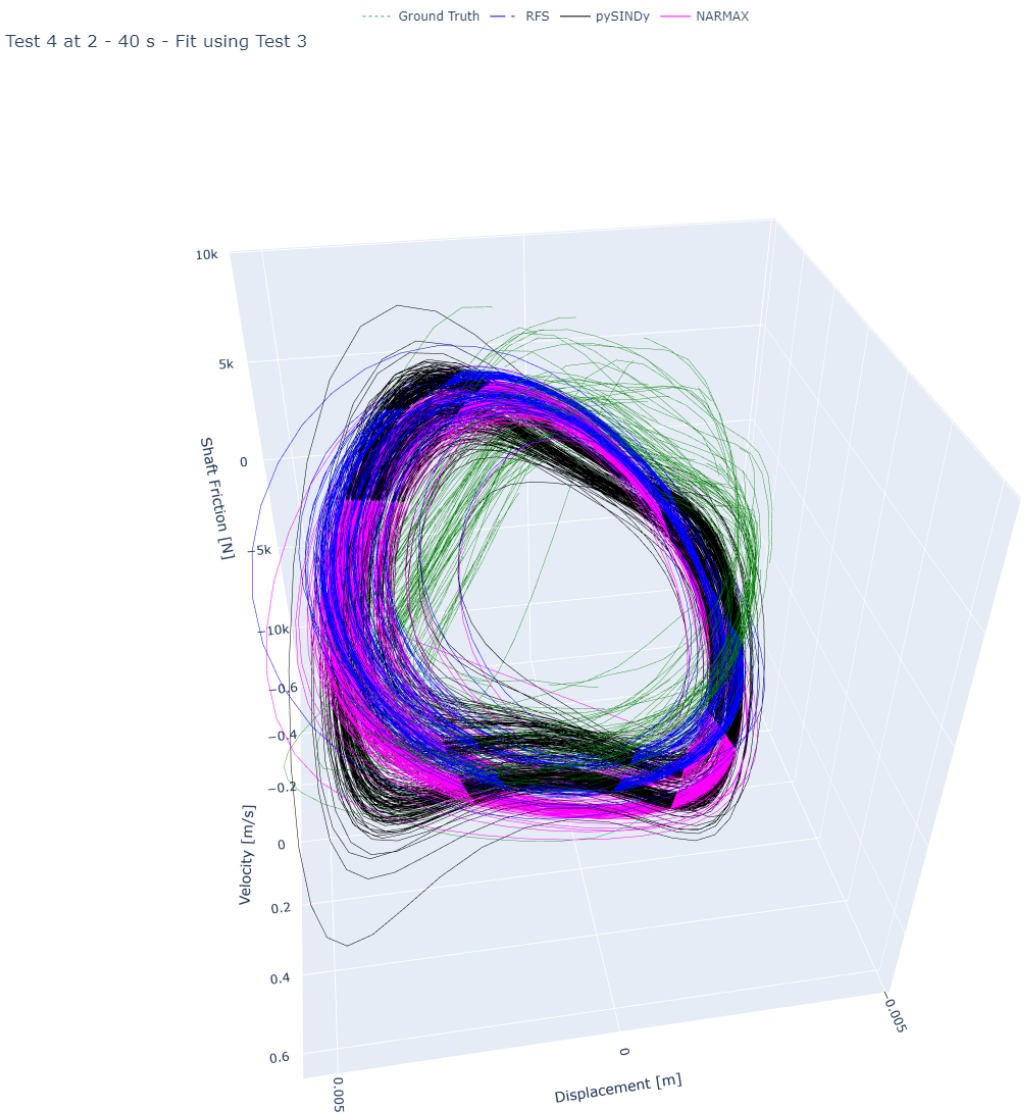


Figure N.23: 3D View-4 of shaft force vs displacement and velocity for shaft force from Test 1

Results for Shaft Force from Test 3

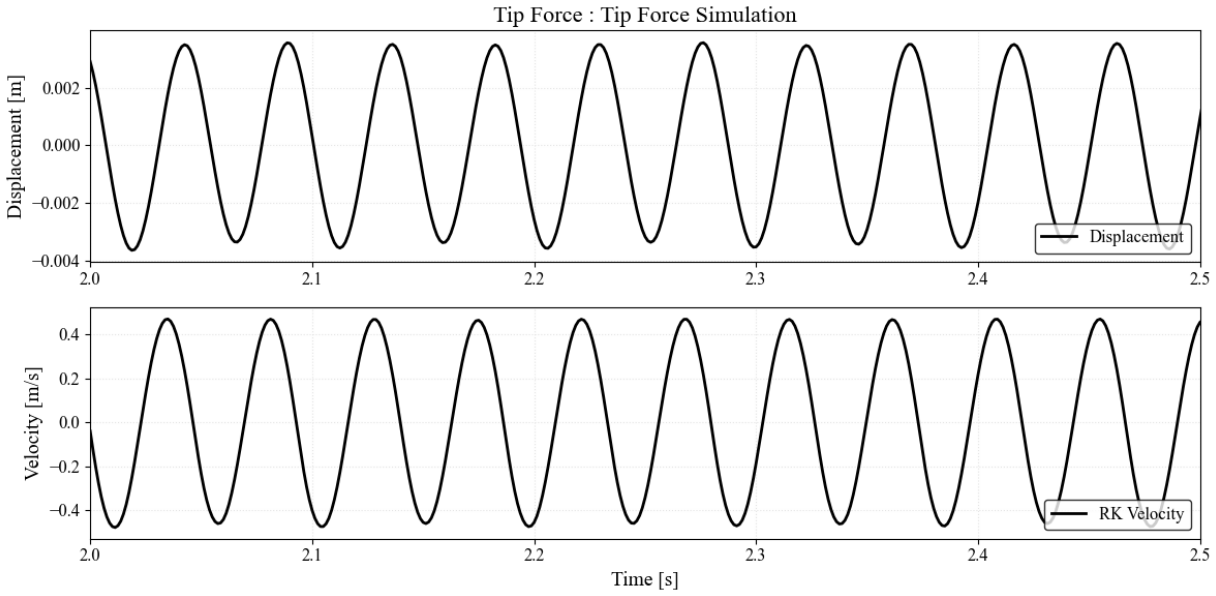


Figure N.24: Shaft Force from Test 3 - Filtered Measurements

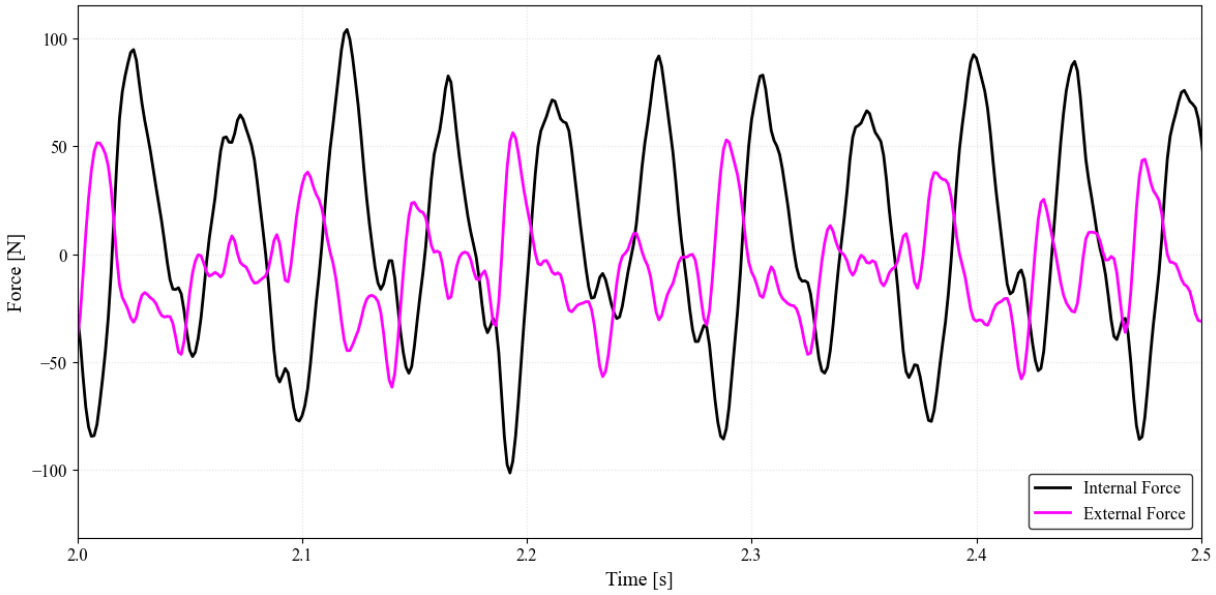


Figure N.25: Shaft Force from Test 3 External forcing & Measured Tip Force

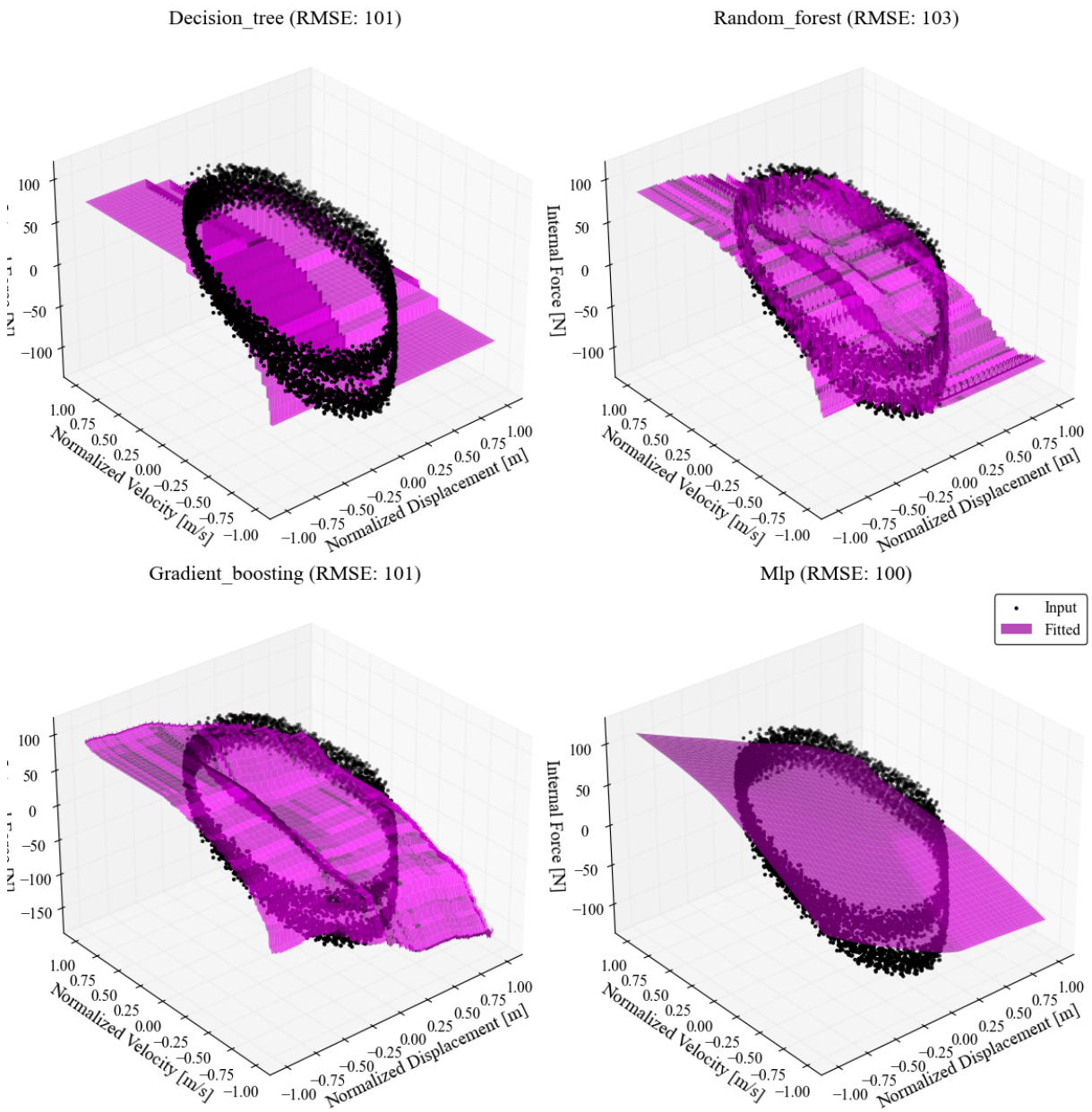


Figure N.26: Shaft Force from Test 3 : Machine Learning fitted 3D Surfaces

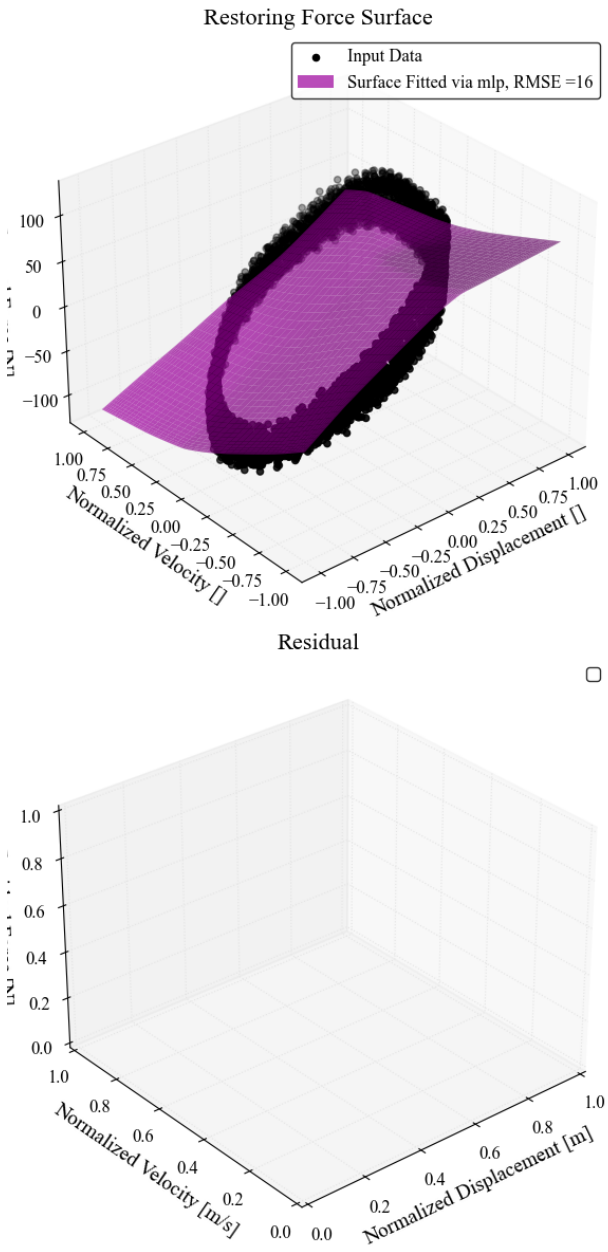


Figure N.27: Shaft Force from Test 3: Selected Restoring Force Surface

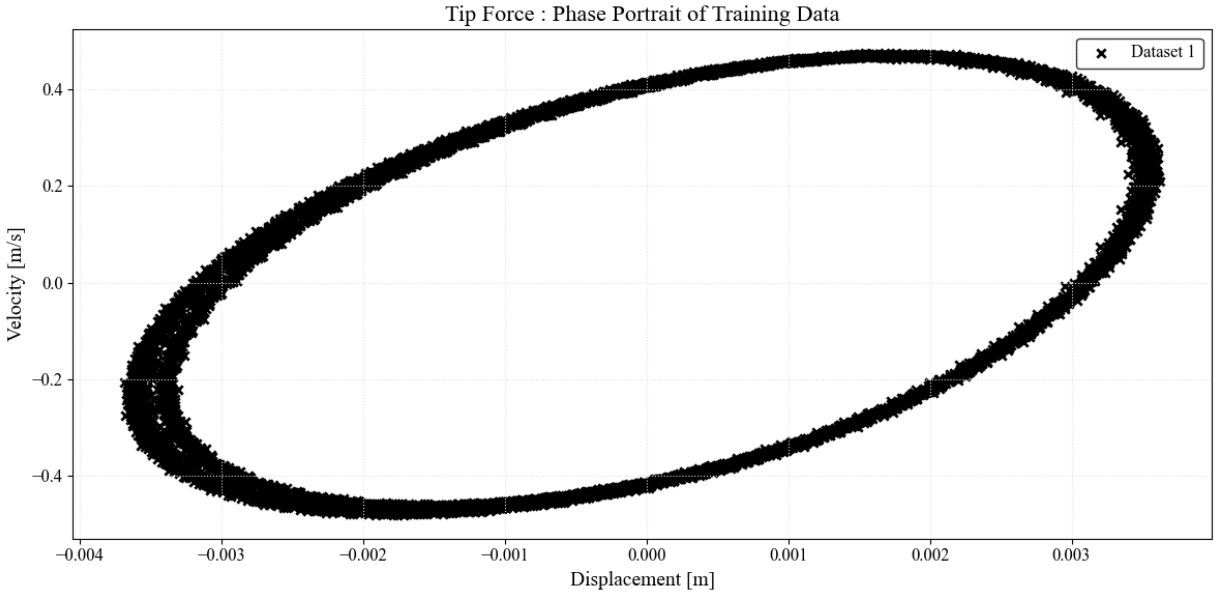


Figure N.28: Shaft Force from Test 3: Phase Portrait of Training Data

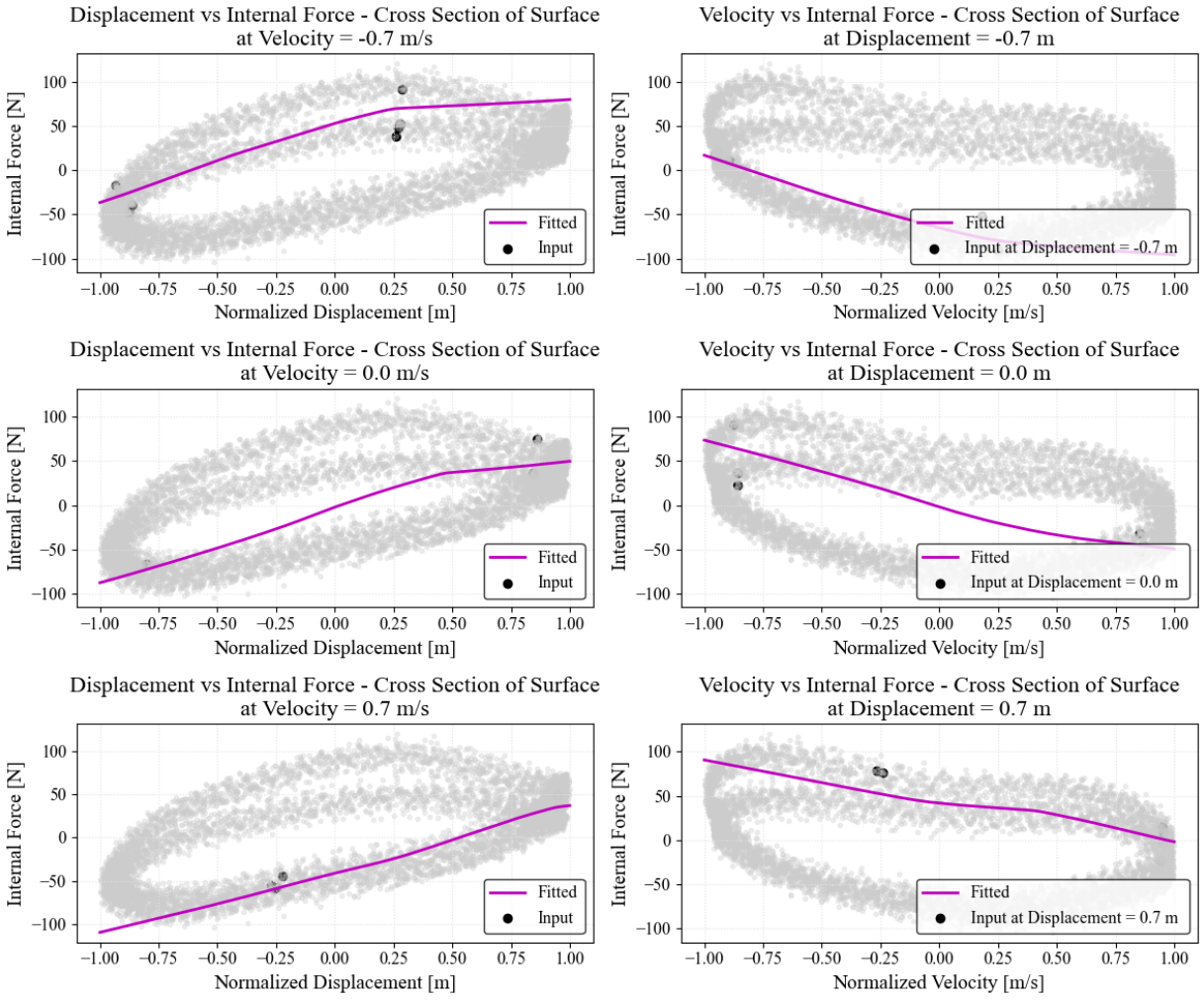


Figure N.29: Shaft Force from Test 3 : Cross Sections of Surface

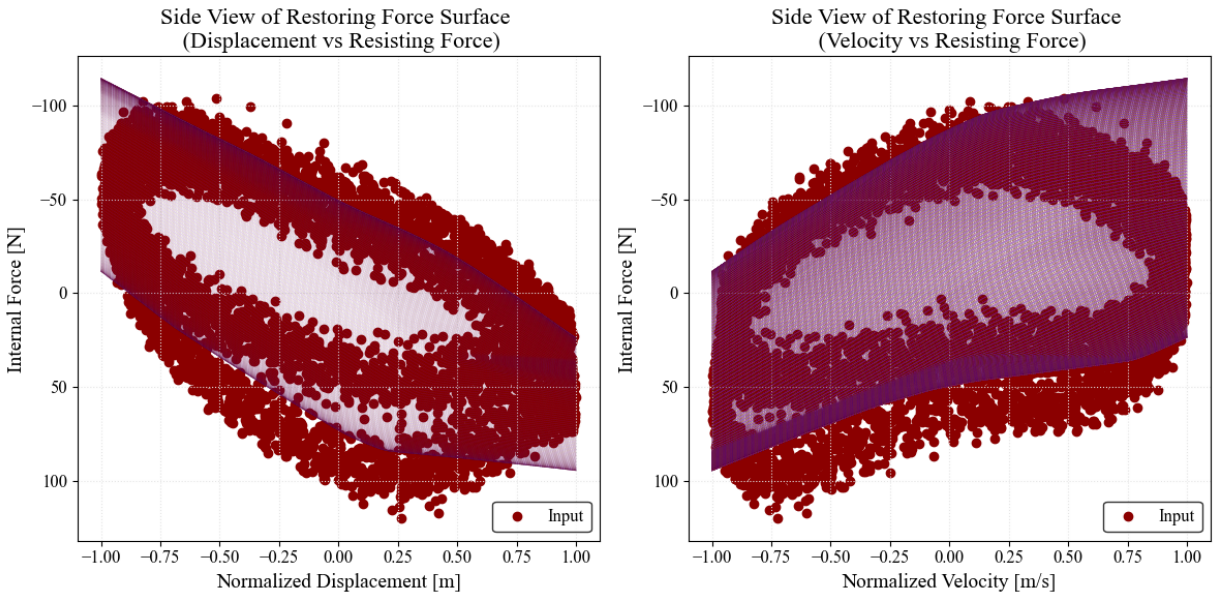


Figure N.30: Shaft Force from Test 3 : Side Views of Surface

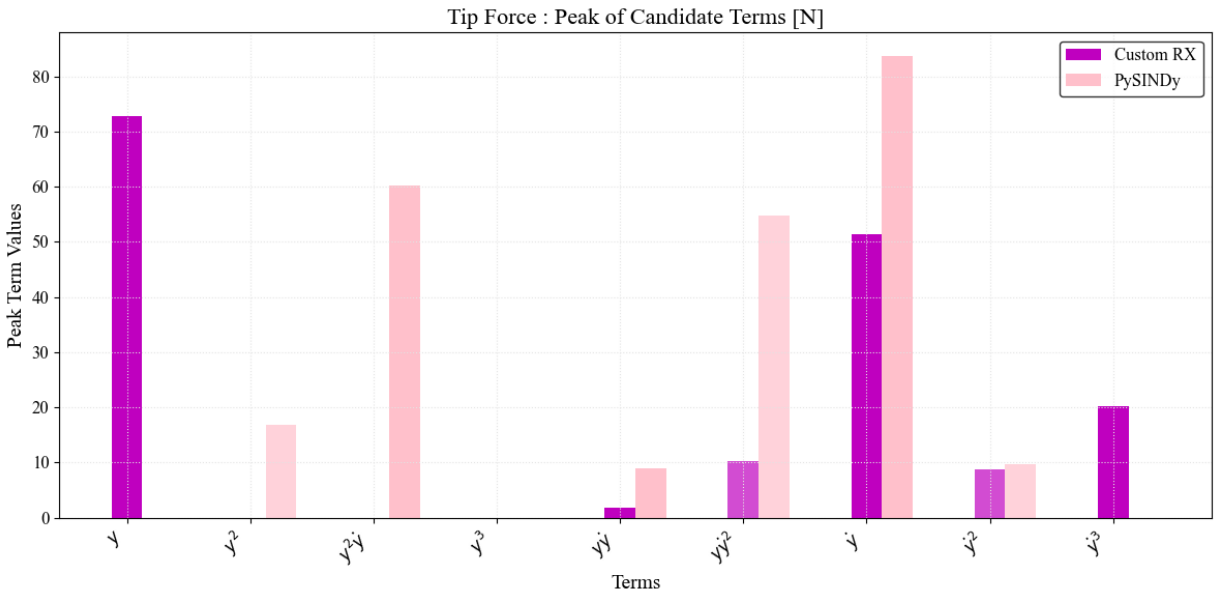


Figure N.31: Shaft Force from Test 3 : Force Features comparison

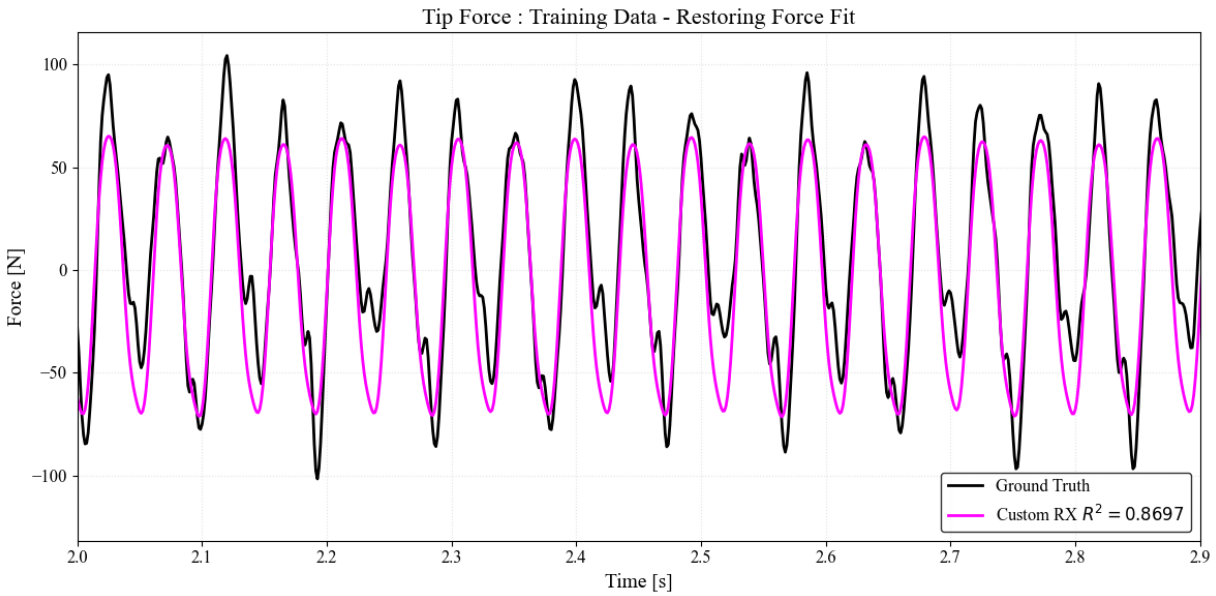


Figure N.32: Shaft Force from Test 3 : Training Data - Restoring Force Fit

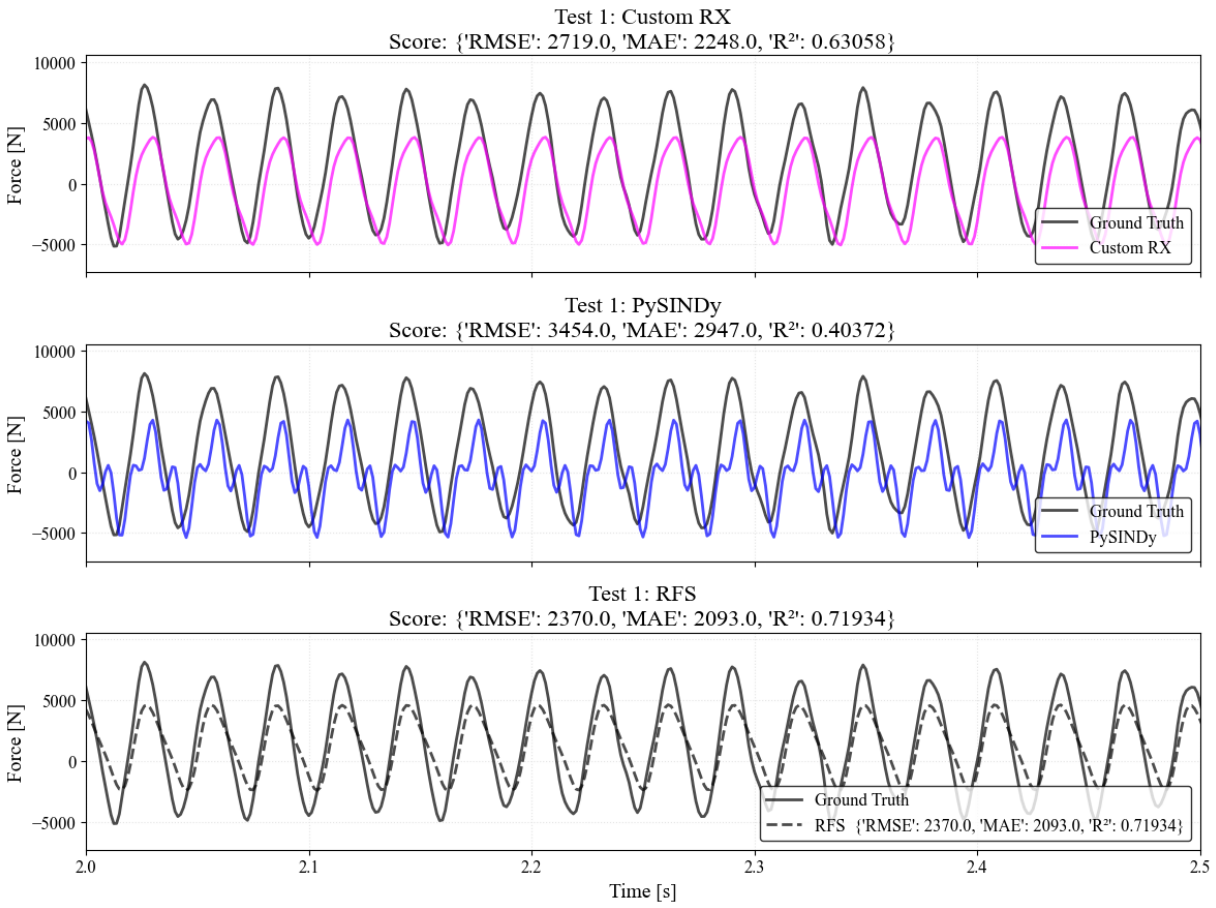


Figure N.33: Shaft Force from Test 3 : Test 1 Data - Fit for t =2-10s

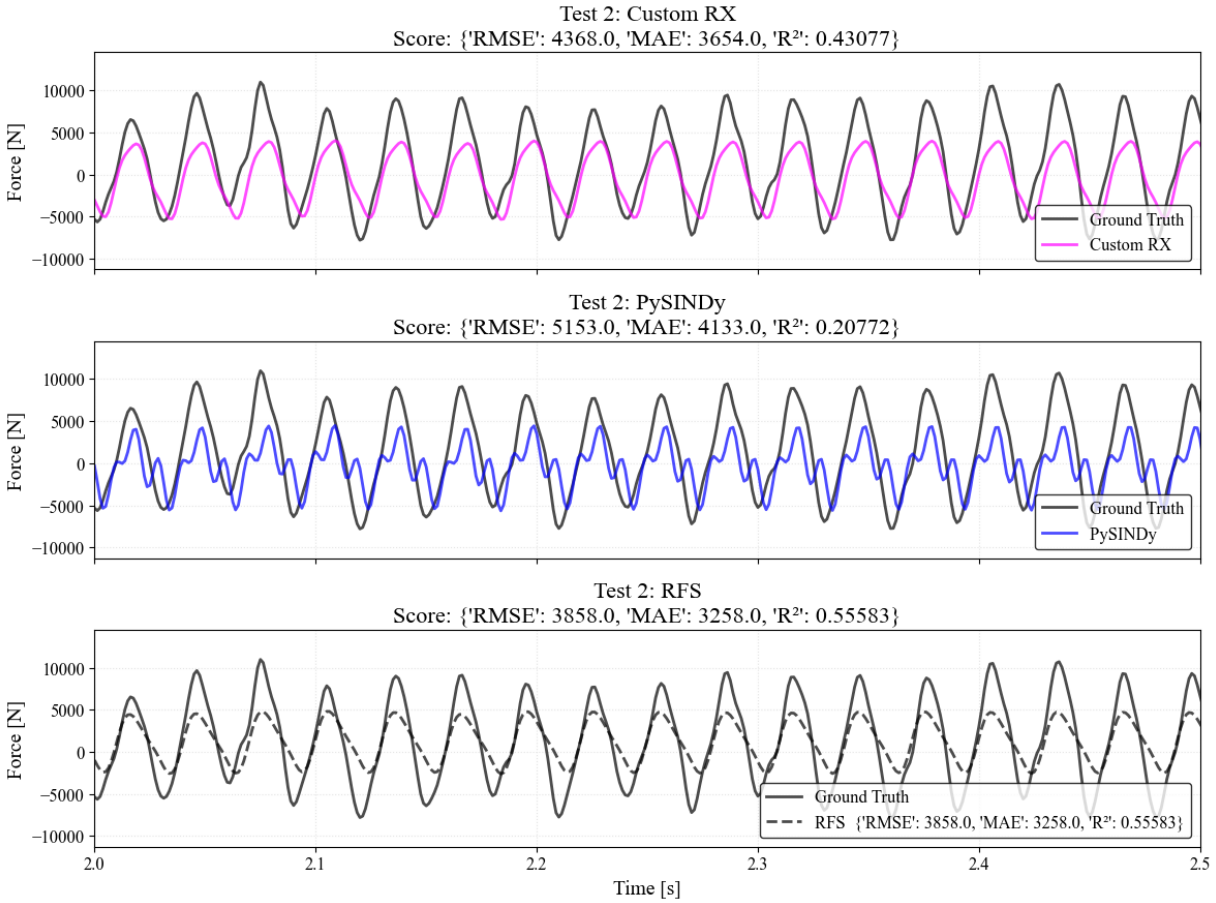


Figure N.34: Shaft Force from Test 3 : Test 2 Data - Fit for t =2-10s

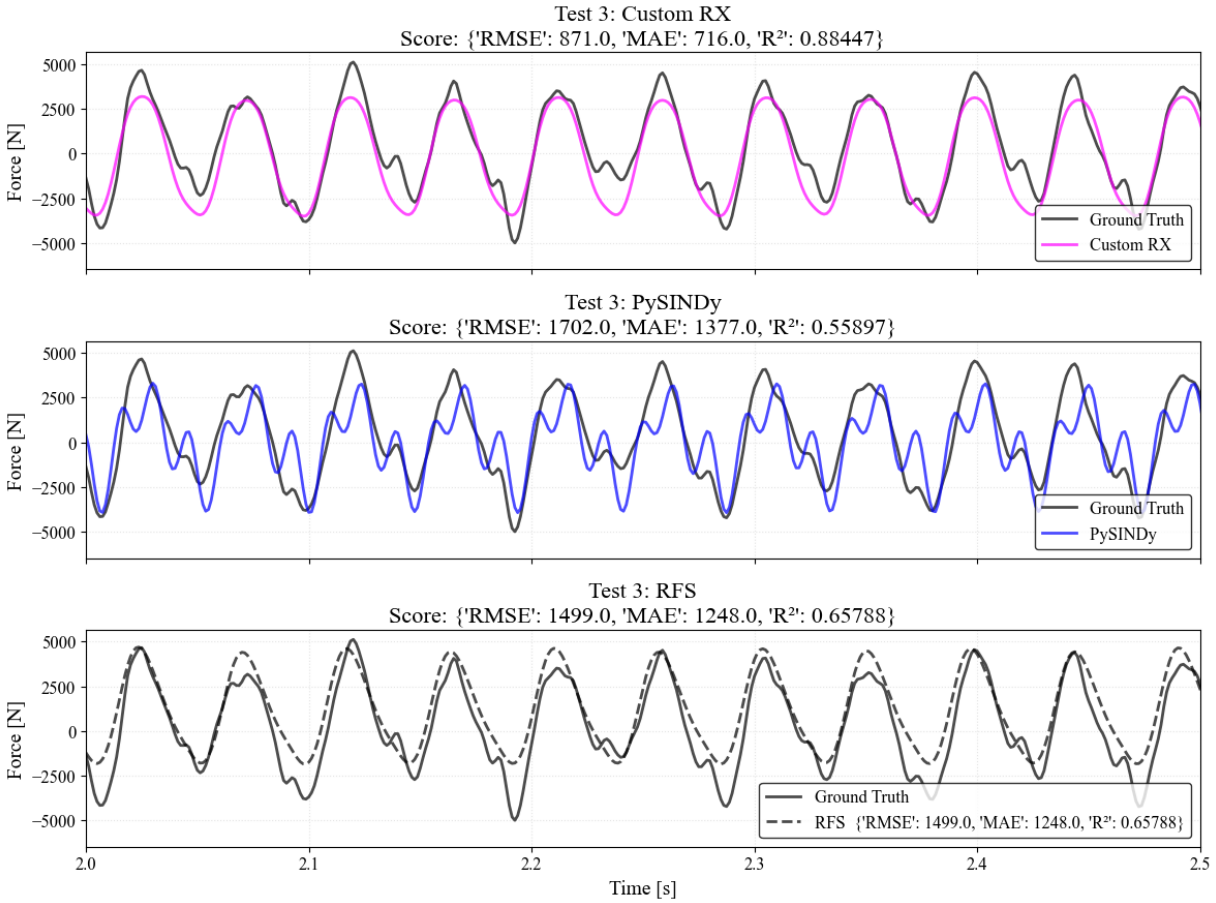


Figure N.35: Shaft Force from Test 3 : Test 3 Data - Fit for t =2-10s

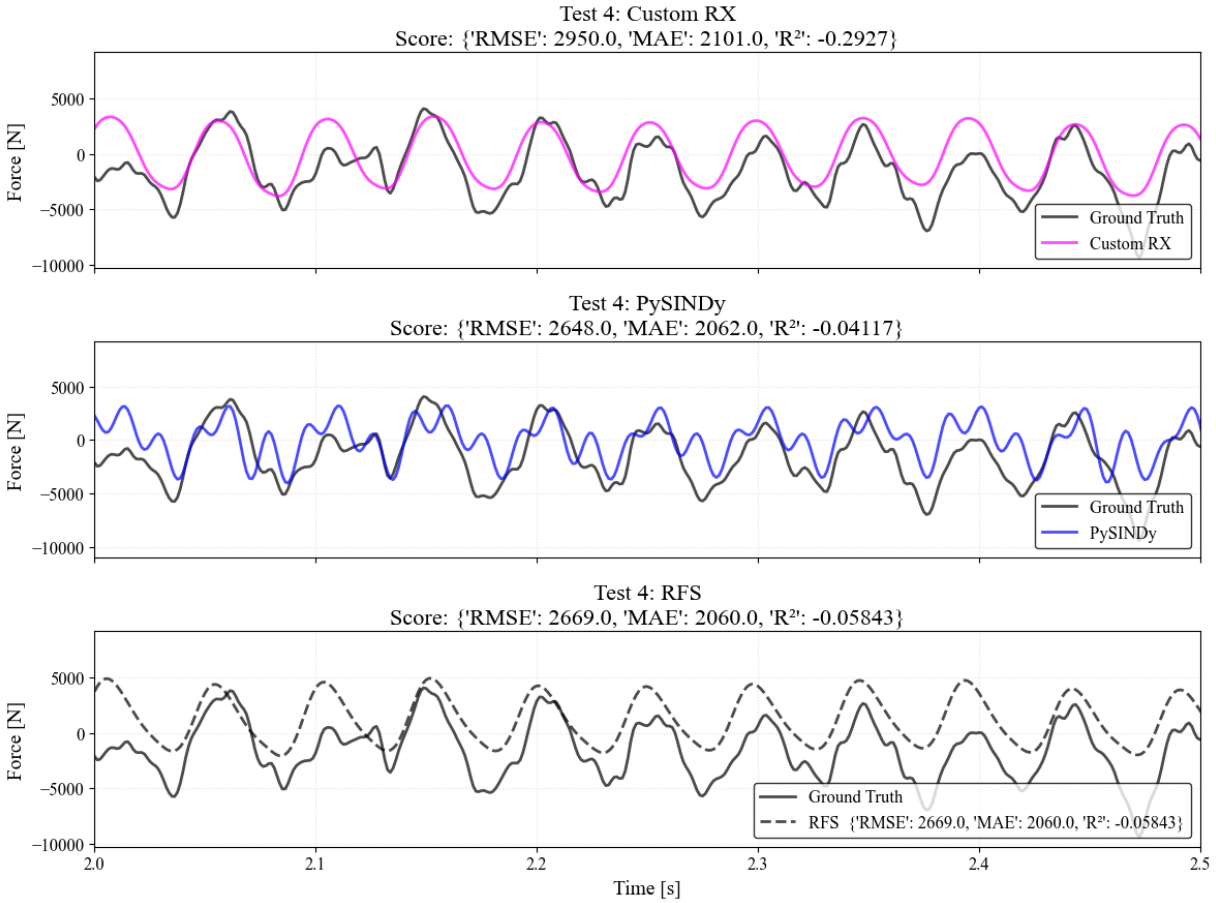


Figure N.36: Shaft Force from Test 3 : Test 4 Data - Fit for t =2-10s

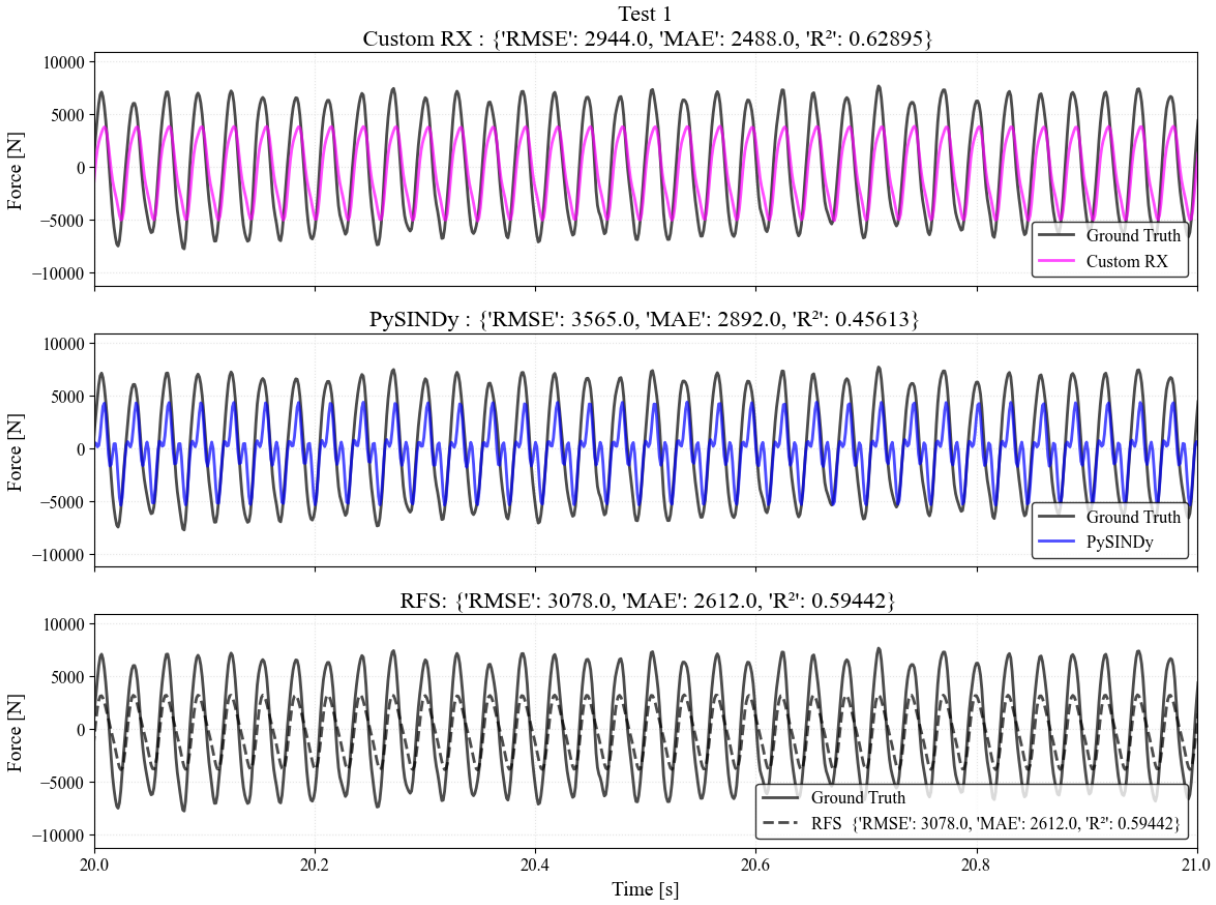


Figure N.37: Shaft Force from Test 3 : Test 1 Data - Fit for t =2-40s

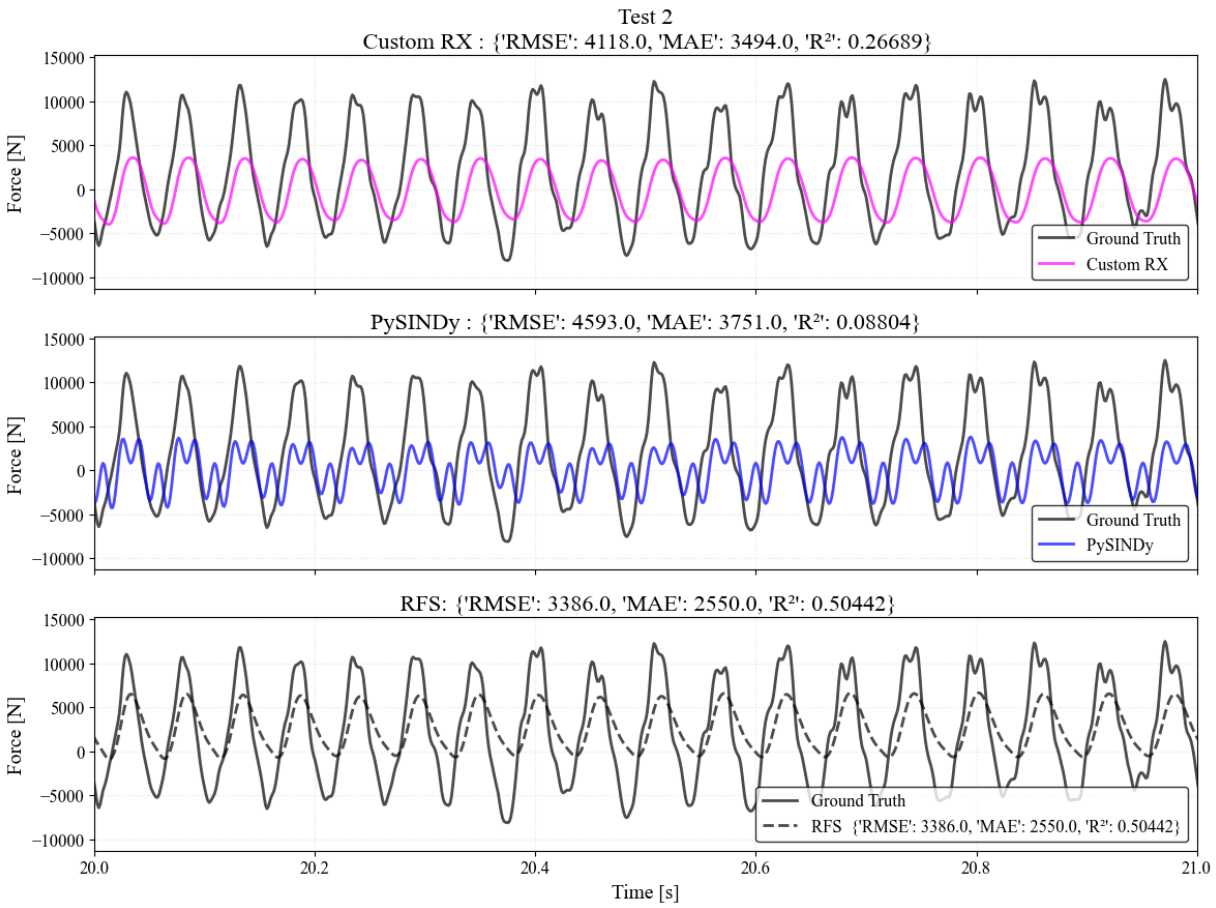


Figure N.38: Shaft Force from Test 3 : Test 2 Data - Fit for t =2-40s

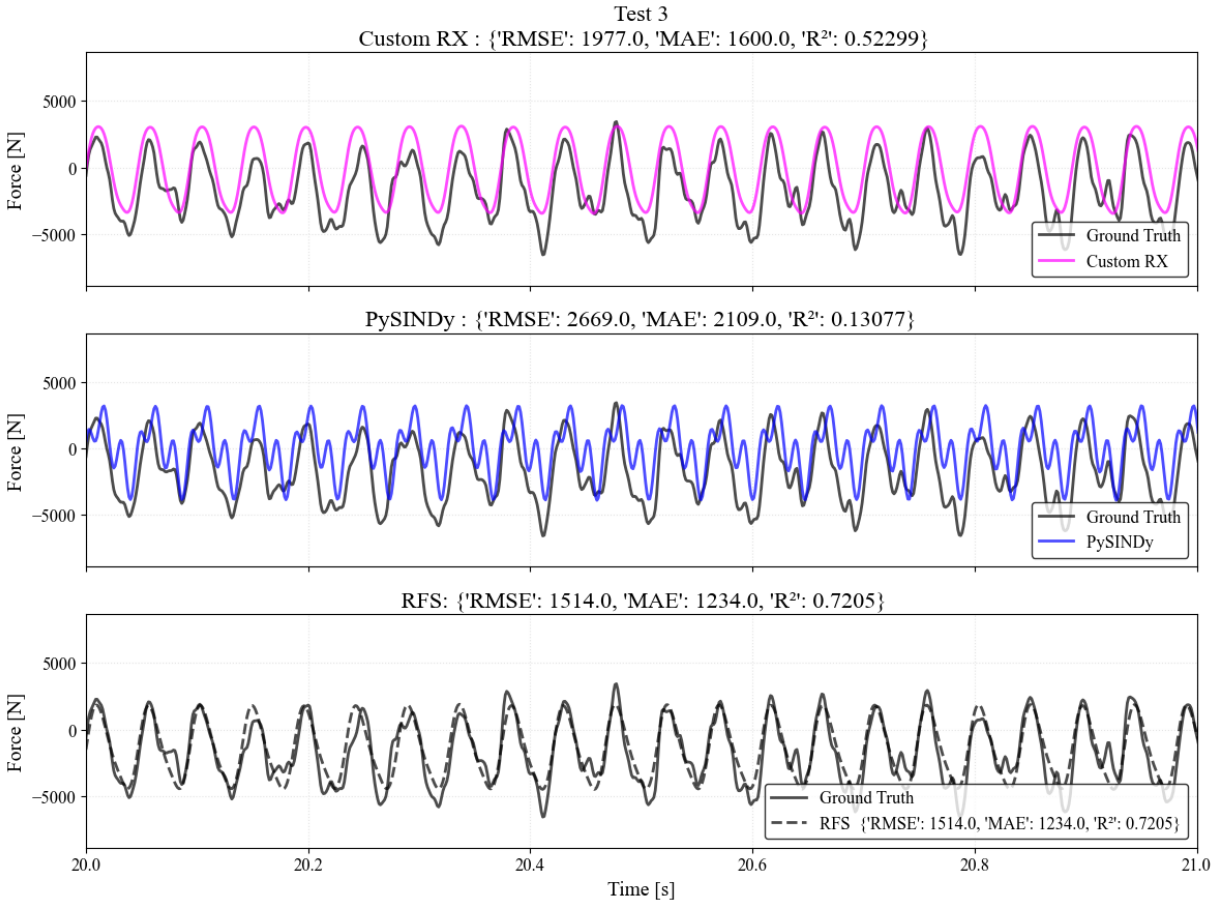


Figure N.39: Shaft Force from Test 3 : Test 3 Data - Fit for t =2-40s

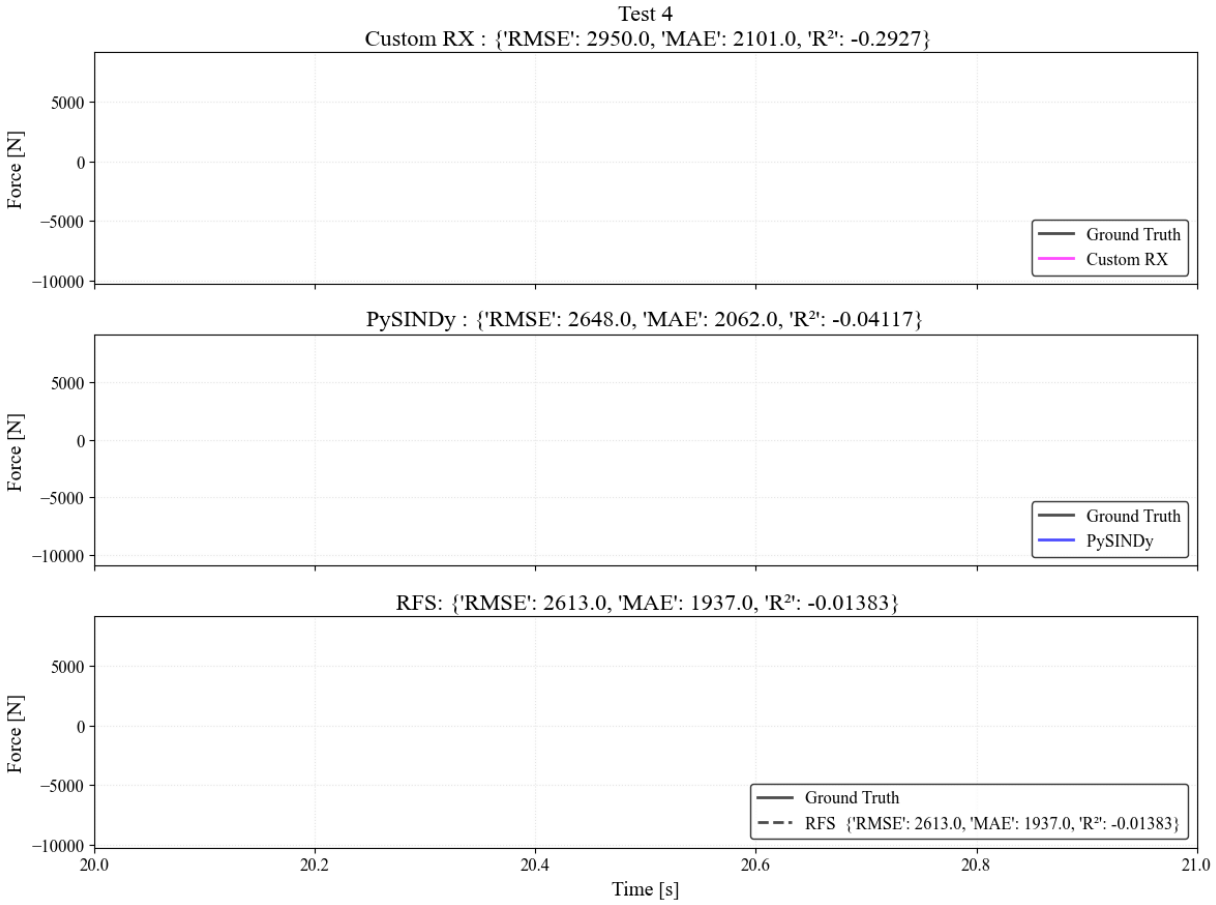


Figure N.40: Shaft Force from Test 3 : Test 4 Data - Fit for t =2-40s

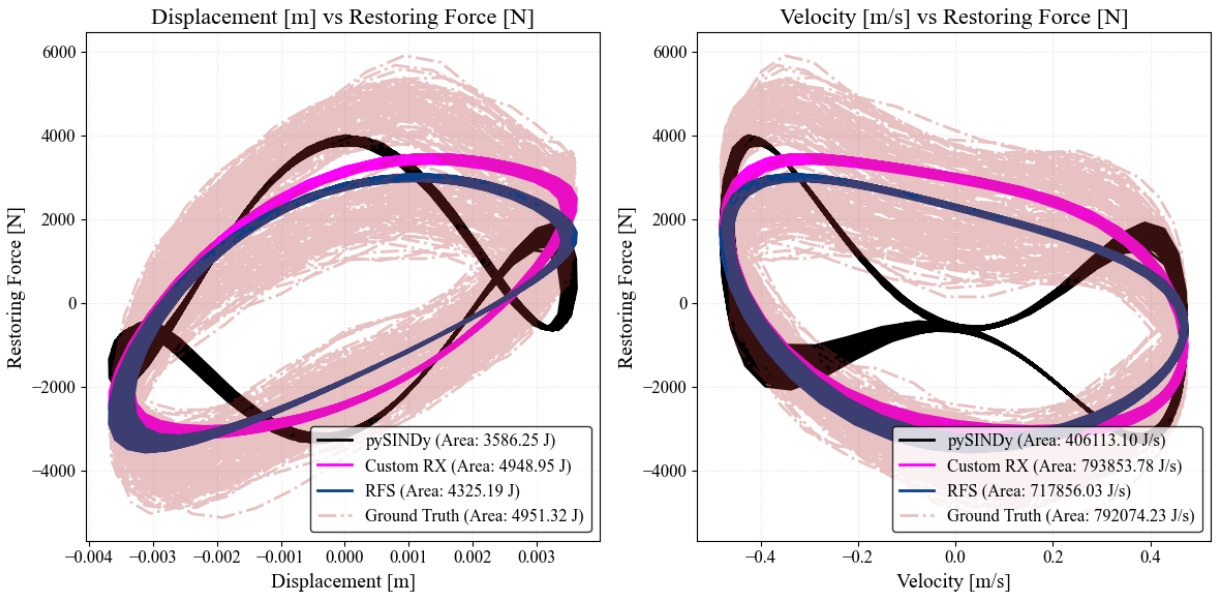


Figure N.41: Shaft Force from Test 3: Fitted loading Cycles

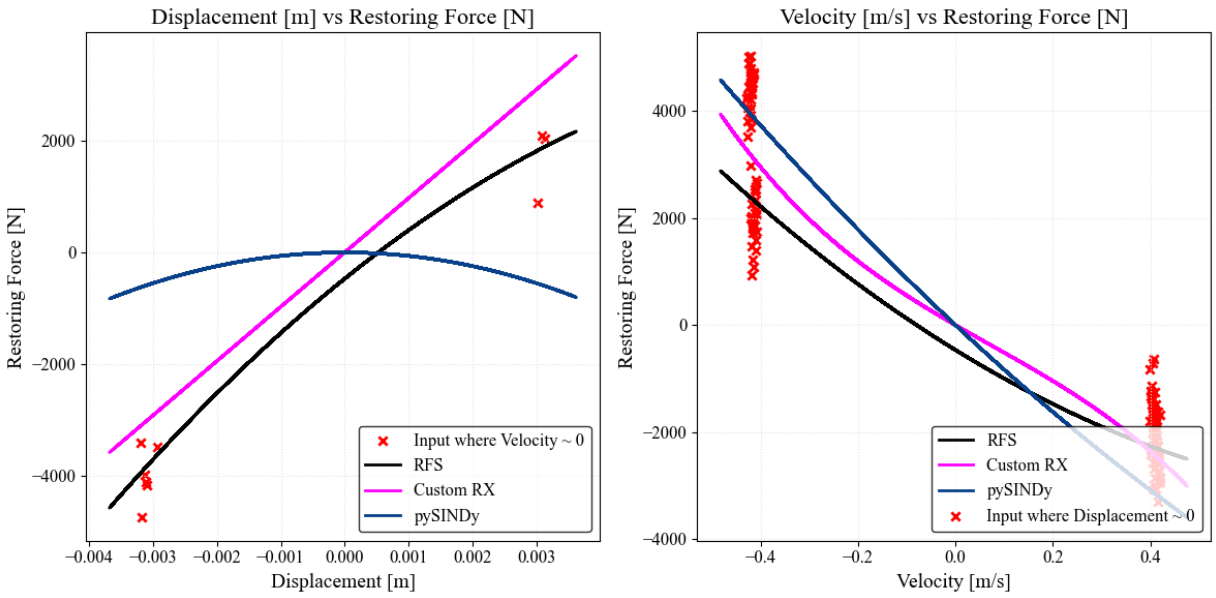


Figure N.42: Shaft Force from Test 3: Fitted loading Cycles - Section

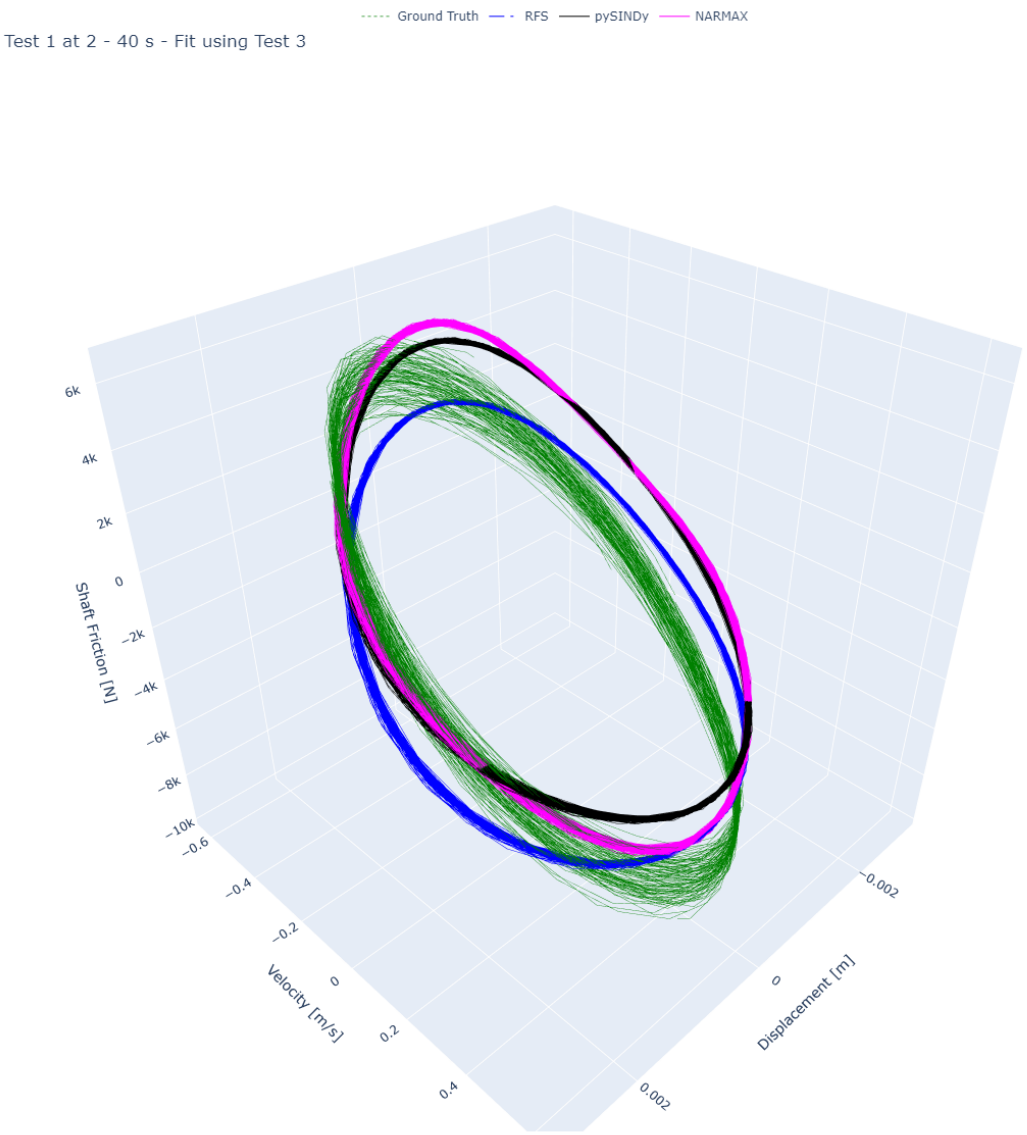


Figure N.43: 3D View-1 of shaft force vs displacement and velocity for shaft force from Test 3

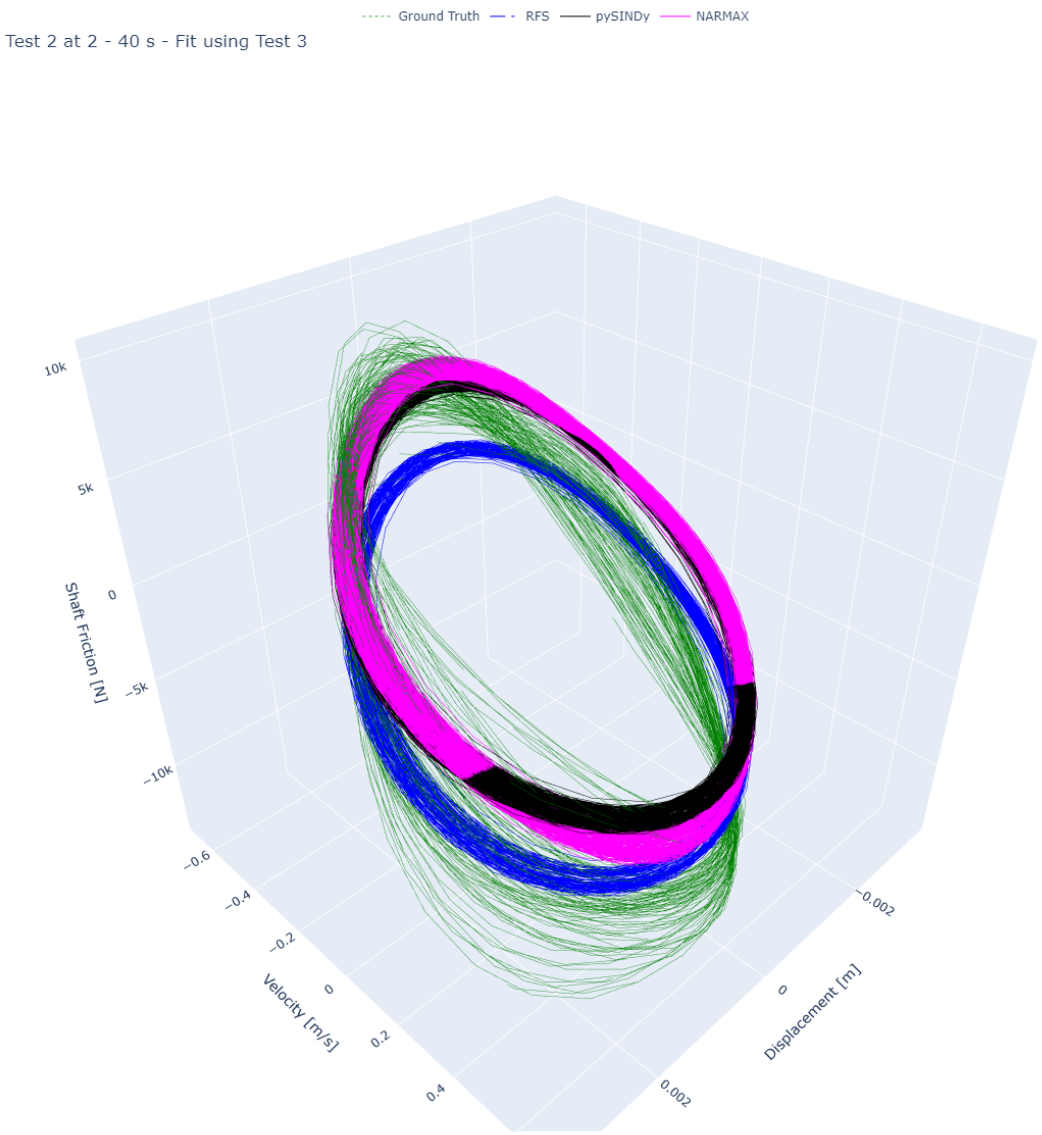


Figure N.44: 3D View-2 of shaft force vs displacement and velocity for shaft force from Test 3

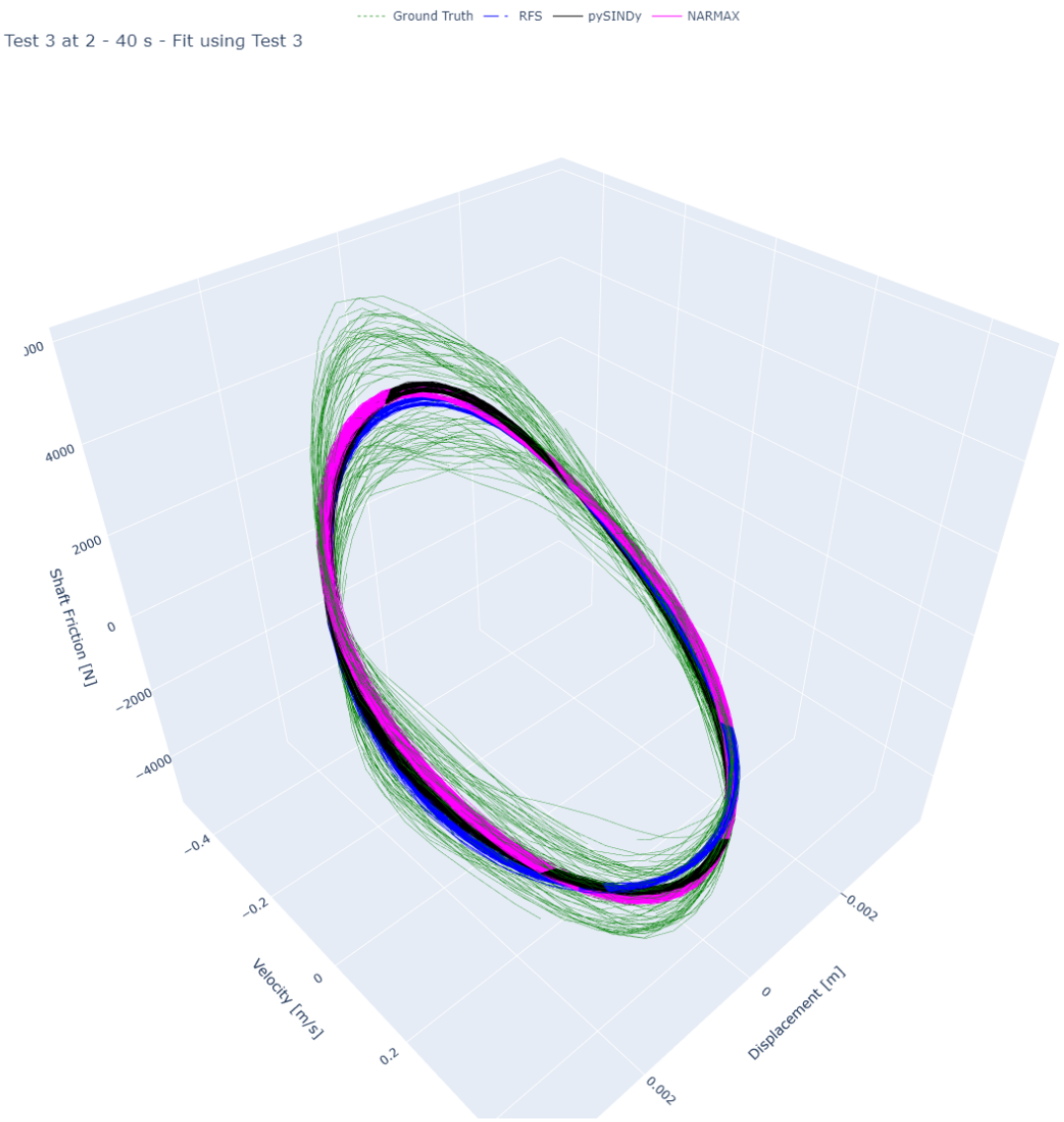


Figure N.45: 3D View-3 of shaft force vs displacement and velocity for shaft force from Test 3

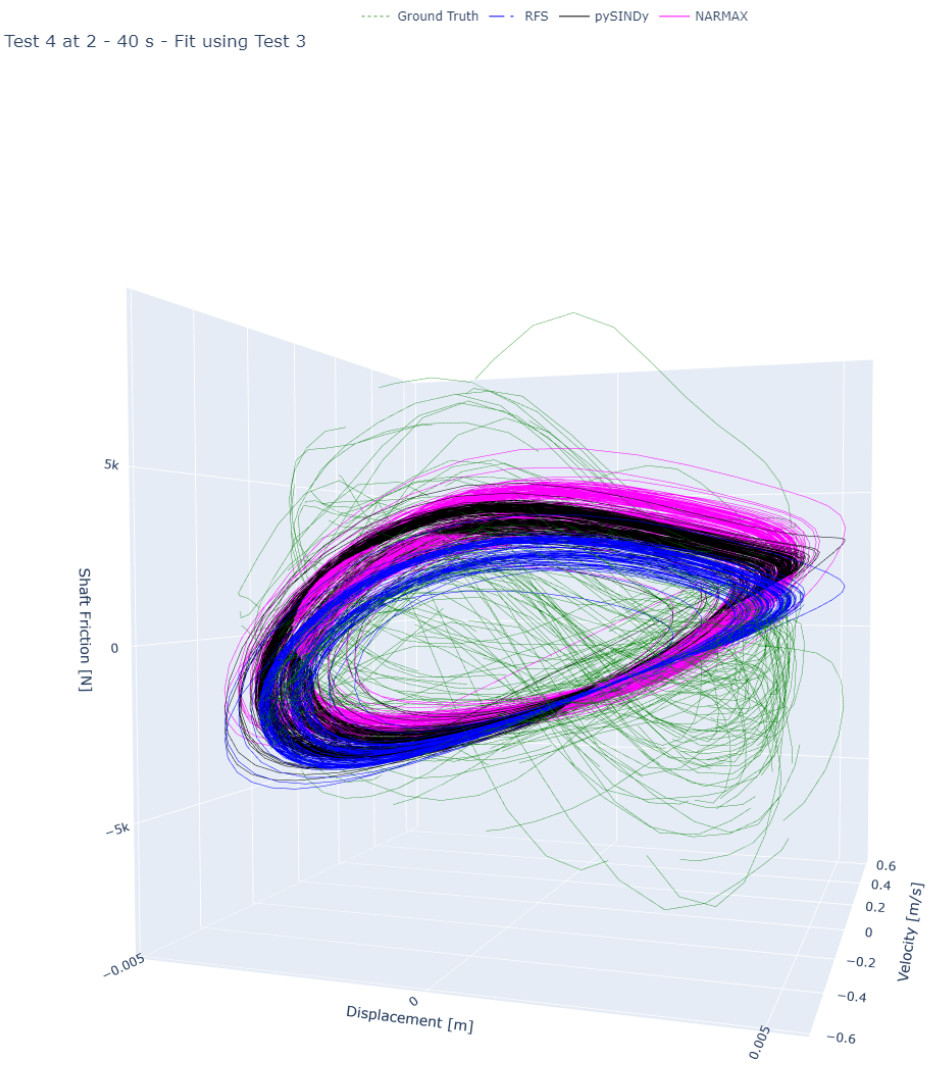


Figure N.46: 3D View-4 of shaft force vs displacement and velocity for shaft force from Test 3

Results for Tip Force from Test 1

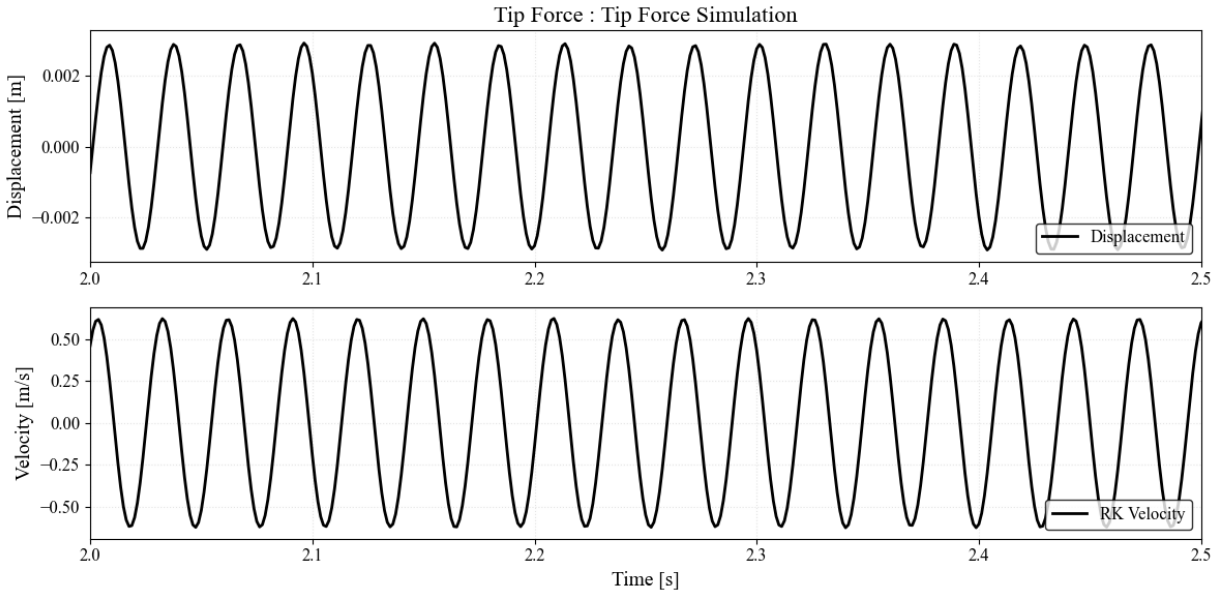


Figure N.47: Tip Force from Test 1 - Filtered Measurements

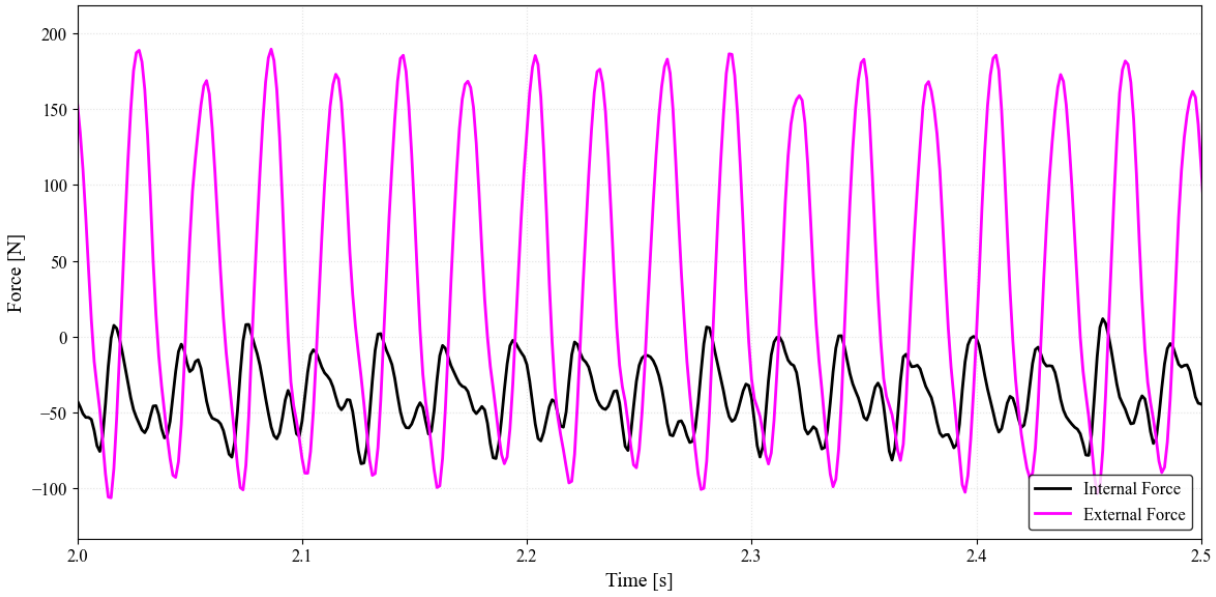


Figure N.48: Tip Force from Test 1 External forcing & Measured Tip Force

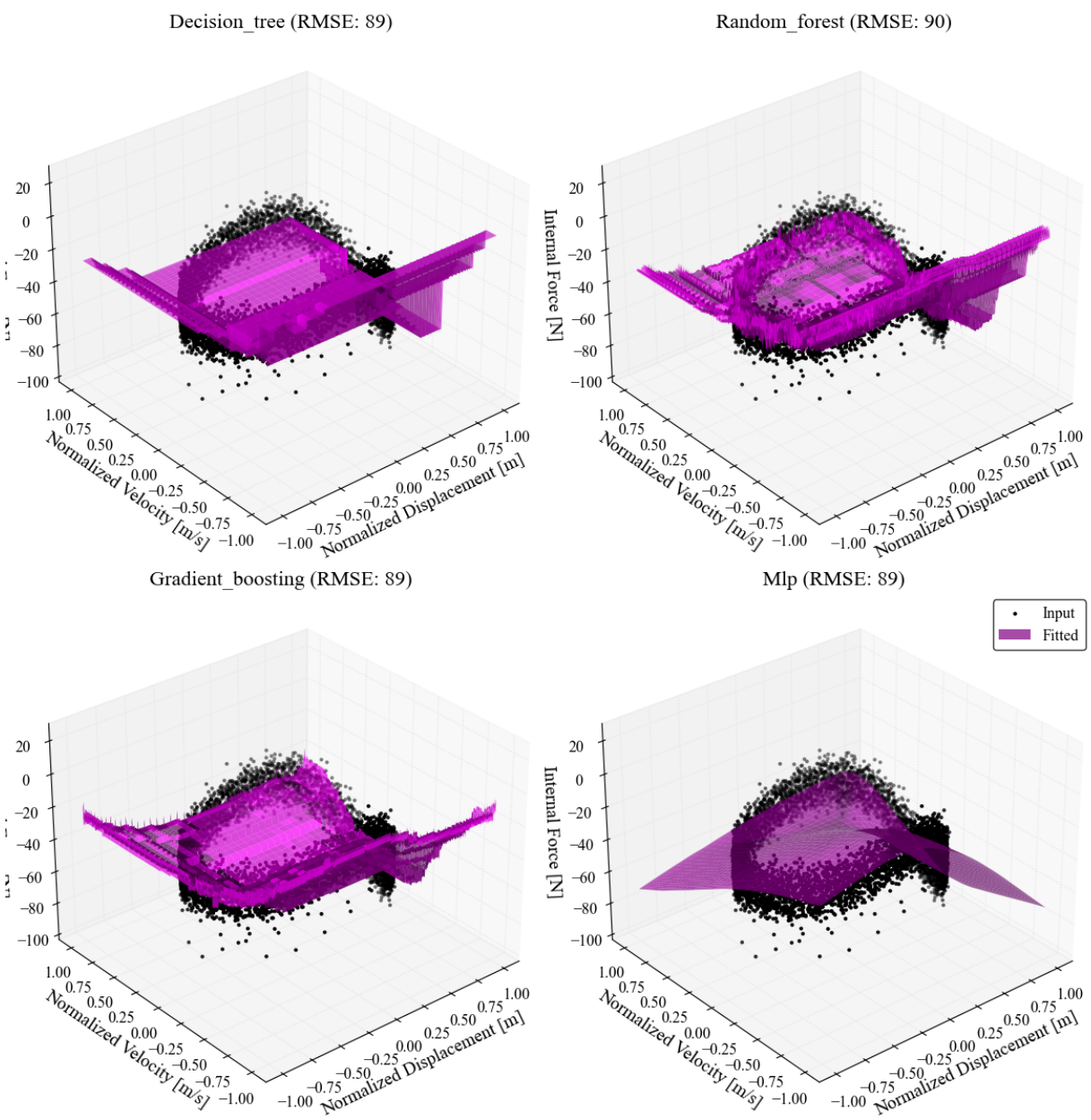


Figure N.49: Tip Force from Test 1 : Machine Learning fitted 3D Surfaces

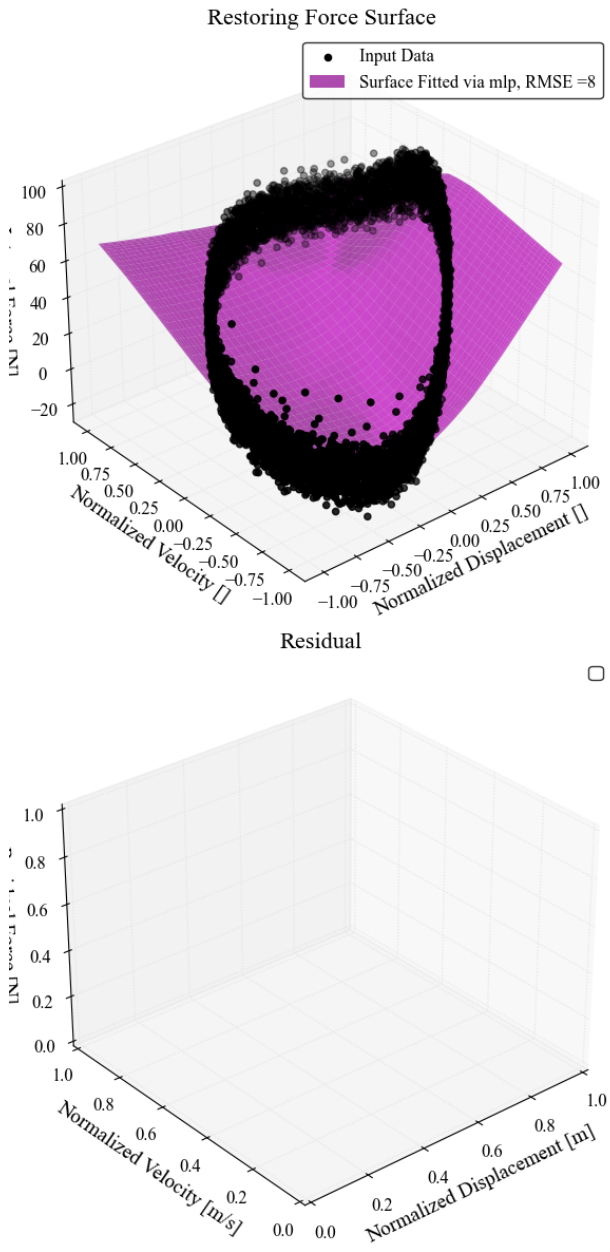


Figure N.50: Tip Force from Test 1: Selected Restoring Force Surface

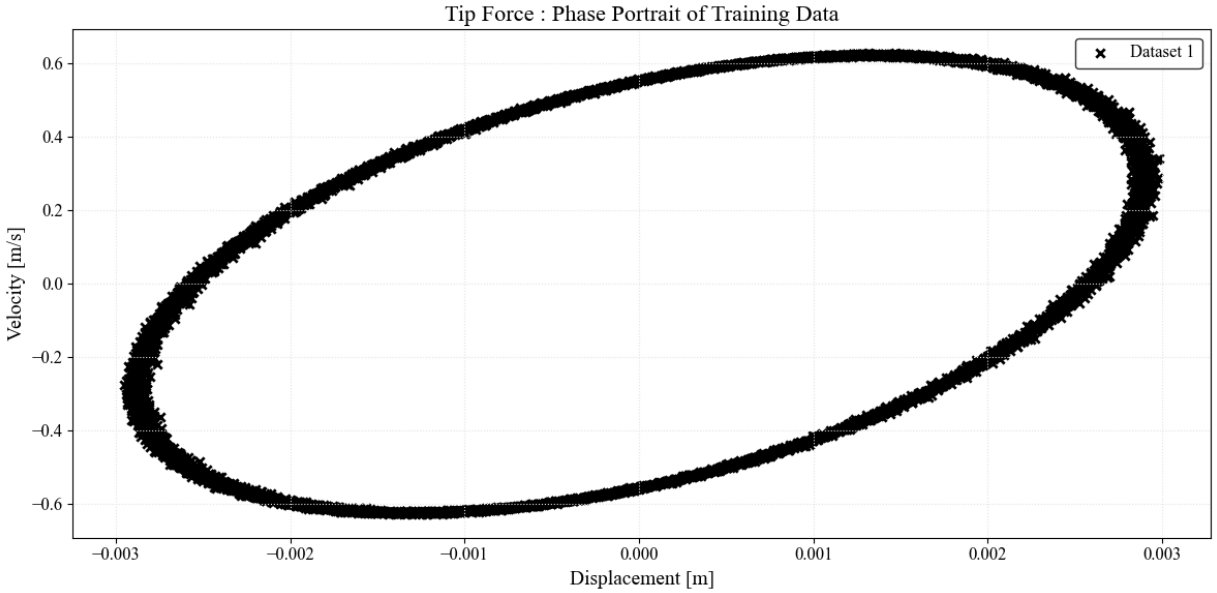


Figure N.51: Tip Force from Test 1: Phase Portrait of Training Data

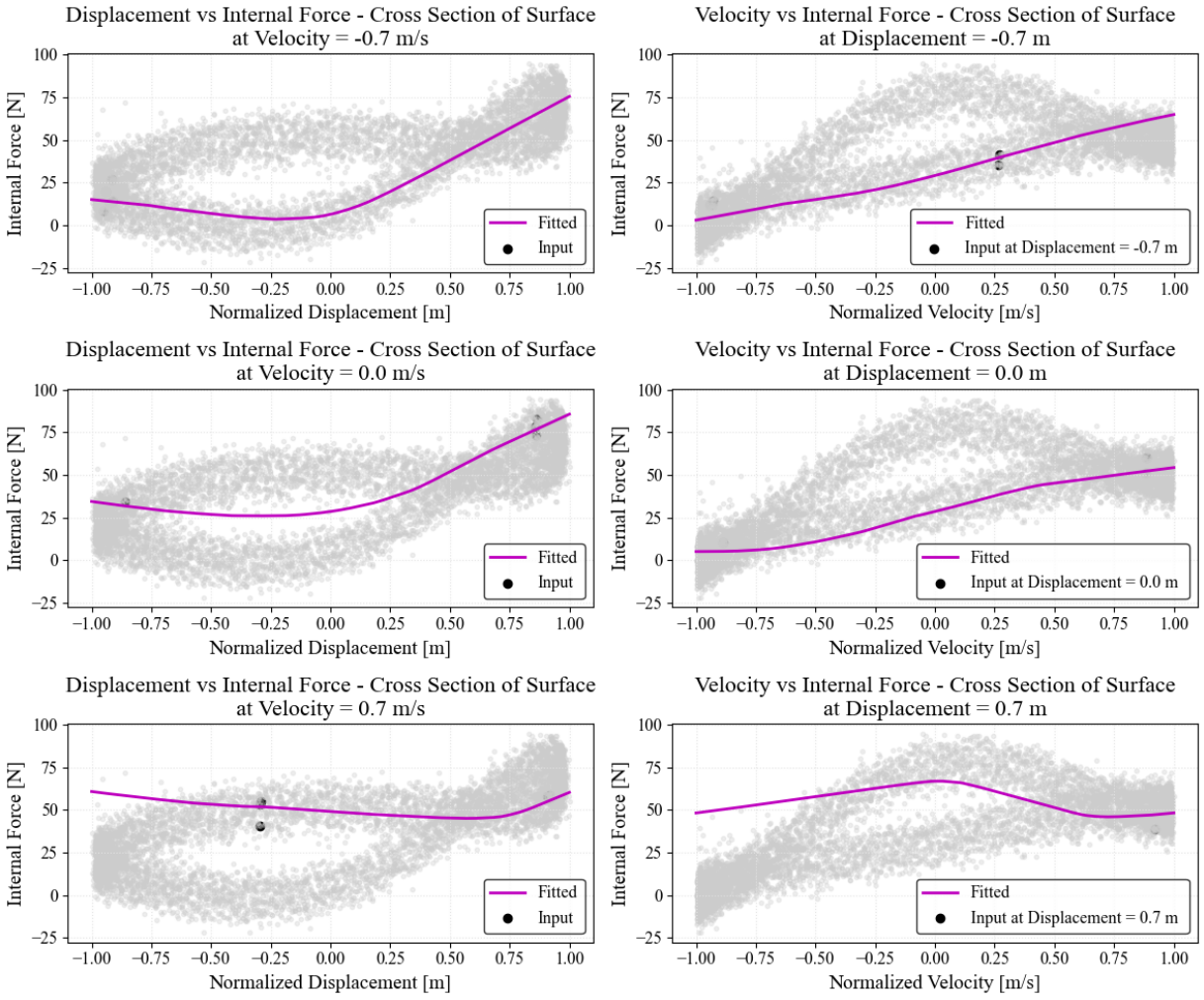


Figure N.52: Tip Force from Test 1 : Cross Sections of Surface

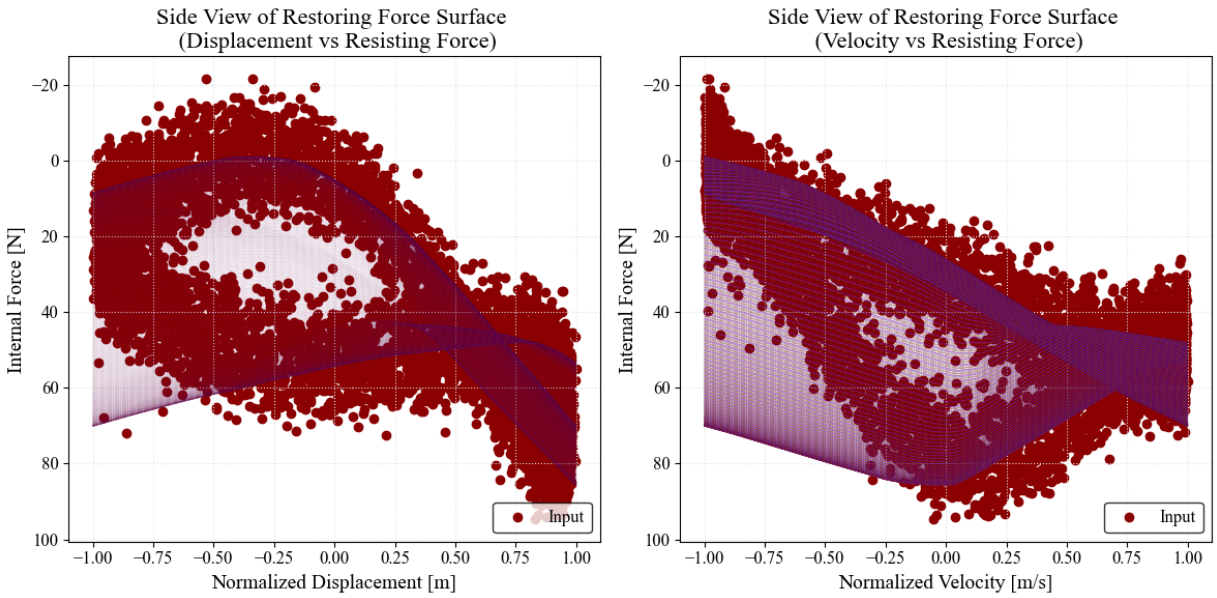


Figure N.53: Tip Force from Test 1 : Side Views of Surface

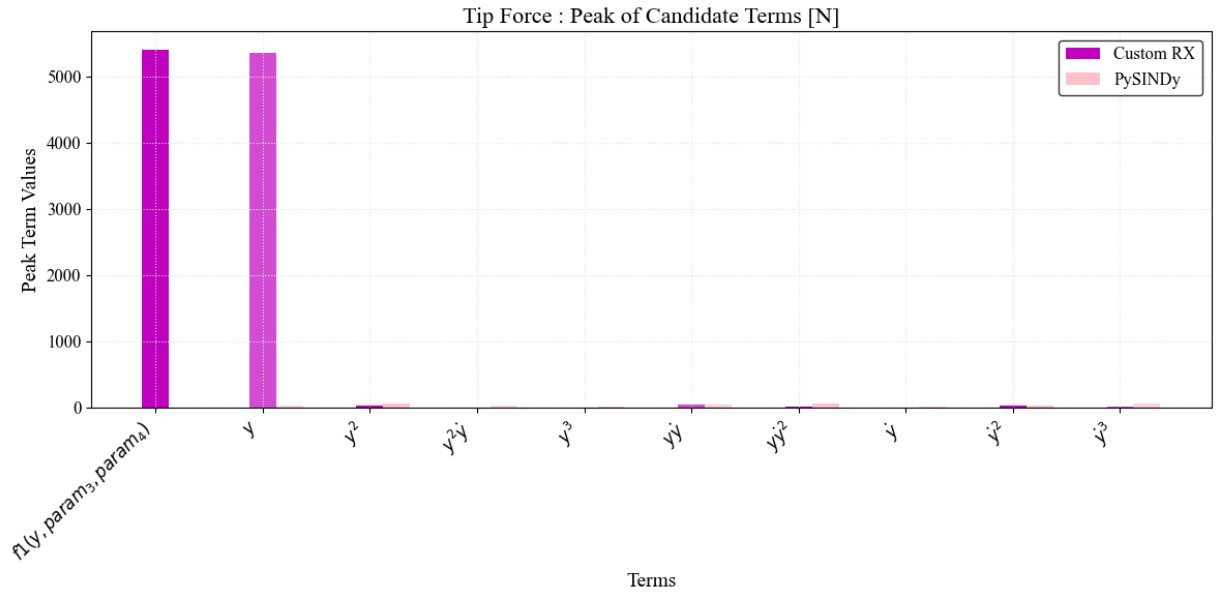


Figure N.54: Tip Force from Test 1 : Force Features comparison

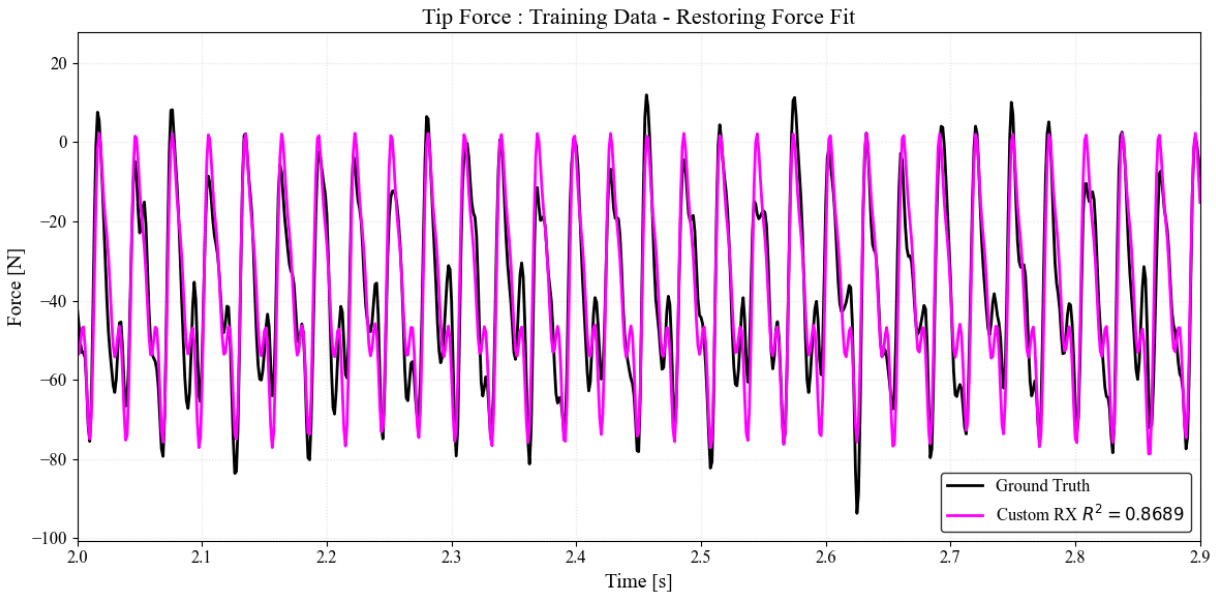


Figure N.55: Tip Force from Test 1 : Training Data - Restoring Force Fit

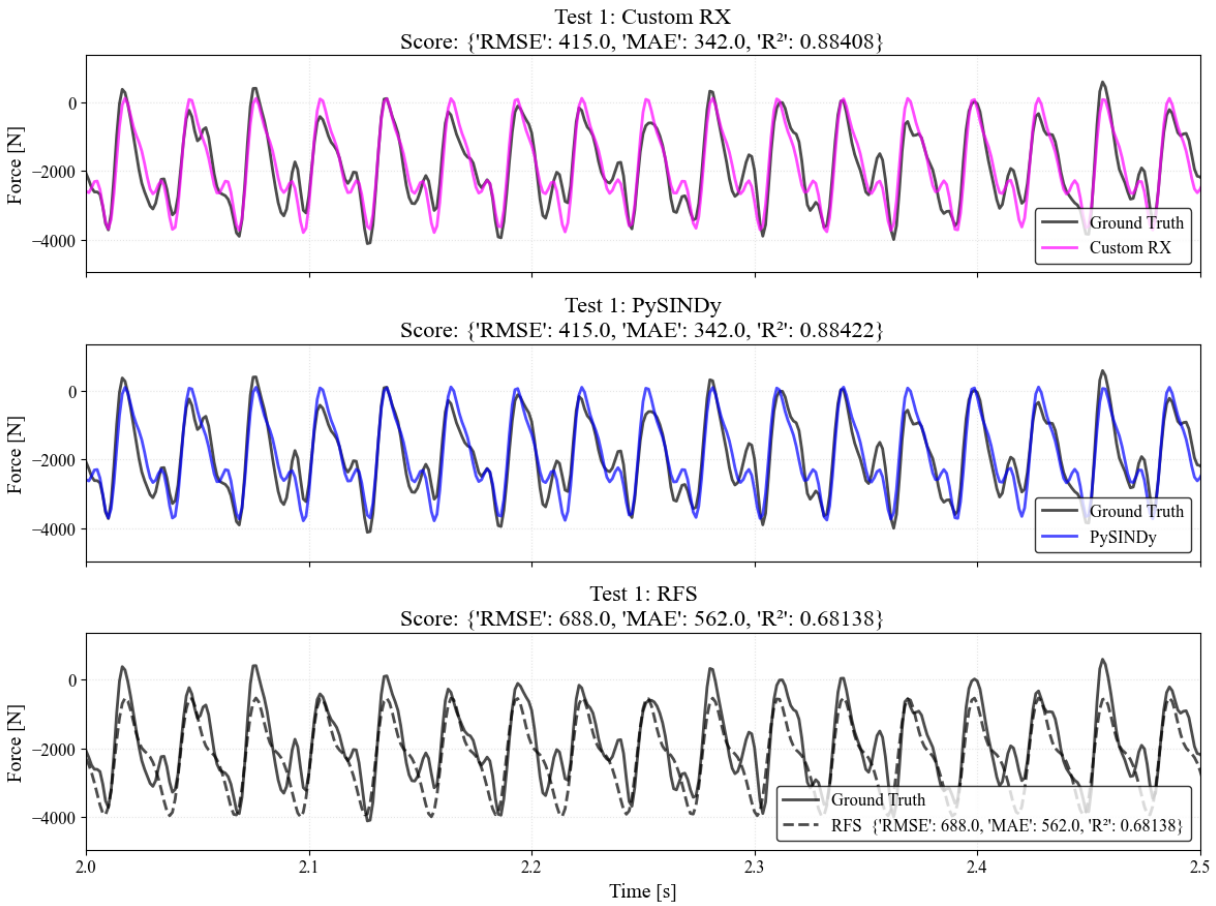


Figure N.56: Tip Force from Test 1 : Test 1 Data - Fit for t =2-10s

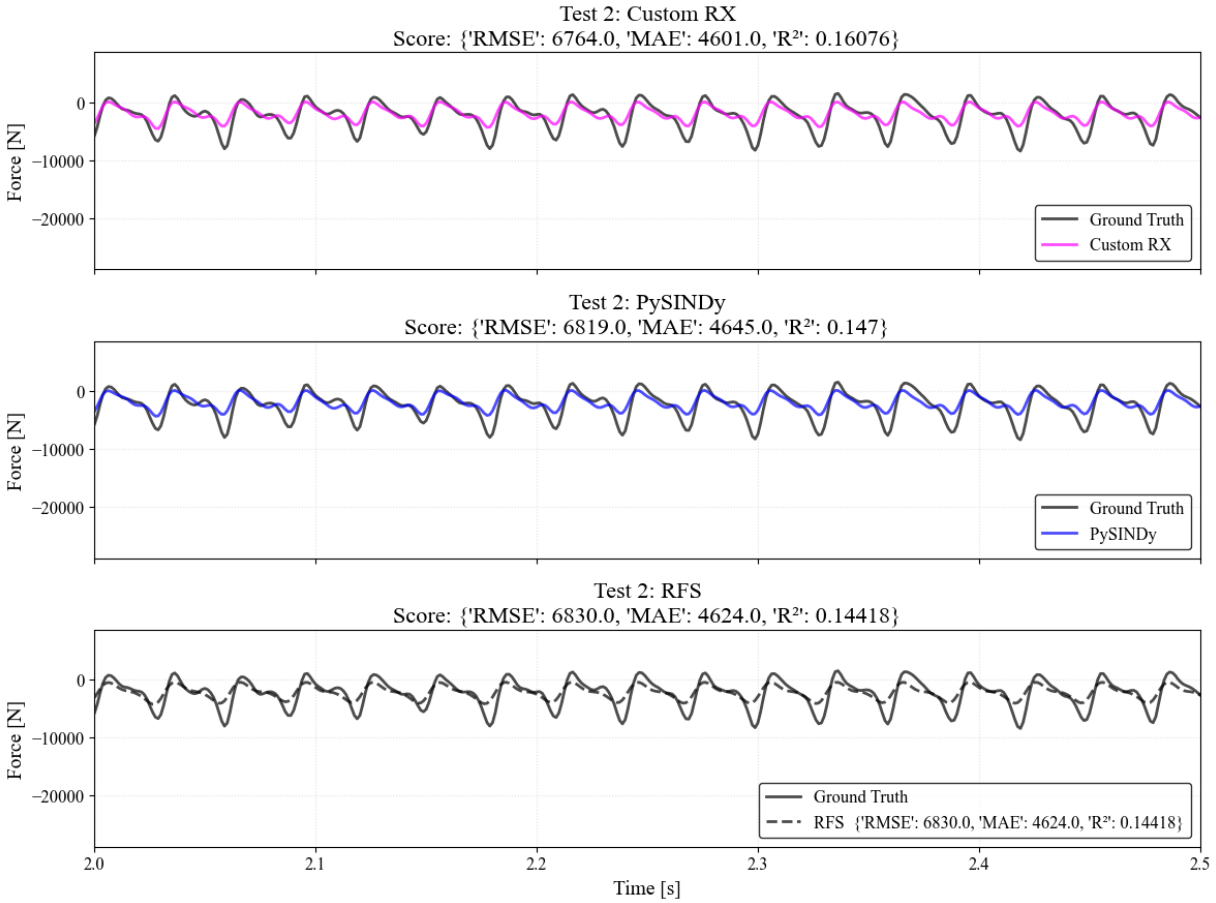


Figure N.57: Tip Force from Test 1 : Test 2 Data - Fit for t =2-10s

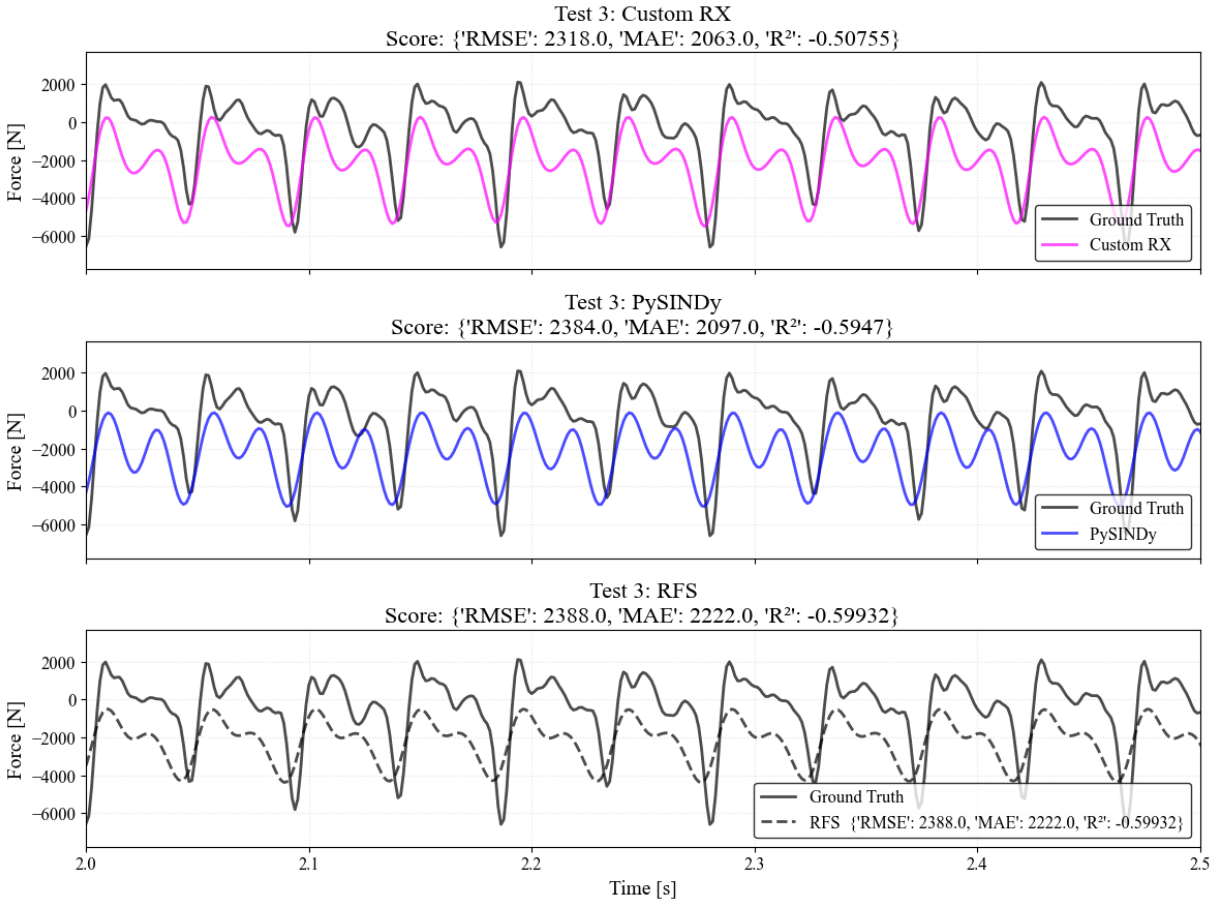


Figure N.58: Tip Force from Test 1 : Test 3 Data - Fit for t =2-10s

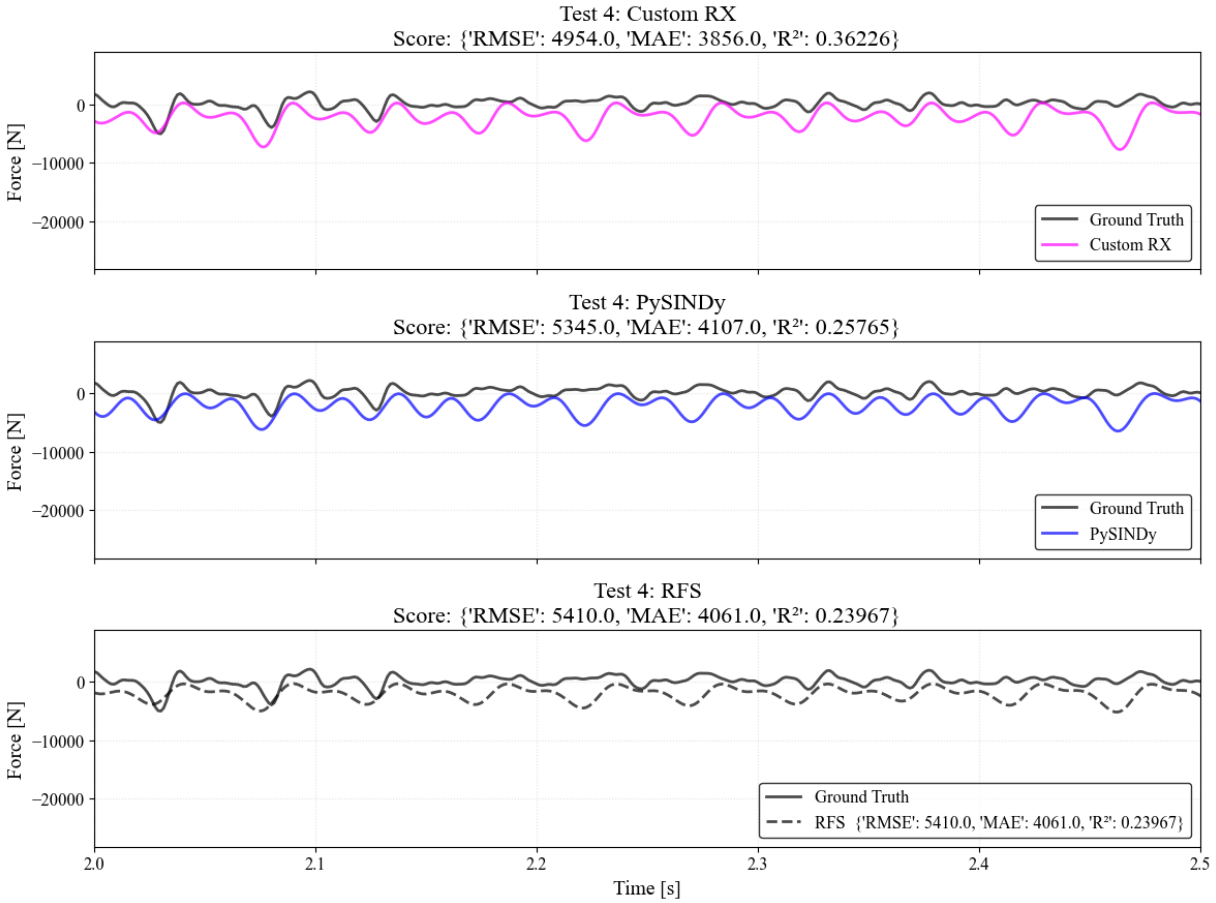


Figure N.59: Tip Force from Test 1 : Test 4 Data - Fit for t =2-10s

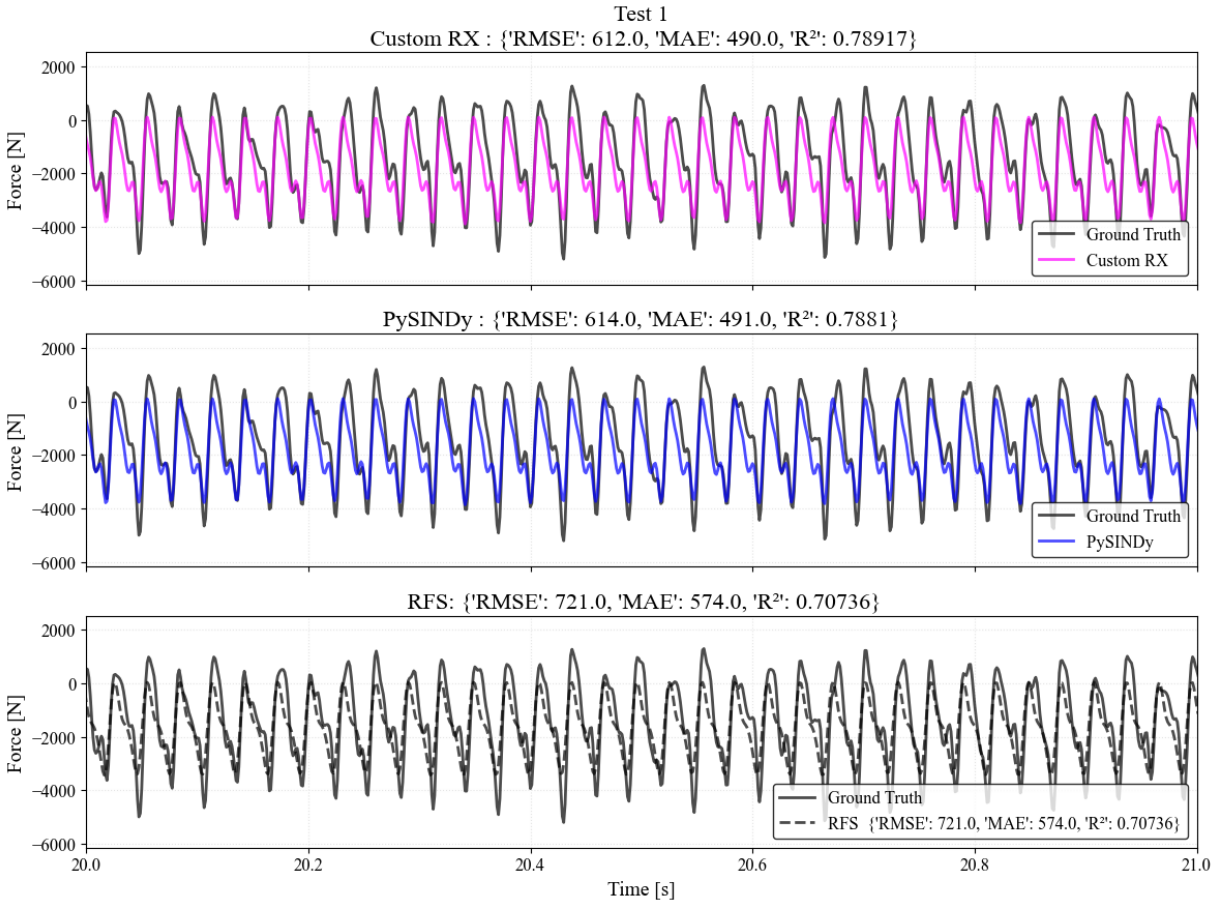


Figure N.60: Tip Force from Test 1 : Test 1 Data - Fit for t =2-40s

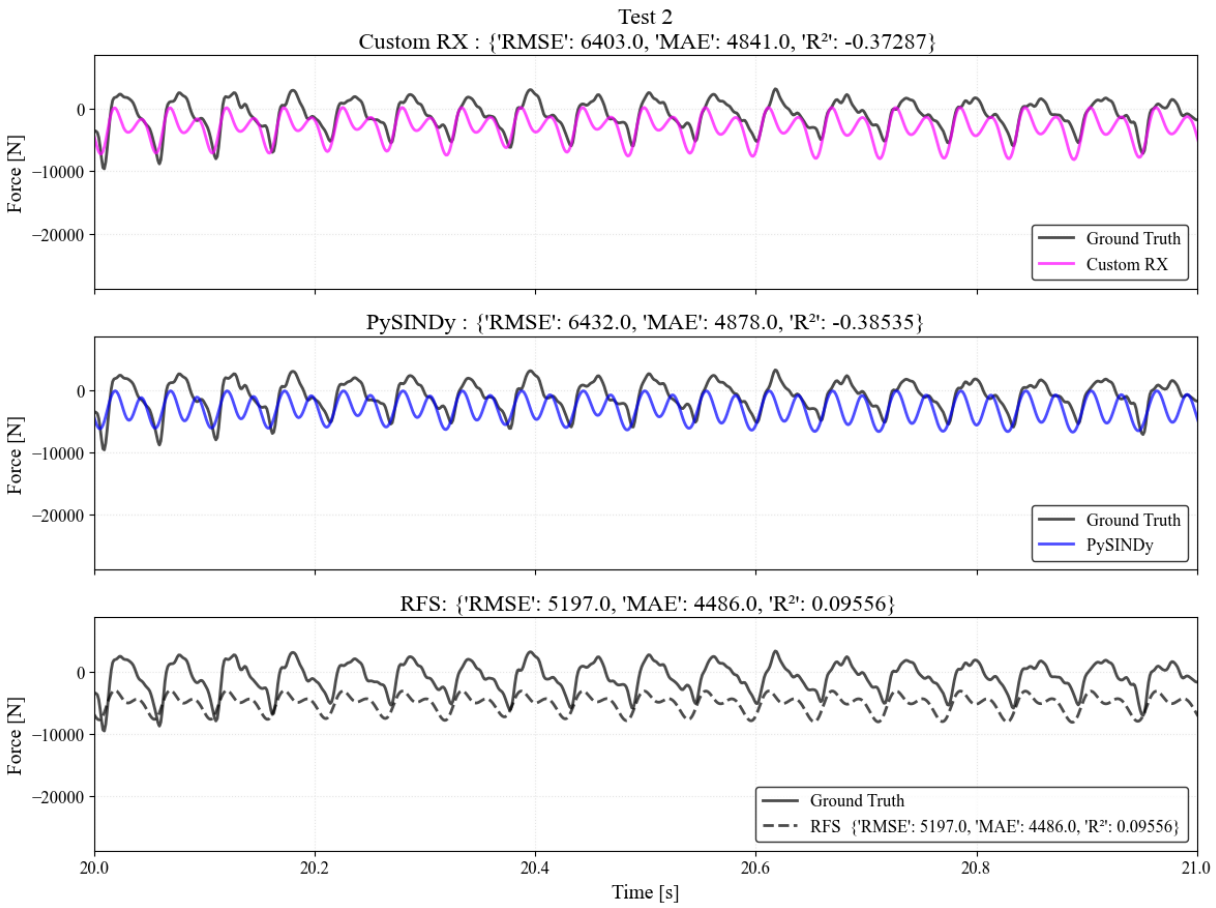


Figure N.61: Tip Force from Test 1 : Test 2 Data - Fit for t =2-40s

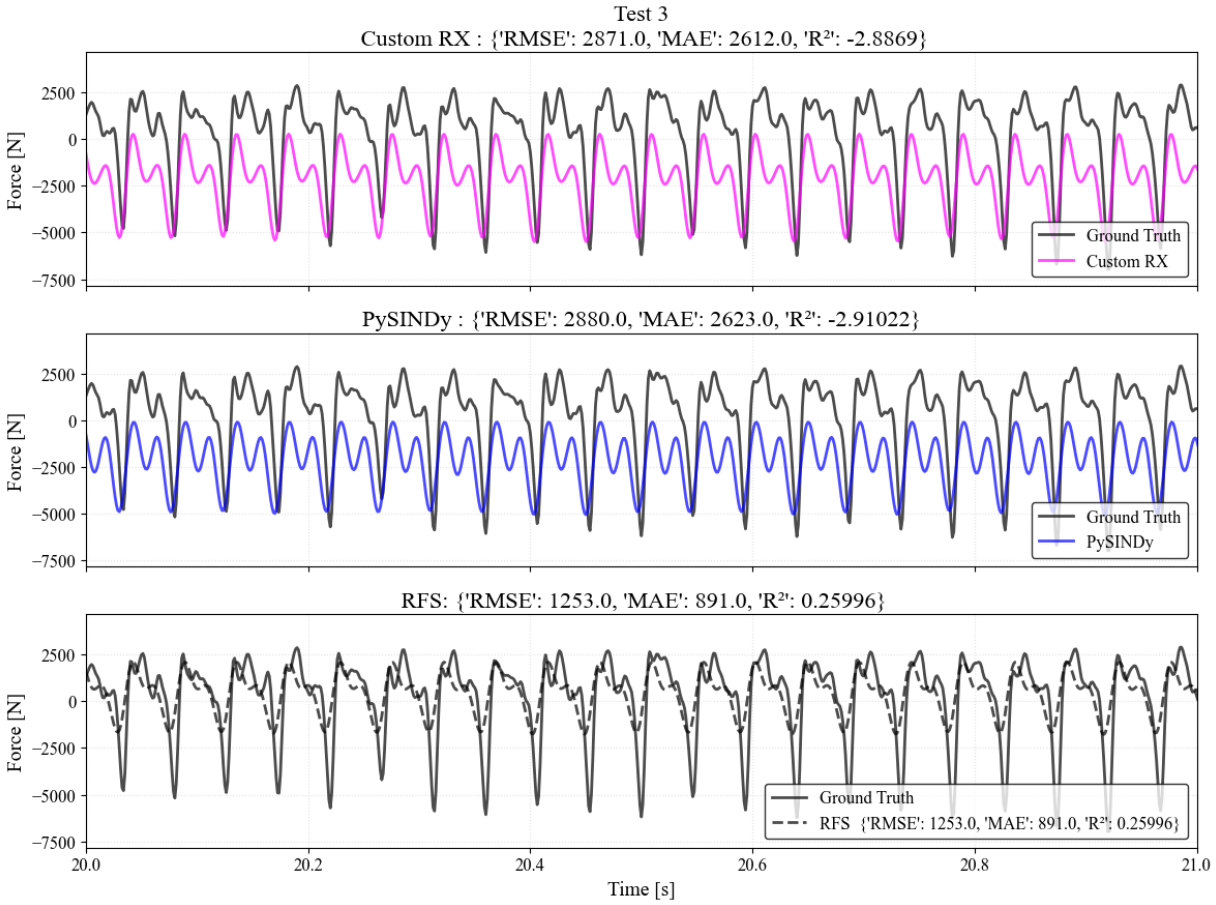


Figure N.62: Tip Force from Test 1 : Test 3 Data - Fit for t =2-40s

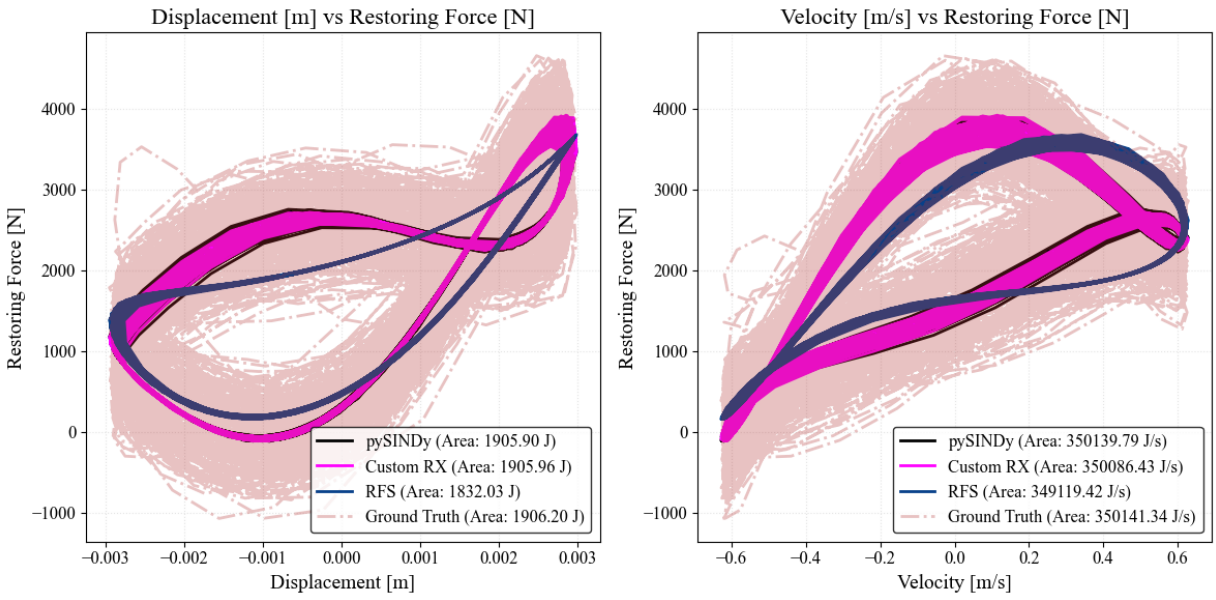


Figure N.63: Tip Force from Test 1: Fitted loading Cycles

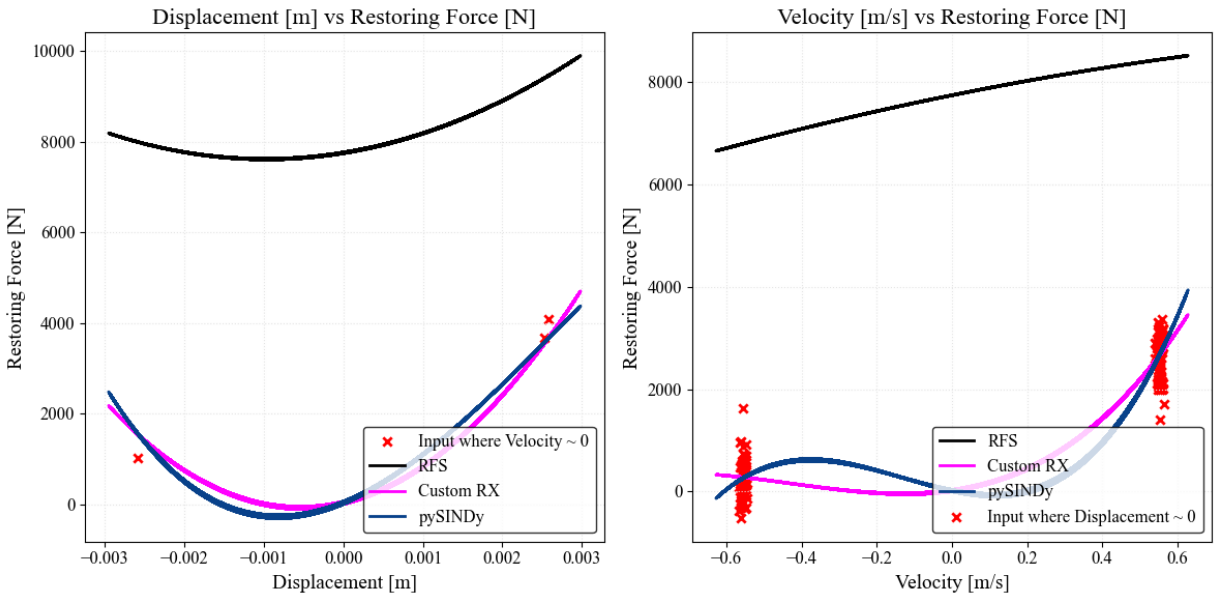


Figure N.64: Tip Force from Test 1: Fitted loading Cycles - Section

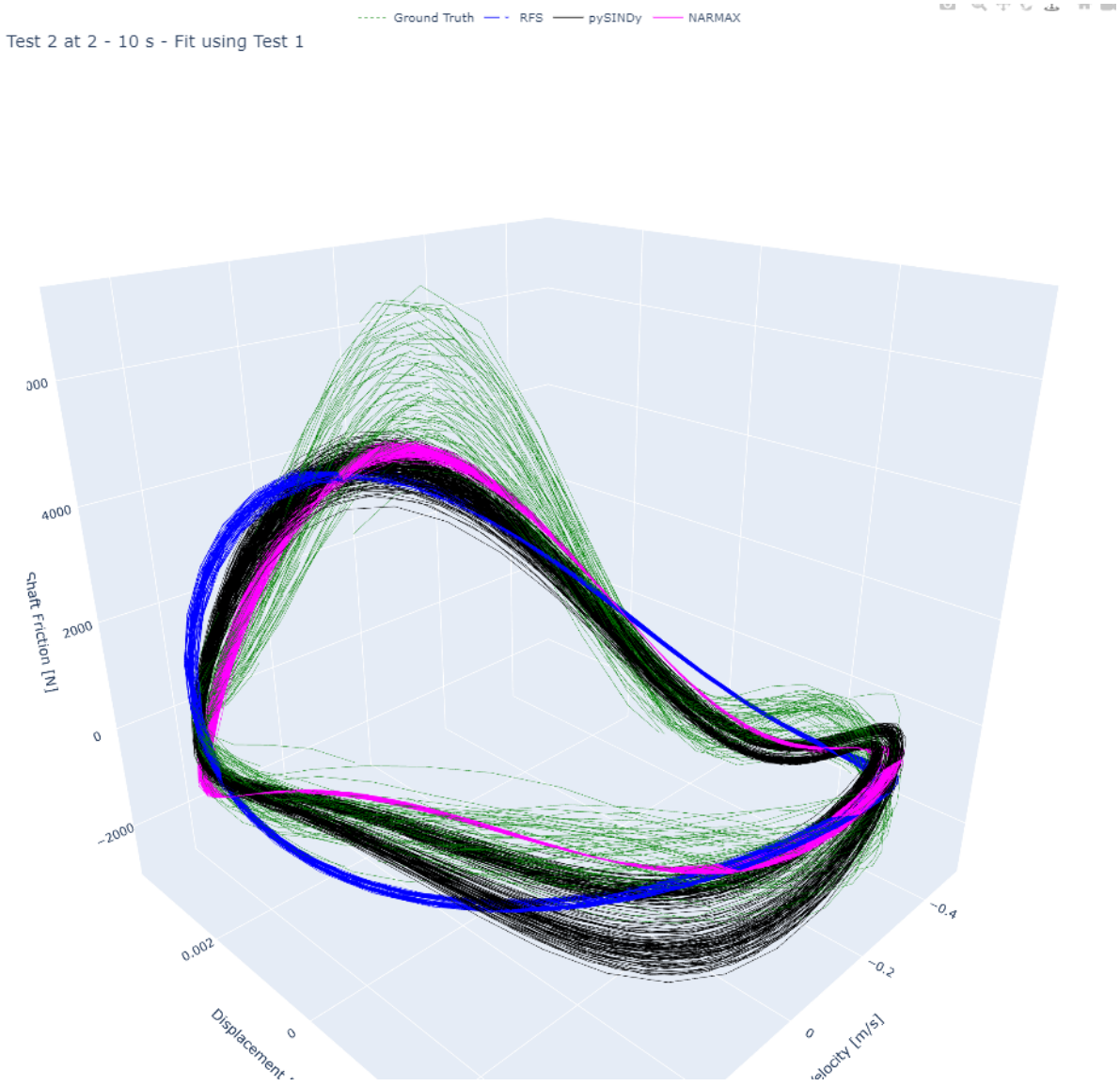


Figure N.65: 3D view of tip force from test 1

Results for Tip Force from Test 3

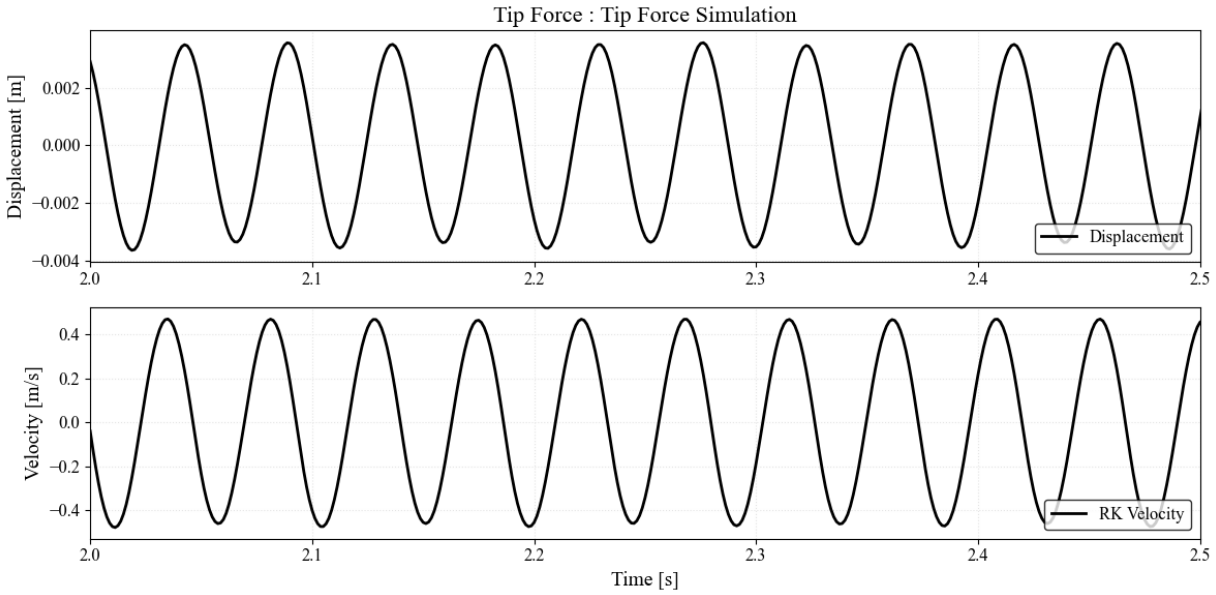


Figure N.66: Tip Force from Test 3 - Filtered Measurements

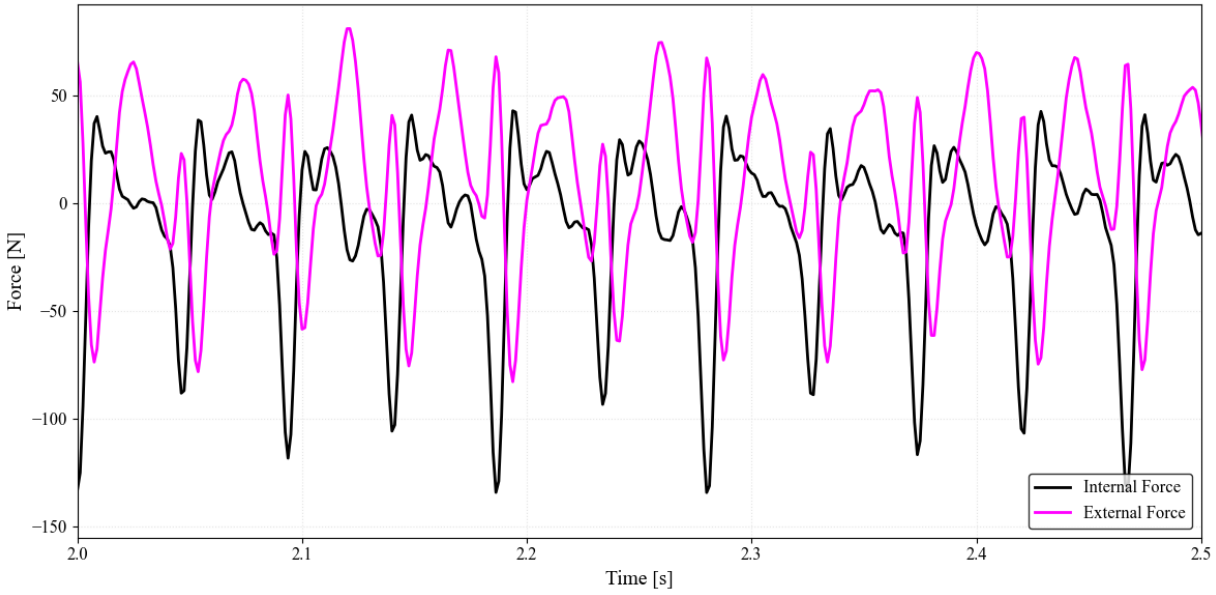


Figure N.67: Tip Force from Test 3 External forcing & Measured Tip Force

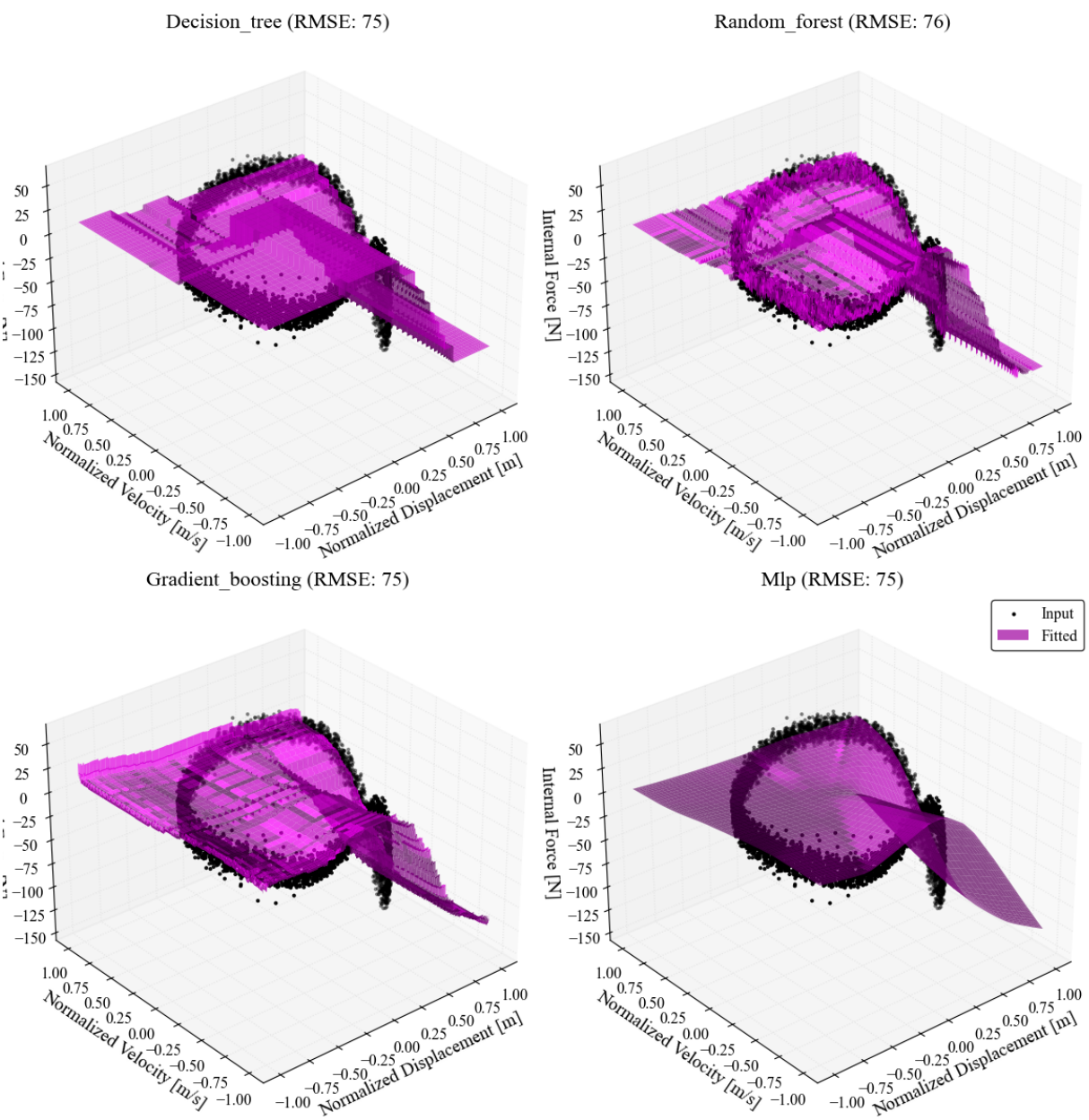


Figure N.68: Tip Force from Test 3 : Machine Learning fitted 3D Surfaces

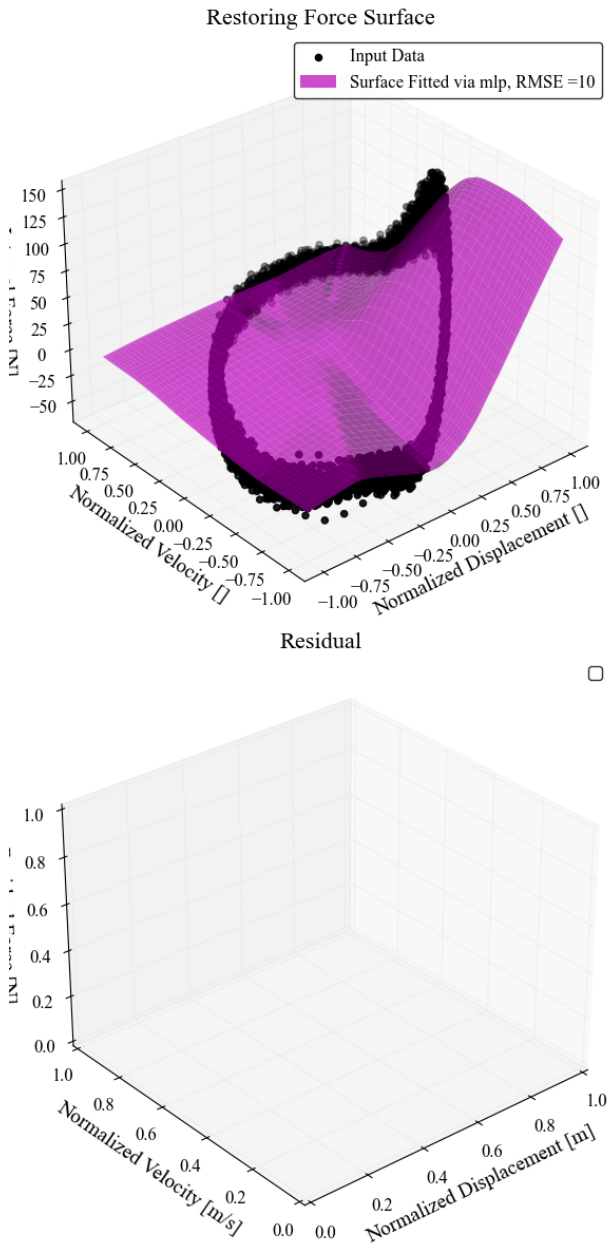


Figure N.69: Tip Force from Test 3: Selected Restoring Force Surface

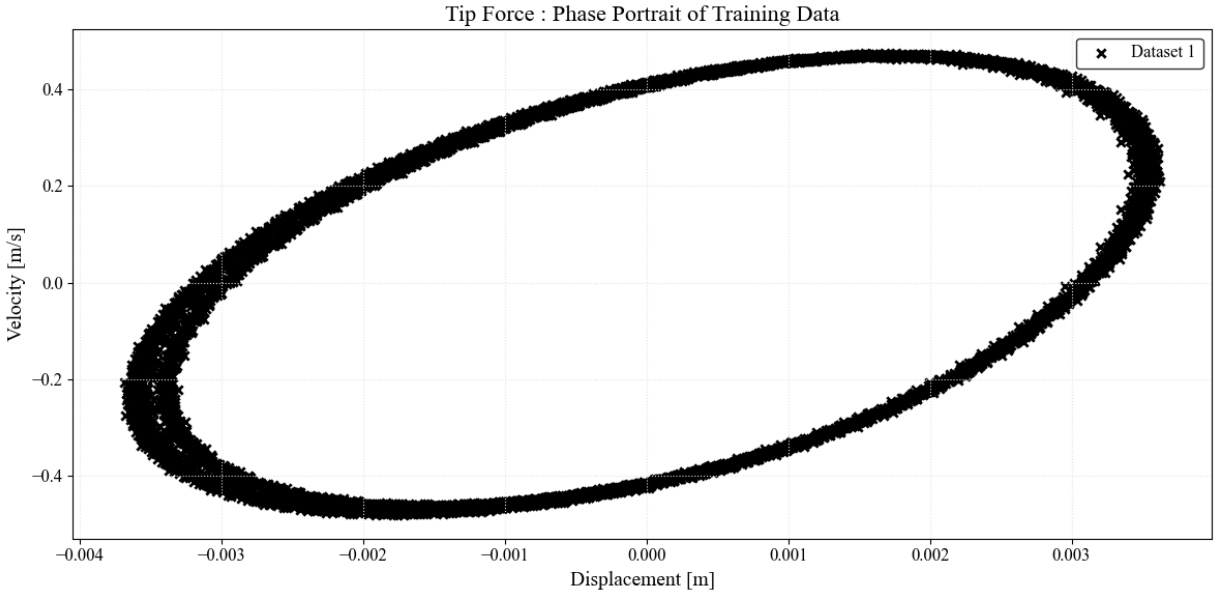


Figure N.70: Tip Force from Test 3: Phase Portrait of Training Data

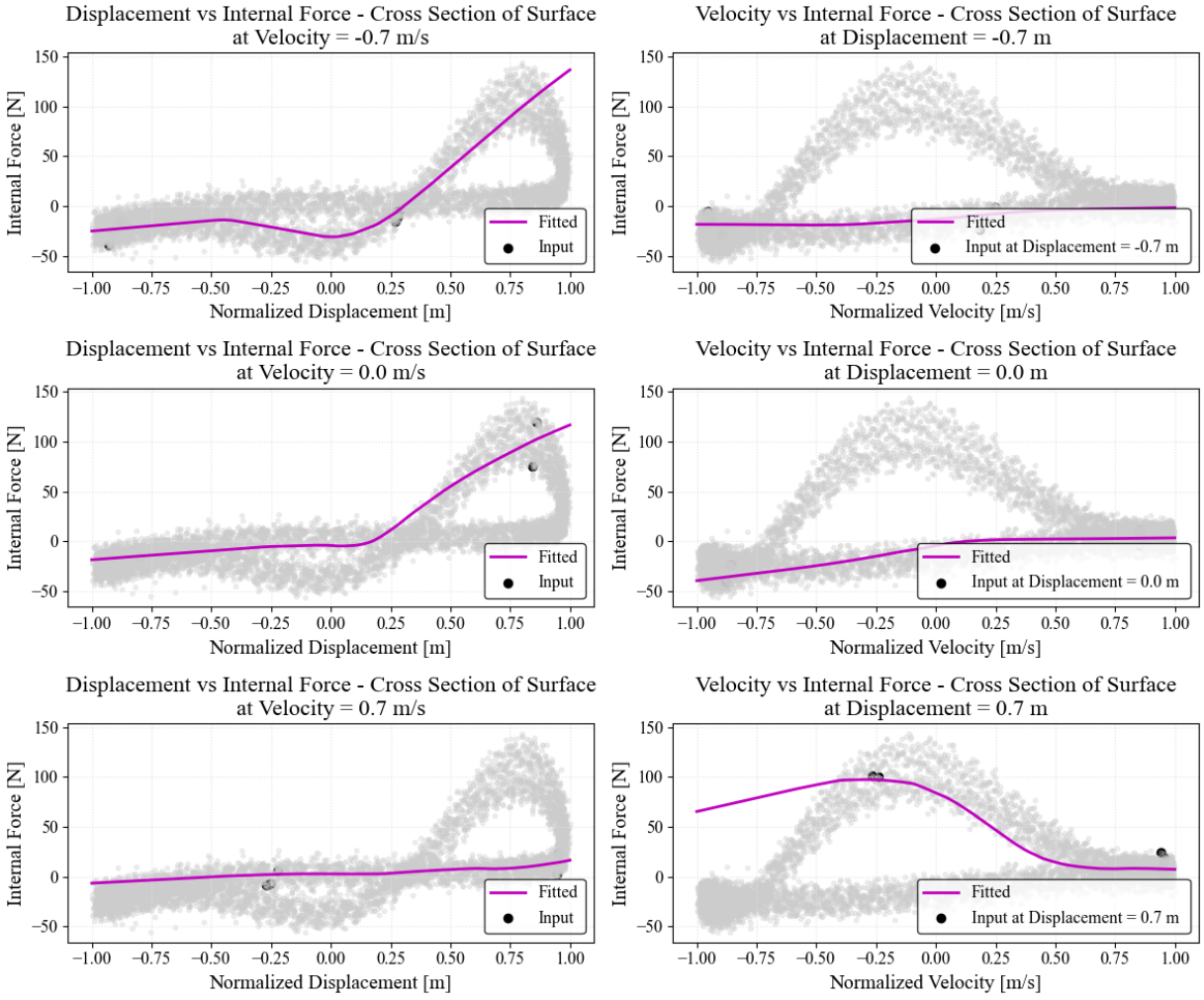


Figure N.71: Tip Force from Test 3 : Cross Sections of Surface

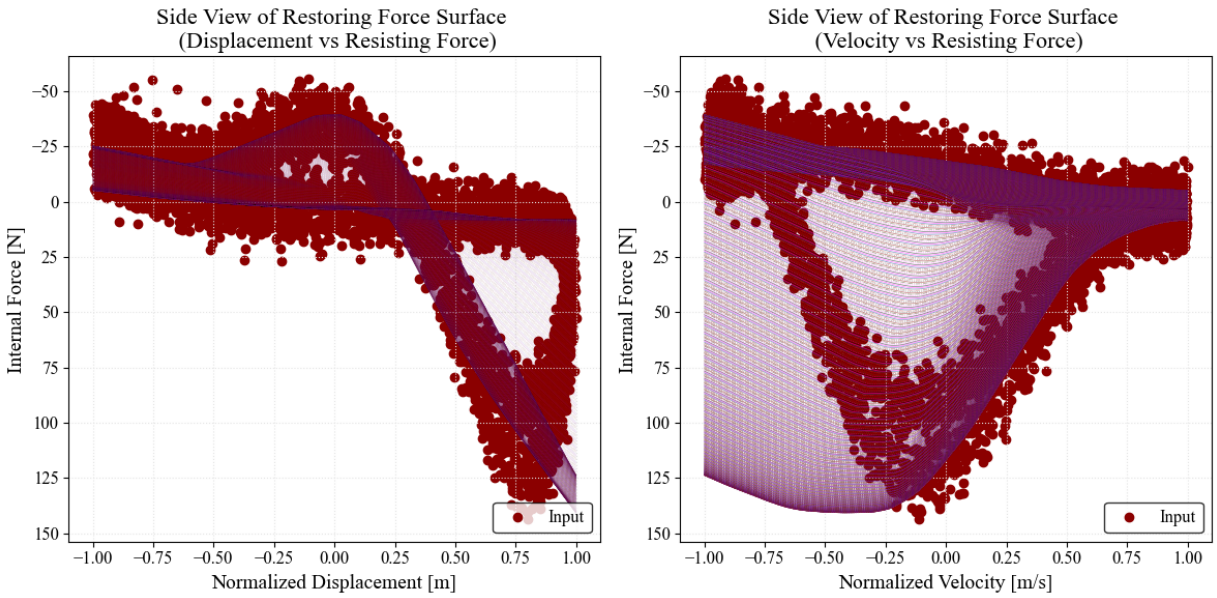


Figure N.72: Tip Force from Test 3 : Side Views of Surface

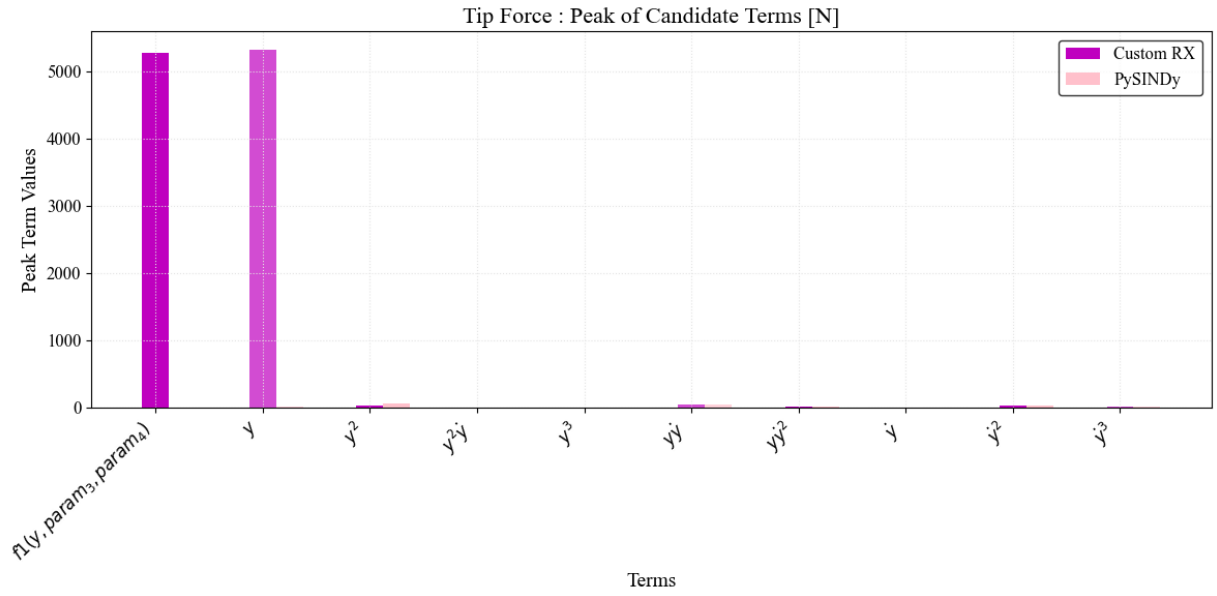


Figure N.73: Tip Force from Test 3 : Force Features comparison

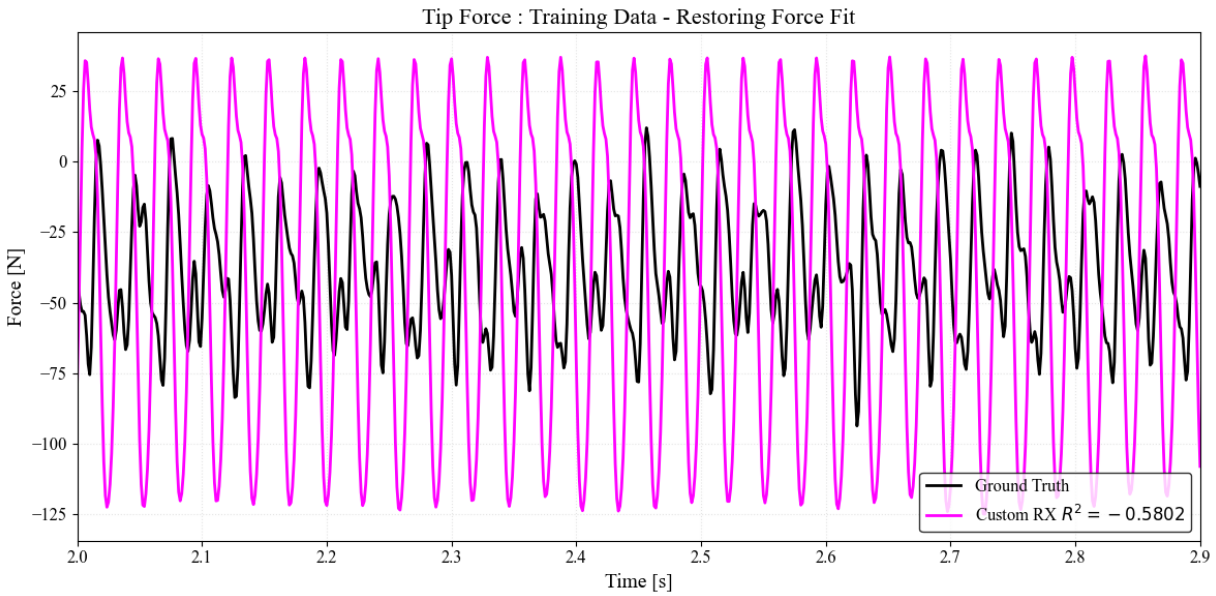


Figure N.74: Tip Force from Test 3 : Training Data - Restoring Force Fit

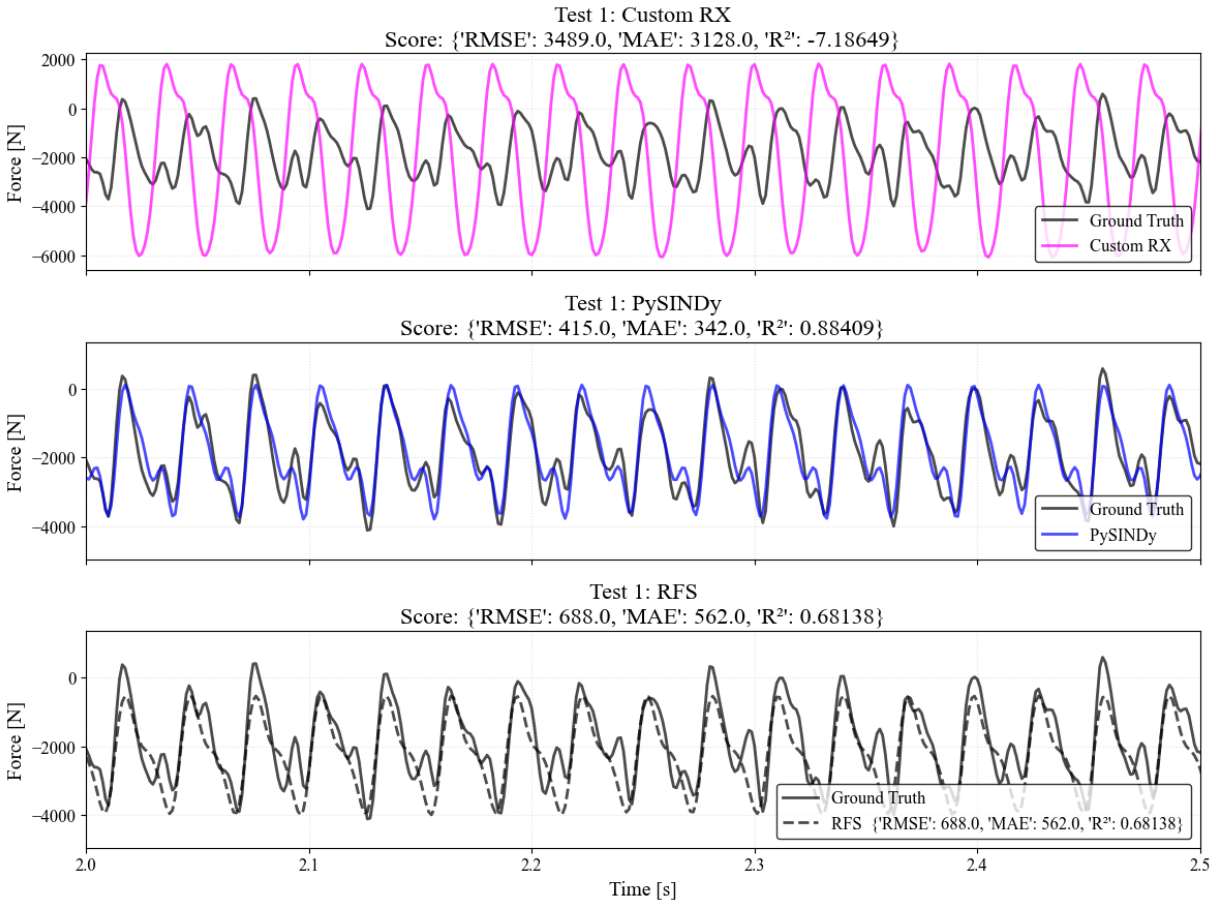


Figure N.75: Tip Force from Test 3 : Test 1 Data - Fit for t =2-10s

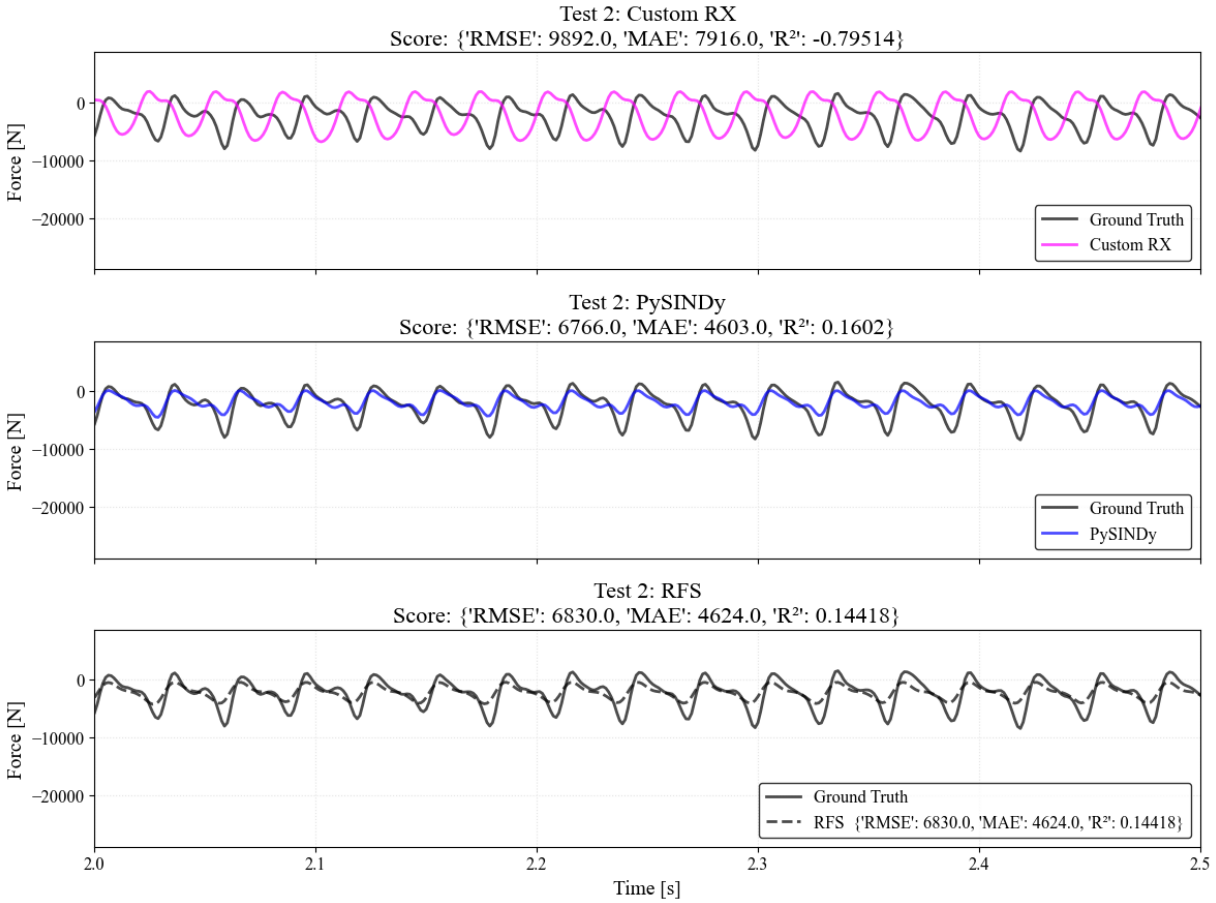


Figure N.76: Tip Force from Test 3 : Test 2 Data - Fit for t =2-10s

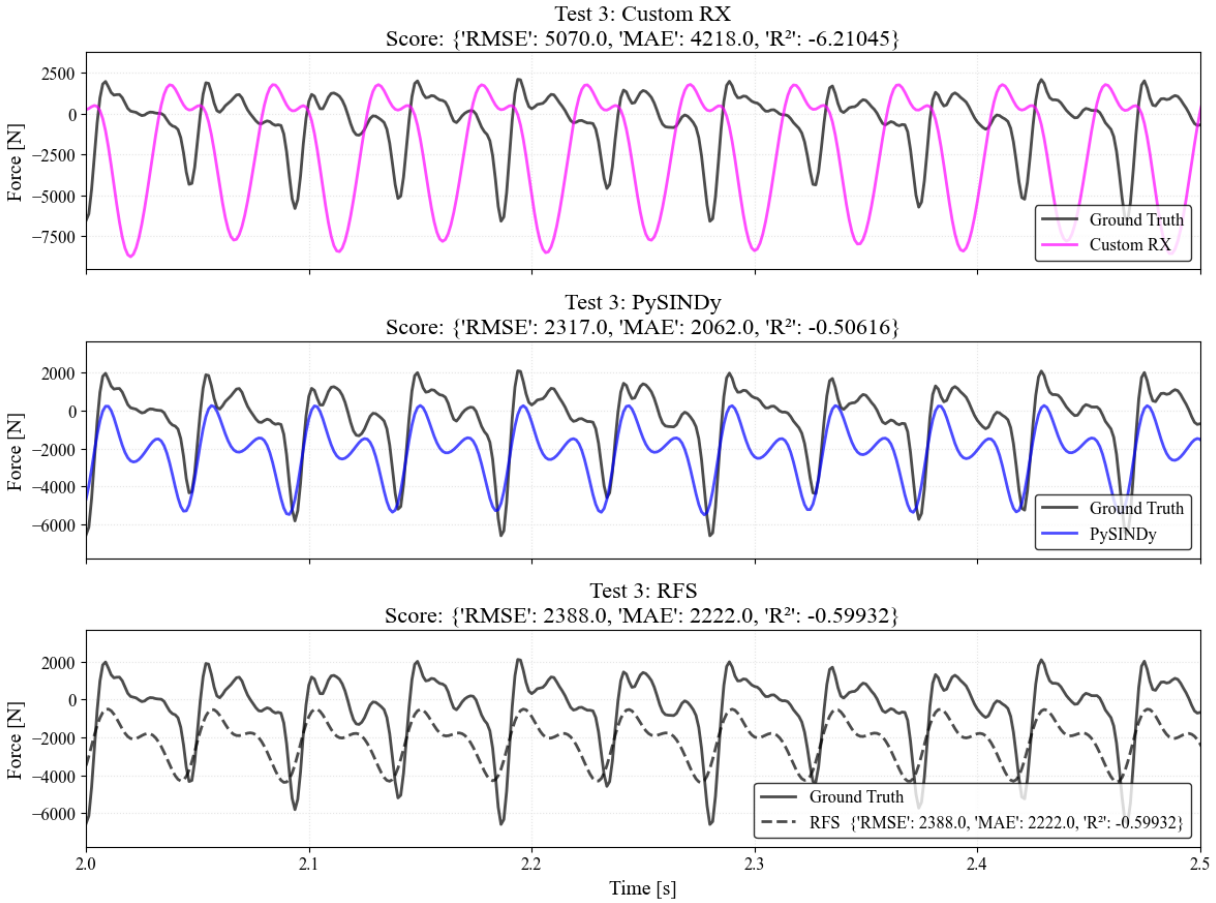


Figure N.77: Tip Force from Test 3 : Test 3 Data - Fit for t =2-10s

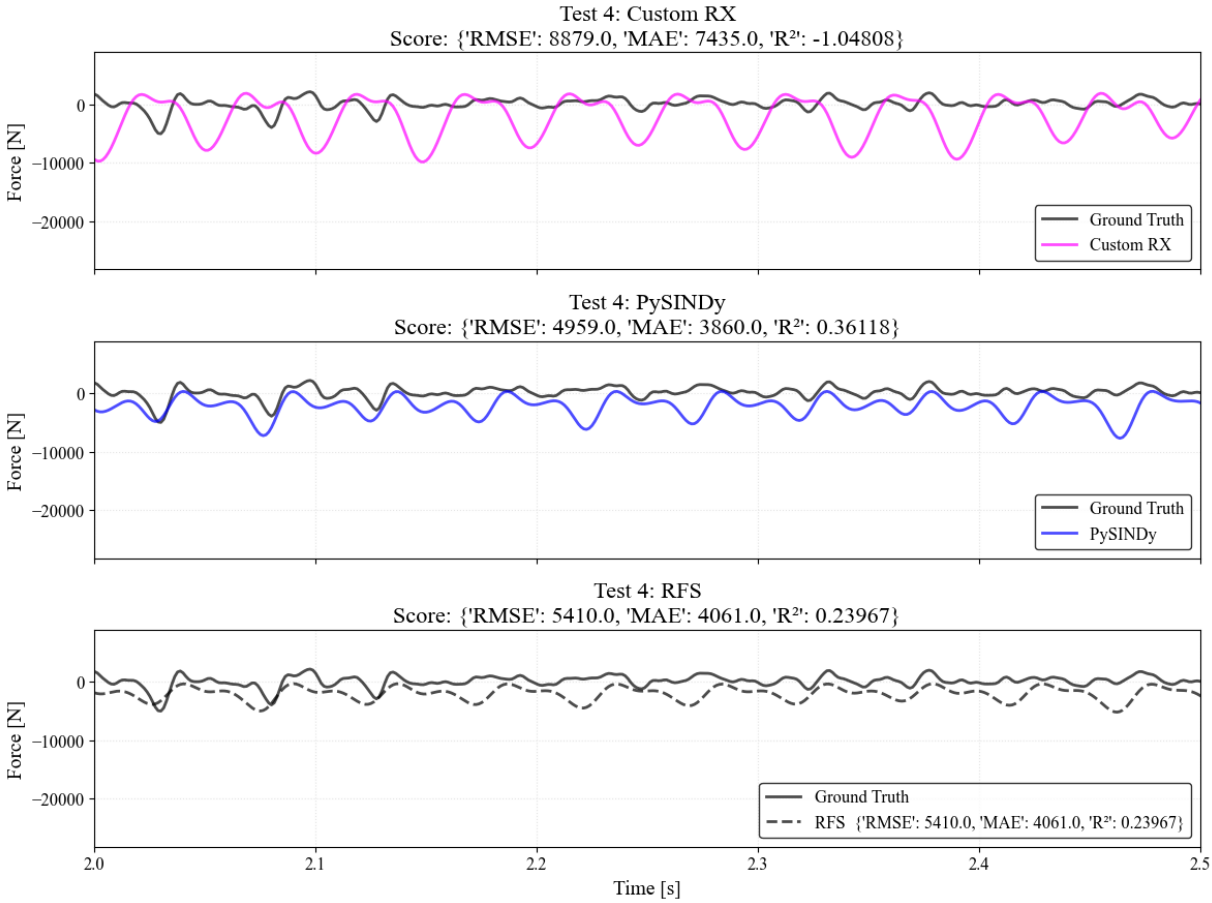


Figure N.78: Tip Force from Test 3 : Test 4 Data - Fit for t =2-10s

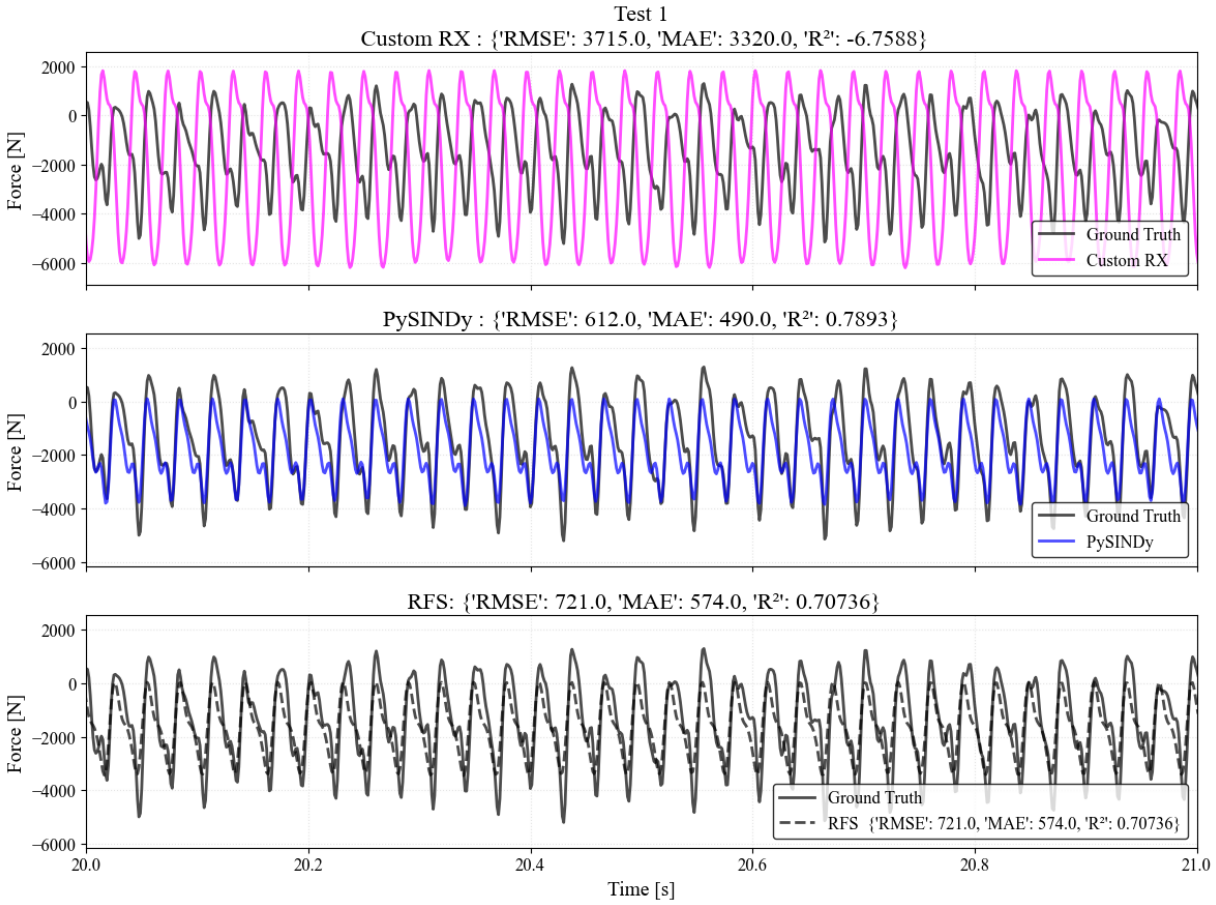


Figure N.79: Tip Force from Test 3 : Test 1 Data - Fit for t =2-40s

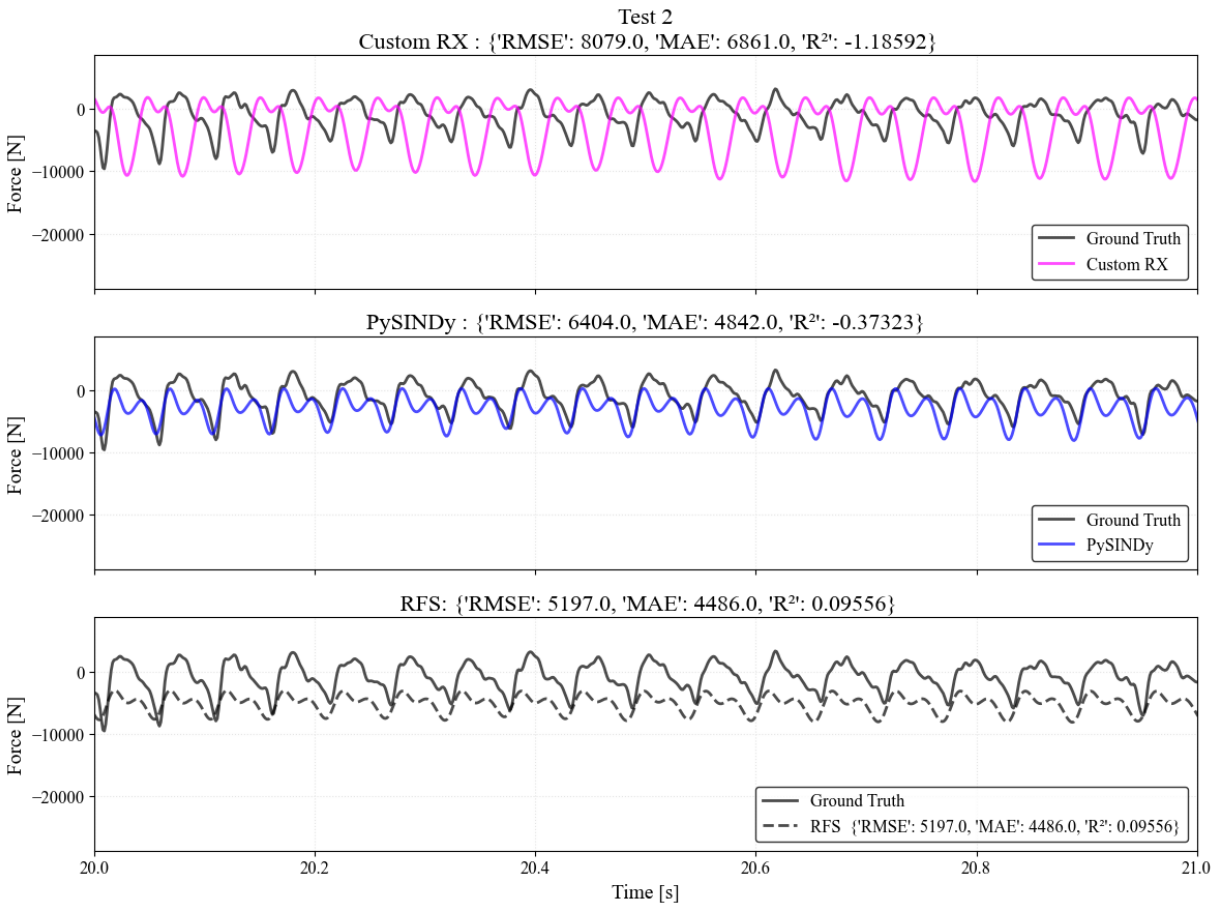


Figure N.80: Tip Force from Test 3 : Test 2 Data - Fit for t =2-40s

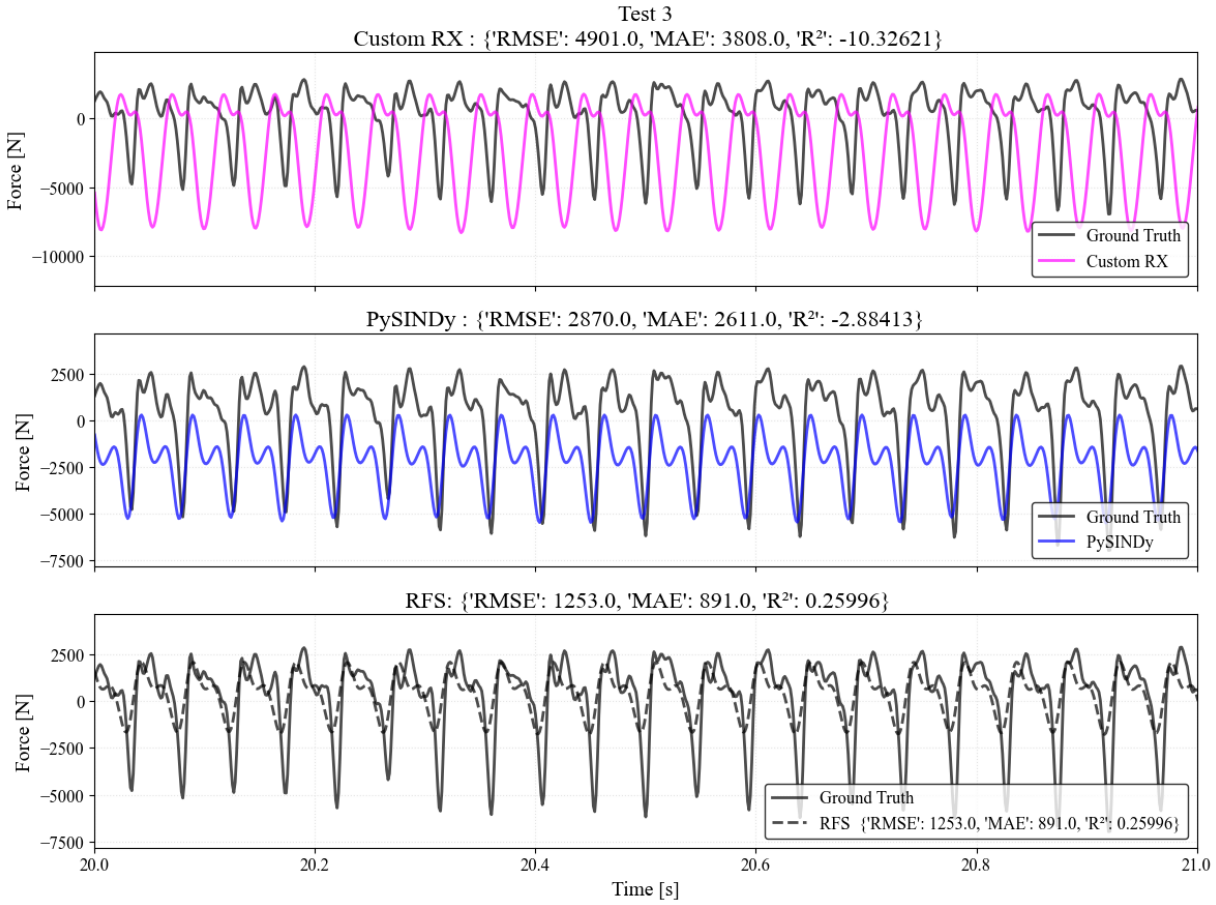


Figure N.81: Tip Force from Test 3 : Test 3 Data - Fit for t =2-40s

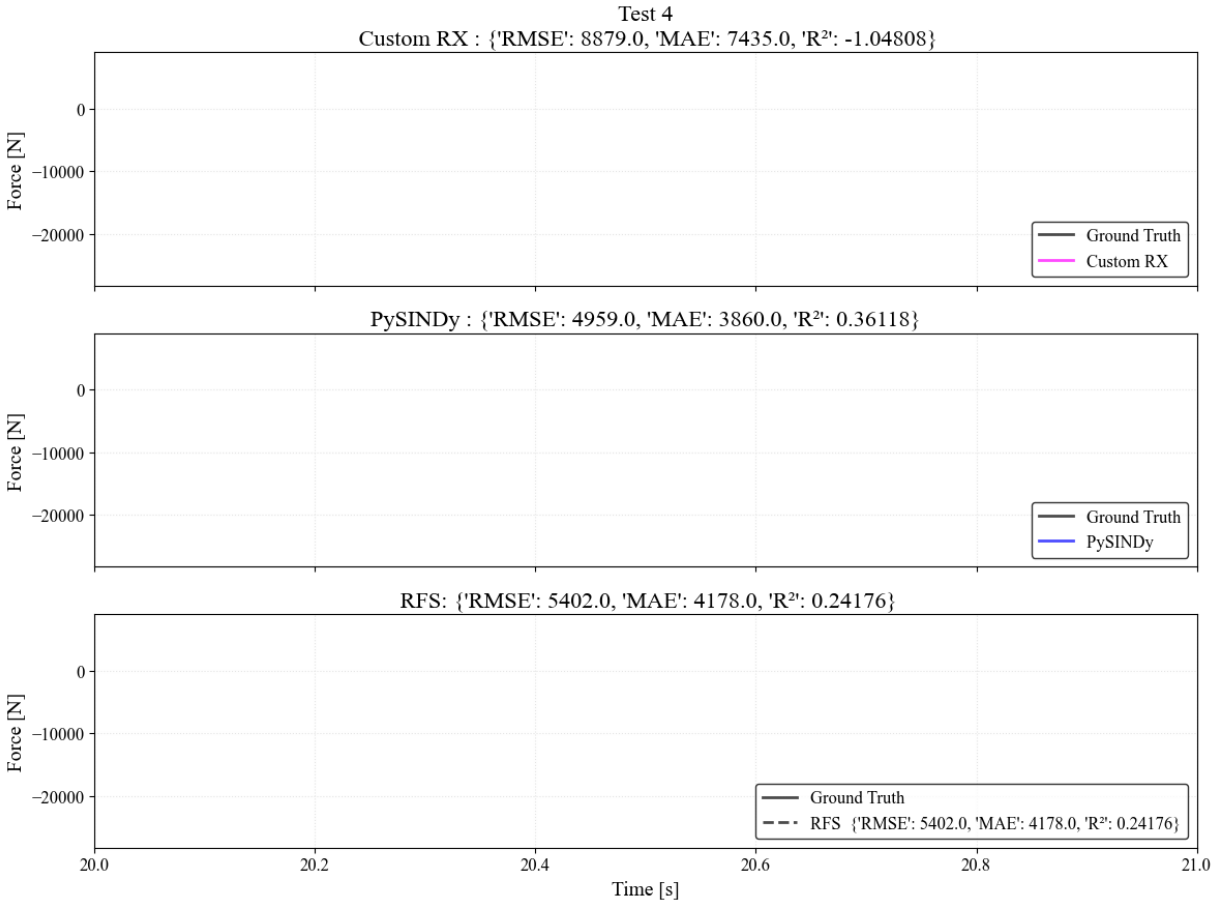


Figure N.82: Tip Force from Test 3 : Test 4 Data - Fit for t =2-40s

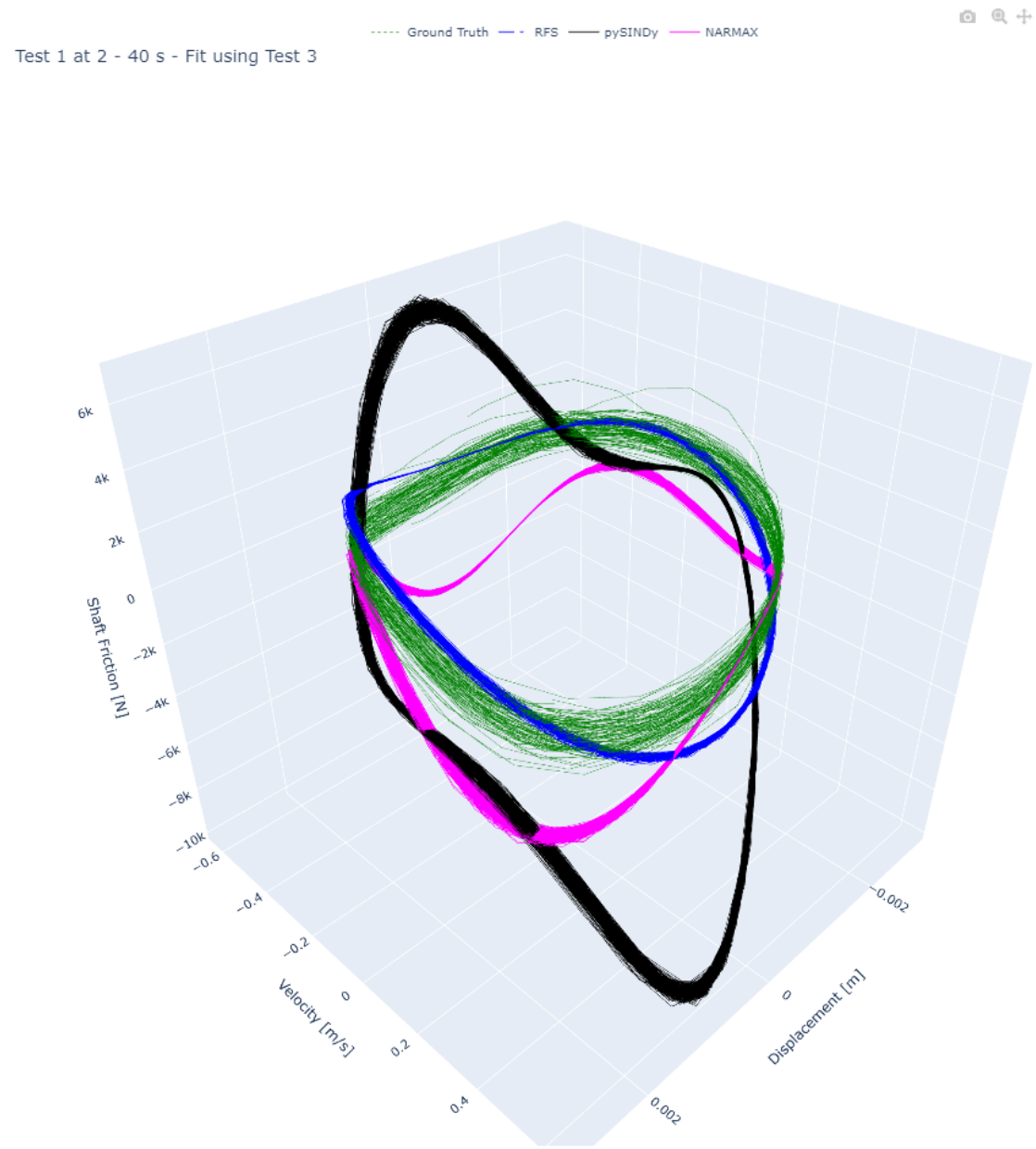


Figure N.83: 3D View 1 - Tip Force vs displacement and velocity from Test 3

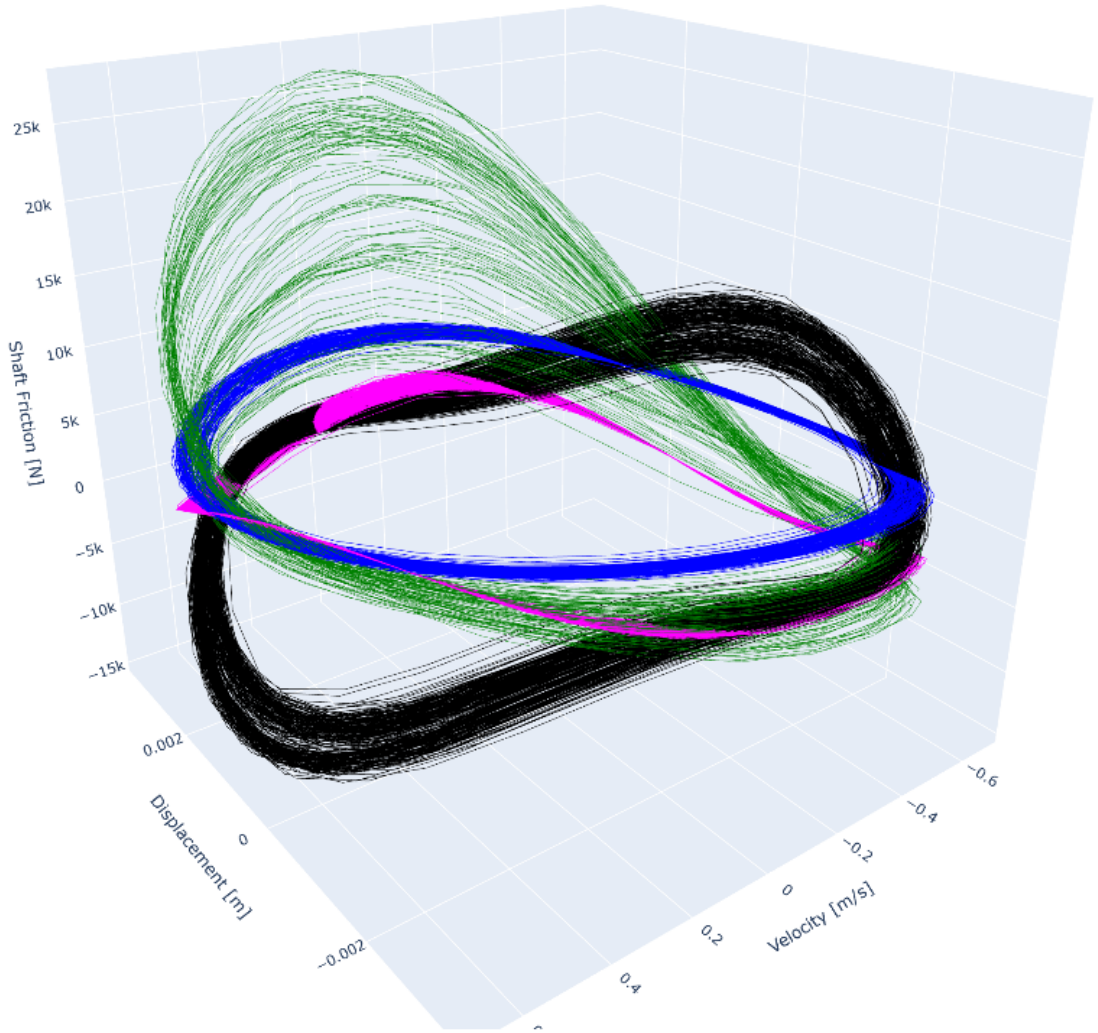


Figure N.84: 3D View 2 - Tip Force vs displacement and velocity from Test 3

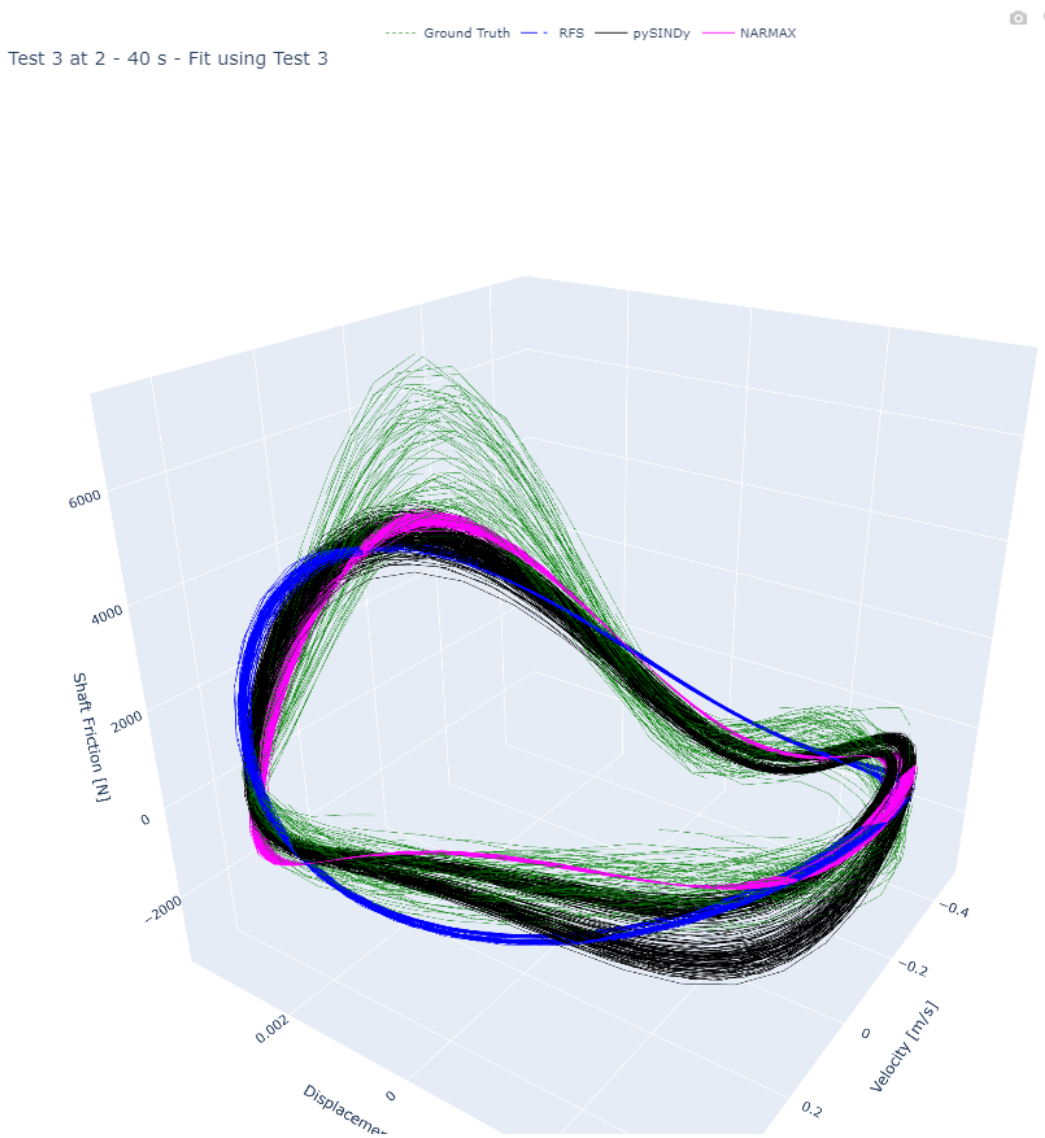


Figure N.85: 3D View 3 - Tip Force vs displacement and velocity from Test 3

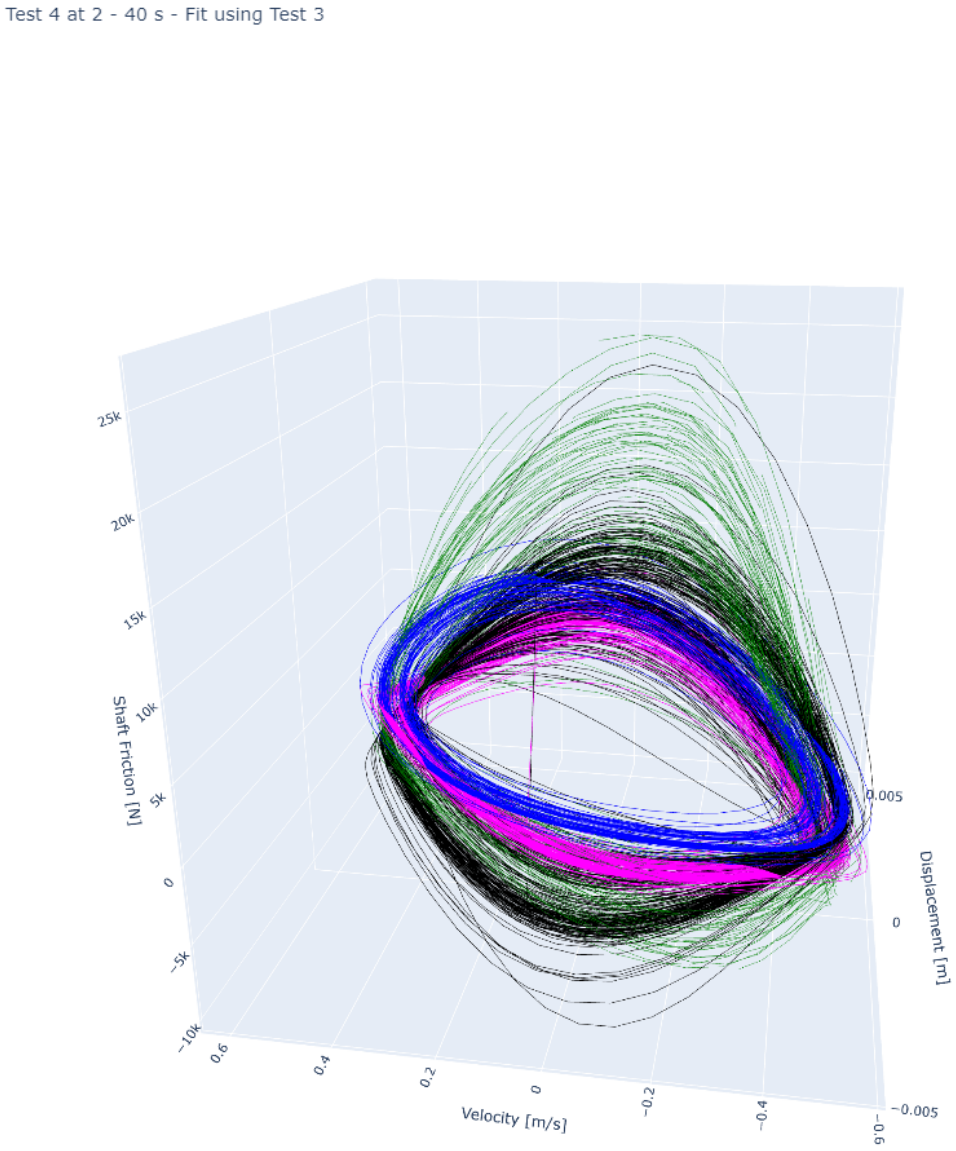


Figure N.86: 3D View 4 - Tip Force vs displacement and velocity from Test 3