Imperial College London

Department of Computing

Moving Obstacle Trajectory Forecasting

by

Cheu Teck Chung Brendan

Submitted in partial fulfilment of the requirements for the MSc Degree in Computing of Imperial College London

September 2023

Contents

1 Abstract

Motion planning for autonomous robots or vehicles requires both safety and performance. Motion planning has two parts:(i) predict the paths of moving obstacles and (ii) determine an avoidance strategy. Being able to accurately predict the trajectory of a moving object to estimate its potential future location is critical in ensuring safety. There are practical solutions when both robot and environmental factors are known, but in real applications, they cannot be assumed since real time conditions are variable and uncertain, and data is generally noisy. Existing solutions tackling unknown environments are heavily affected by noise. Sparse model identification is a potential solution that enables the discovery of nonlinear dynamical systems purely from data, and extensions such as ensembling and the weak formulation are proposed to tolerate noise. Our work implements and evaluates the weak formulation of SINDy in predicting the future motion of objects without prior knowledge of their behaviour. We also explored that having variable data size depending on error metrics can improve prediction accuracy. Hence our work implements a direct and simple elastic moving window approach that improves the model's robustness to objects that exhibit changes in their motion similar to the data drift behaviour in time series forecasting problems. The proposed moving elastic window increases or decreases by predefined size depending on whether model is getting more or less accurate with each prediction respectively. Our work shows that weak SINDy is able to accurately simulate future motion using only past coordinate data with no prior knowledge of the environment and that an elastic moving window is effective in adapting to changes in motion.

2 Acknowledgements

First and foremost, the completion of this project could have not been accomplished without the continuous support of my advisor, Dr Bahadir Kocer. I would also like to thank Dr Urban Fasel and Associate Professor Ronald Clark for the time they have devoted to the regular meetings throughout the project.

I am grateful for the support of my family and friends, for their continuous support and encouragement throughout the demands of this project. I would also like to thank my partner Jeanette, who has supported me and encouraged me through this project, both emotionally and through her selflessness.

Last but also most importantly, I am grateful to God for the opportunity to complete this project, and for the continuous grace He pours out on me.

3 Introduction

For autonomous robots or vehicles, safety and performance are critical factors for motion planning – which can be loosely defined as how robots decide what motions to perform in order to safely travel from point A to B (1) - when navigating real world environments. For example, autonomous vehicles must be able to safely transport passengers from their start to end points, and be able to do so in real time. There has been extensive research completed in relation to robot motion planning in dynamic environments (RMPDE) such as the works in (2; 3; 4; 5). The diagram in figure 1 further illustrates the diversity of research approaches in RMPDE.

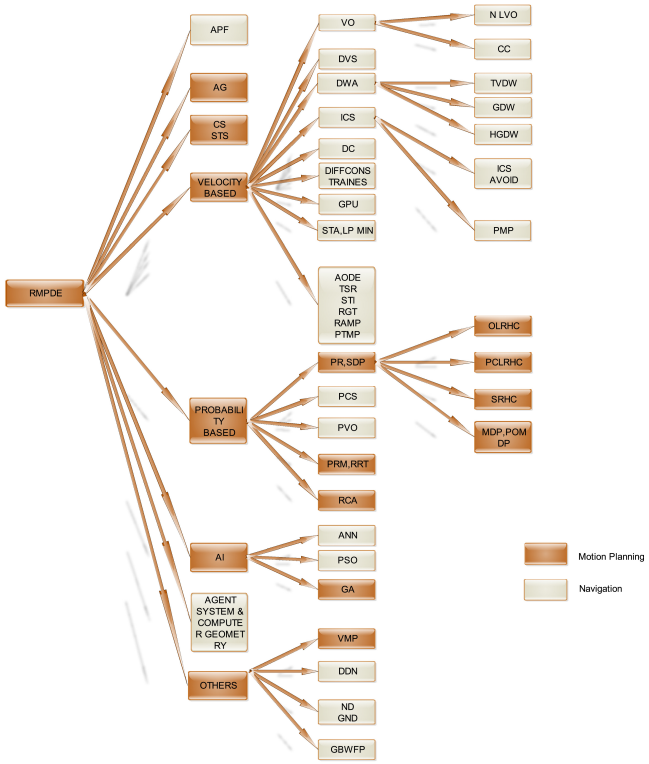


Figure 1: Diagrammatic representation of diverse research approaches to motion planning. Adapted from (6)

While humans have no apparent difficulty in navigating real world environments, it is difficult to duplicate this using an autonomous robot. The problem of two-dimensional motion planning with an arbitrary number of obstacles moving with bounded velocity is NP-hard (7). There are practical solutions that are effective when the robot and environmental factors are exactly known, but in real-world settings where those assumptions are no longer valid, safety is not guaranteed.

When environments are dynamic and not known in advance, the robot would need to reason about uncertainty. Within current research, there are predictive and reactive approaches. Reactive approaches such as active collision avoidance and reactive target-tracking (8; 9) usually model simplified dynamics and do not optimise results. Predictive approaches such as Singular Spectrum Analysis and Adaptive conformal prediction(10; 11) take into account predictions of the future trajectory of the dynamic agent and can optimise performance.

However, those predictive approaches each have their advantages and disadvantages. S.X.Wei et al(10) uses Singular Spectrum Analysis within its motion planning but it's performance degrades quickly when data input gets noisier. A.Dixit et al(11) uses adaptive conformal prediction which quantifies the true prediction uncertainty instead of heuristically, but the feasibility of the model predictive control (MPC) optimisation is worse than in Singular Spectrum Analysis (10).

This project forms part of the predictive approach. In the predictive approach, there are two steps:

1. Forecasting the dynamic object's future motion

2. Incorporating it into a MPC scheme that implements an optimal avoidance strategy.

This project will focus on implementing a new approach to the first step, and evaluating its performance. The main idea of the project is to use an extension of the sparse identification of the nonlinear dynamics (SINDy) algorithm first developed in discovering governing equations from data by sparse identification of nonlinear dynamical systems (12). The version of SINDy developed is a combination of weak SINDy with ensembling, first introduced in U.Fasel et al and Daniel A. Messenger et al(13; 14). SINDy is a sparsity promoting model that does not require any training, meaning that it can be utilised to find the dynamical system of an unseen object.

The code forecasts the trajectory of a dynamic moving obstacle without any pre-existing knowledge of the obstacle and its path. The weak SINDy algorithm is used to identify a model for the obstacle's motion in real time. The model of the obstacle's motion could in future be incorporated into a risk-aware MPC scheme that enables the drones to avoid dynamic moving obstacles. A SINDy-MPC model has better performance, needs less data, is more computationally efficient, and can better handle noise in data than Neural Network models (low data limit) (15). SINDy also has an advantage over Neural Networks in that it does not need to be trained. A model implementing SINDy could make predictions on the moving object just by being given trajectory data.

MPC is defined as an optimal control technique in which the calculated control actions minimise a cost function for a constrained dynamical system over a finite, receding, horizon (16). An MPC for drone movement planning plots the best flight path for a UAV from one point to another, while considering safety and performance.

To my knowledge, prior to this project, there have been no existing solutions that utilise SINDy or an extension of SINDy to predict trajectories based on high fidelity simulation data (such as those generated by GAZEBO). Current research only extends to testing SINDy using a Monte Carlo Simulation like in (10). This research evaluates SINDy's performance and limitations when it is applied to dynamical systems in which we do not know the true dynamical equations of the data.

There is a variety of potential applications for being able to predict an unknown moving object's trajectory, of which a few are listed below.

One future application of this project will be in relation to the project on "Environmental Sensing with Drones Towards Cyber Forests"(17). Currently, drones are controlled by humans. In future, the direction of the project is to have a swarm of autonomous drones for environmental sensing. For this to happen, the systems must be able to estimate the positions and flight path of other agents by predicting their trajectory based on camera data input.

Another application is in drone catching. Using information on the future motion of a drone, catchers could be placed in the drone's path path to catch it. For use cases such as in drone catcxhing using cooperative UAVs(18), the aim is to catch possibly dangerous UAVs. (the catcher is unlikely to have any prior knowledge of the UAV's movement.) In current systems such as (18), they rely on the assumption that the UAV to be captured will fly in a straight line between points. Although this currently holds true for most commercial UAVs, UAVs will no doubt become more capable of complex movements. When (not if) this happens, methods such as the one proposed in this project will be useful in providing accurate predictions based on limited and noisy data.

Other potential applications could be tracking moving aerial vehicles, such as planes, giving a real-time estimate on the future flight path of a plane just by providing coordinate data.

The contributions of this paper will be to:

- Apply the weak formulation of SINDy to predict the future motion of unknown objects modelled using a high fidelity trajectory simulator Gazebo, and evaluate its performance.
- Improve accuracy and robustness to sudden changes in motion by implementing an elastic moving window that specifies how much of the trajectory data is relevant for fitting.

4 Background and Related work

4.1 Preliminaries

4.1.1 Projectile motion

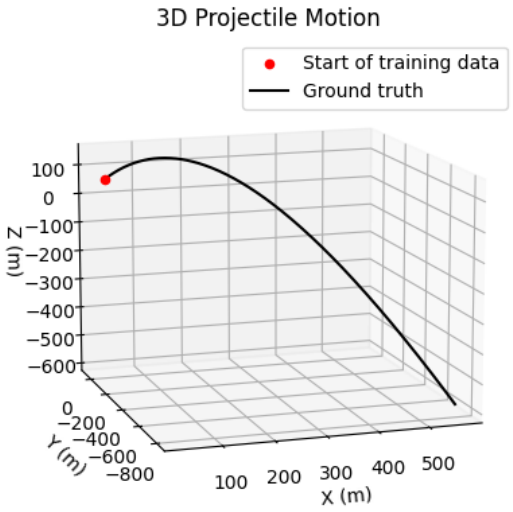


Figure 2: Trajectory data of a projectile motion with $x_0 = 30$, $y_0 = 40$, $z_0 = 60$, $v_0 = 40$, $\theta = 30°$.

As a simple test case, a way to generate three dimensional projectile data is defined using parabolic motion. While there are several other simple motions that could be used, parabolic motion is utilised as a test case as it has been extensively studied in D.Bajc, S.K.Bose, F.D.Medina(19; 20; 21). The coordinate data is generated using the projectile motion equations:

$$
\begin{aligned}
x &= x_0 + v_{x0} \cdot t \\
y &= y_0 + v_{y0} \cdot t - 0.5 \cdot g \cdot t^2 \\
z &= z_0 + v_{z0} \cdot t - 0.5 \cdot g \cdot t^2
\end{aligned}
$$

where v_0 is the initial velocity.

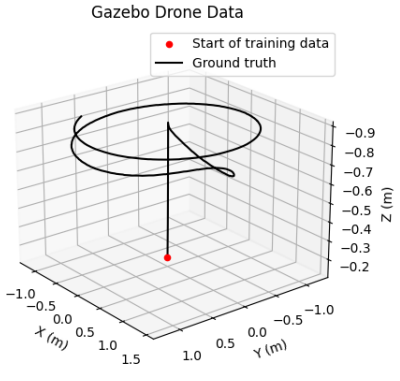
4.1.2 Gazebo

In this work, test trajectory data of a UAV is generated using a open source software called Gazebo. Gazebo is a widely used 3D dynamic simulator that has the ability to accurately simulate robot movements in complex indoor or outdoor environments. It can be likened to game engines, but it offers physics simulations at a much higher degree of fidelity, a variety of sensors and interfaces for both users and programs. Examples of Gazebo data are as shown in Figure 3.

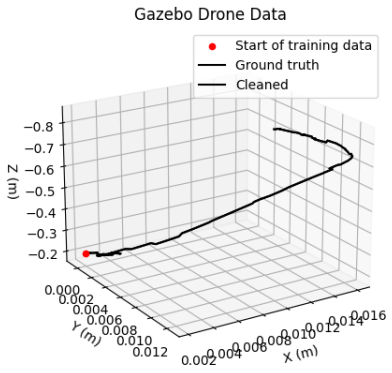
Gazebo is used for the following situations:

1. Realistic Simulation: Gazebo provides a realistic 3D simulation environment, accurately simulating the physical dynamics and sensors of a UAV.

2. Trajectory Generation and Testing: Gazebo can generate trajectories of different dynamics and gather the relevant data needed without extensive setup.



(a) Trajectory data of UAV takeoff and performing a circular motion generated by Gazebo



(b) Trajectory of UAV takeoff generated by Gazebo

Figure 3: Examples of test datasets

An existing GitHub project available at : https://github.com/fdcl-gwu/uav_simulator is adapted, converting the output of save function on the control GUI to a format that the model expects. The adapted code saves only the time, x-coordinates, y-coordinates and z-coordinates of the UAV in a csv file. To generate trajectory data, the requirements are to (a) use ROS melodic and Ubuntu 18.04 on a virtual machine, and (b) follow the instructions for the ros-melodic branch. csv files of different UAV behaviours are generated, from taking off, to the UAV making a circle, to landing. These datasets can be found in this project's GitHub repository.

4.1.3 Frisbee

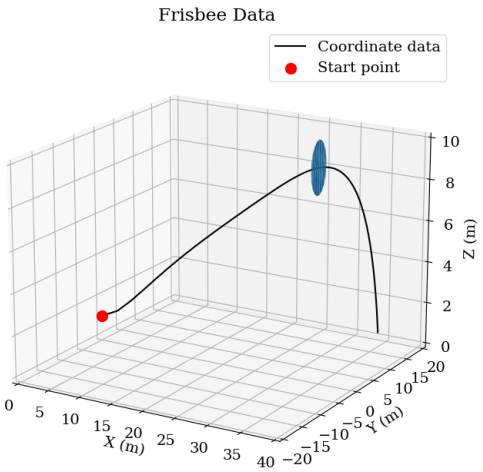


Figure 4: Trajectory data of a frisbee with $vx_0 = 40$, $v_y = 10$, $\theta = -0.25rad$.

The frisbee is often used as a test case for evaluating trajectory prediction accuracy because it presents several interesting challenges that are relevant to real-world applications. Firstly, it has non-linear flight dynamics influenced by various factors (such as spin, angle of release, wind, and airfoil shape), making it challenging to model accurately. It is also sensitive to the initial conditions, such as the angle and speed of the throw, that can result in significantly different trajectories. Lastly, it also has been researched extensively, meaning that there is extensive existing data

modelling a Frisbee's flight. For example, a Frisbee's flight can be simulated (22).

To generate test data trajectories of a Frisbee for testing, I used the python Frisbee model package Frispy which is available at: https://github.com/tmcclintock/FrisPy. The package can simulate trajectories of discs with varying initial conditions, while also changing the underlying physical model.

The data for a few different Frisbee trajectories has been generated and stored in csv files.

4.1.4 AIC

The Akaike information criterion (AIC)(23) is an information criteria-based relative fit index. It gives an approximation of the out-of-sample predictive accuracy of a model based on the available data. AIC gives an estimation of prediction error. Therefore it is used as a mathematical method to evaluate how well the model fits the data it was generated from. AIC can be used to compare between different possible models, and determine the best fit for the data by picking the minimum. AIC is calculated from the number of independent variables used to build the model and the maximum maximum likelihood estimate of the model (how well the model reproduces the data).

Mathematically, AIC is calculated using the following equation:

$$AIC = -2 \cdot \frac{l}{n} + 2 \cdot \frac{k}{n}$$

where n is the number of data, k is the number of estimated parameters (regressors+intercepts) and l is the log likelihood function where l is calculated using the following equation:

$$\ell = -\frac{n}{2}\left(1 + \ln(2\pi) + \ln\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right)\right)$$

The best model will have the lowest scored AIC. This will be the model that best fits the data while utilising the fewest possible independent variables.

4.1.5 SR3 optimizer

The Sparse relaxed regularized regression (SR3) optimizer introduced in (24; 25) is an extension of the default Sequentially thresholded least squares (STLSQ) introduced with standard SINDy in (12).

The STLSQ optimizer is as follows:

$$\min_{u} 0.5|y - Xw|^2 + \lambda R(w) \tag{1}$$

where $R(w)$ is a regularizer that promotes sparsity.

The math for the SR3 optimizer is as follows:

$$\min_{w,u} 0.5|y - Xw|^2 + \lambda R(u) + (0.5/\nu)|w - u|^2 \tag{2}$$

where $R(u)$ is a regularization function.

As outlined in (25), SR3 has three main advantages: (i) having features to handle corrupt data such as through trimming; (ii) the ability to impose constraints and (iii) the ability to learn parametrizations.

One advantage of SR3 is that it can be adapted to utilise trimming of outliers, which is problematic for model identification methods such as SINDy as they corrupt derivative computations. A trimming faction argument can specify what fraction of the samples should be trimmed. This removes terms with coefficients below the trimming faction from the model, reducing what would look like overfitting.

In this project, SR3 is to be preferred over the default STLSQ is used in normal SINDy. This is because is SR3 involves a few more hyperparameters that can be tuned for greater accuracy when identifying parsimonious models from noisy data.

4.2 Related work

Planning in dynamic environments have broad interests, as shown in diagram 1. In terms of the closest work related to this work, the "Moving obstacle avoidance: A data-driven risk-aware approach" is most similar, in terms of using a predictive method to predict the trajectory of the moving obstacle.

4.2.1 Singular Spectrum Analysis

In S.X.Wei et al.(10), the paper proposes a new predictive approach to dynamic obstacle avoidance using Singular Spectrum Analysis (SSA) from Bagging prediction algorithms(26; 27) to distinguish the noise in the data and extract a predictive model of the obstacle behaviour. Bootstrapping – a sampling with replacement method – is applied to obtain a set of obstacle trajectory forecasts. The paper then incorporated the predictions into an MPC planner that performs dynamic obstacle avoidance.

While other approaches handling dynamic obstacles have used obstacle trajectory/model classification, (7) proposes a method that does not utilise classification while still enabling online computation. This is advantageous as classification-based methods require identifiable object behaviour and prior knowledge about the dynamic environment, which may not always be possible.

The paper is relevant as the bootstrap-SSA-forecast architecture proposed can be computed online, making it suitable for use in a UAV that would need to compute this in real time.

Monte Carlo Simulations of the MPC planner were conducted, with one dynamic obstacle introduced in each run. Three different types of dynamic obstacles were tested, with the results bench-marked against an artificial potential field alternative.

4.2.2 SSA - an ensemble forecast

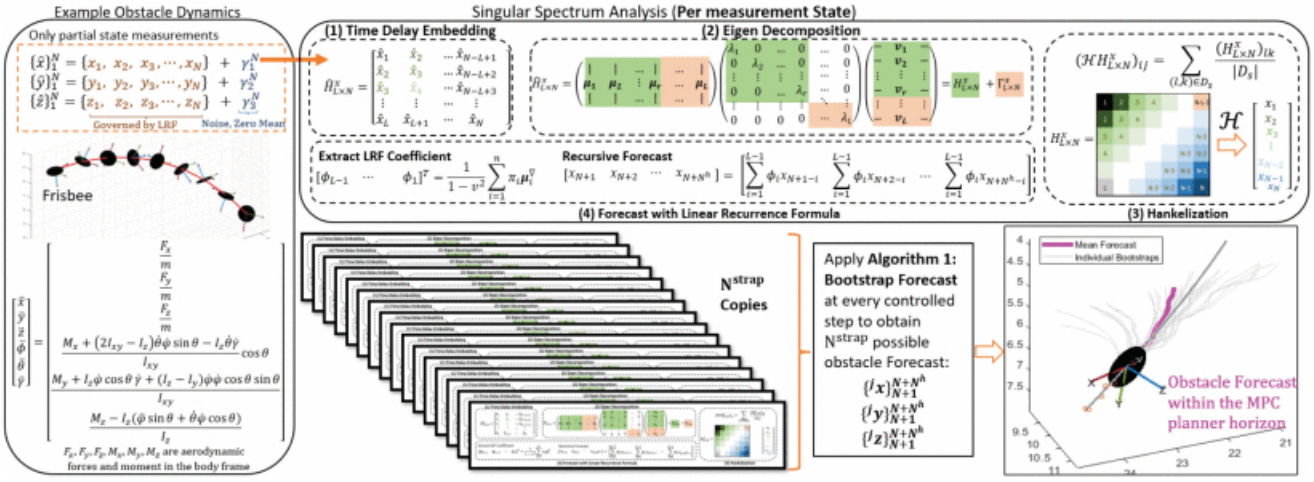


Figure 5: A description of bootstrap-SSA-forecast architecture in forecasting the trajectory of a Frisbee where the stochastic observables (corrupted by zero-mean, noise) consist of $\{\hat{o}\}_1^N = \left[\{\hat{x}\}_1^N, \{\hat{y}\}_1^N, \{\hat{z}\}_1^N\right]$
, the Frisbee's center positions with respect to an inertial frame. The SSA analysis and bootstrap forecast is applied to every observable state. Despite its 12-state governing dynamics and with only centre position measurements of the Frisbee, we show an example N^{strap} forecasts of the Frisbee trajectory for future time steps $\{1, 2, \ldots, N^h\}$ using our proposed framework. Adapted from S.X.Wei(10).

In (11), the paper proposes an Adaptive Conformal Prediction (APC) algorithm that uses the data collected online to create prediction regions that quantify multistep-ahead prediction uncertainty and a MPC that leverages the uncertainty quantification to plan safe paths around dynamic obstacles. The difference between the approach in SSA(10) vs APC(11) is that the former quantify prediction uncertainty heuristically, where the latter quantify the true prediction uncertainty.

The Monte Carlo simulation result showed that the APC method can avoid dynamic obstacles such as a Frisbee with a significantly smaller divergence distance (minimum distance between obstacle and drone), while having a

similar success rate. However, the feasibility of the model predictive control (MPC) optimisation is worse than in SSA.

For both SSA and APC, the results were not tested on high fidelity data such as the one that is generated by Gazebo and only on obstacles with simple motions such as a ball dropping or Frisbee. The accuracy of the prediction also drops significantly in the presence of noise, which is characteristic of real data.

4.2.3 Sparse Identification of Nonlinear Dynamics (SINDy)

The SINDy algorithm first developed in(28) is a data-driven model discovery framework that utilises sparsity-promoting techniques and machine learning to discover the governing equation(s) of a dynamical system purely from data measurements. Data-driven model discovery is useful as deriving the underlying governing equation of an unknown obstacle is not possible ahead of time. While there exists other data-driven models, they have limited effectiveness when data is limited and noisy, which is characteristic of the data we get from the UAV. SINDy's potential lies in its ability to balance accuracy and efficiency when discovering models.

E-SINDy is an extension of SINDy. Sparse model identification is sensitive to noise, especially when data is low or noisy. E-SINDy leverages bootstrap aggregation to be maintain accuracy while increasing robustness in model discovery when data is limited or noisy. Limited or noisy data is to be expected of real data from cameras or sensors.

Weak SINDy is another extension to SINDy that tackles noisy and limited data using a novel weak formalization and discretization(14). While E-SINDy does better than standard SINDy with regards to noisy data, in the presence of very limited or noisy data it still will not be able to identify any equation at all. Moreover, it is unable to handle spatiotemporal data or high-dimensional measurements (12), which is the data we work with when identifying trajectories of moving obstacles. Lastly, while SINDy and e-SINDy are able to identify ODEs from data, they cannot identify Partial Differential Equations (PDEs). Hence Weak SINDy further improves the robustness to noise of this work.

The SINDy algorithm assumes that most dynamical systems only have a few relevant terms that define the dynamics. Using sparse regression, SINDy find the relevant terms from a library of candidate terms.

The goal is to identify models of nonlinear Ordinary Differential Equations (ODEs) of the form

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{u} = \mathrm{f}(\mathbf{u}), \quad \mathbf{u}(0) = \mathbf{u}_0 \tag{3}$$

With state $\approxeq \in \mathbb{R}^n$ and dynamics $\mathrm{f}(\mathbf{u})$, and for nonlinear Partial Differential Equations (PDEs) of the form

$$\mathrm{u}_t = \mathrm{N}\left(\mathrm{u}_1, \mathrm{u}_x, \mathrm{u}_{xx}, \ldots, x, \mu\right) \tag{4}$$

with $\mathbb{N}(.)$ a system of nonlinear functions of the state $\mathrm{u}(x,t)$, its derivatives and parameters μ ; partial derivatives are denoted with subscripts, such that $u_t := \partial\mathbf{u}/\partial t$.

4.2.4 Standard SINDy

Measure $\mathbf{m}$ snapshots of the state $\mathbf{u}$ in time and arrange these into a data matrix

$$\mathrm{U} = \left[\mathrm{u}_1 \mathrm{u}_2 \cdots \mathrm{u}_m\right]^T \tag{5}$$

Next, we compute the library $\mathbb{D}$ of candidate nonlinear functions $\Theta(\mathbf{U}) \in \mathbb{R}^{m \times D}$.

$$\Theta(\mathbf{U}) = \left[1\mathbf{U}\mathbf{U}^2 \cdots \mathbf{U}^d \cdots \sin(\mathbf{U}) \cdots\right] \tag{6}$$

This library includes any functions that might describe the data. This is an important step since the underlying dynamical system is not known and we cannot guarantee that the library contains functions that can model the dynamics of the dynamical system. It is recommended to start with low order polynomials first, before adding higher order polynomials until a sparse and accurate model is obtained.

Next, we also compute the time derivatives of the state $\mathbf{U}_t = \lceil \mathbf{u}_1 \mathbf{u}_2 \cdots \mathbf{u}_m \rceil^T$. The system in equation may then be written in terms of these data matrices

$$\mathrm{U}_t = \boldsymbol{\theta}(\mathrm{U})\boldsymbol{\Xi} \tag{7}$$

Every term in $\boldsymbol{\Xi} \in \mathbb{R}^{D \times n}$ is a coefficient of a term in the dynamical system. Since there are only a few active terms in the governing equation of the dynamics, we use sparse regression to identify a matrix of coefficients $\boldsymbol{\Xi}$ that is a good fit:

$$\arg\min_{\mathbf{z}} \frac{1}{2} \left\| \mathbf{U}_t - \boldsymbol{\theta}(\mathbf{U})\hat{\boldsymbol{\Xi}} \right\|^2 + R(\hat{\boldsymbol{\Xi}}) \tag{8}$$

The regularizer $R(\hat{\boldsymbol{\Xi}})$ is chosen to promote sparsity in $\boldsymbol{\Xi}$. The exact choice of regularizer depends on the type of differential equation being found – ODEs or PDEs – and how we want to optimize it. In (6), a sequentially thresholded least-squares (STLS) (28) is proposed for ODEs, and sequentially thresholded ridge regression (STRidge) (29) is proposed for PDEs.

Standard SINDy has limitations. Like other model discovery methods, the accuracy of the prediction drops significantly in the presence of noise or when there is limited data. SINDy is sensitive to noise because it relies on the precise calculation of derivatives. And this problem is greater when modelling PDEs as noise can be amplified when it comes to higher-order spatial derivatives.

4.2.5 Ensemble SINDy

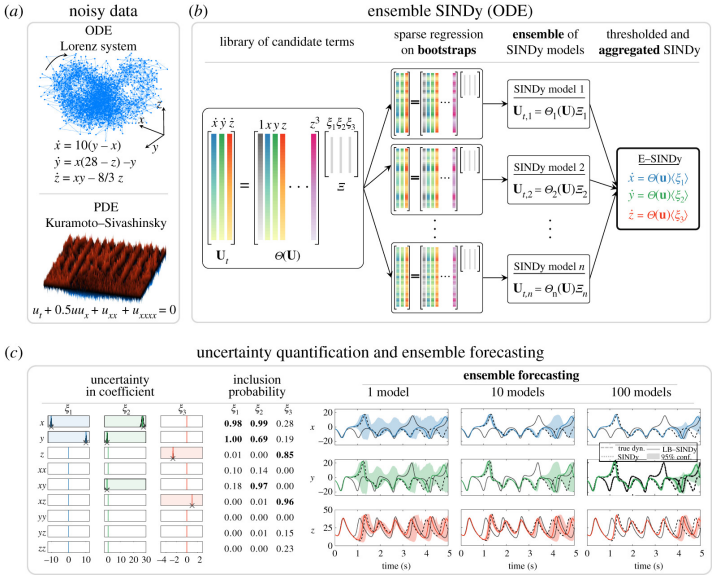


Figure 6: Schematic of the E-SINDy framework. Adapted from (6)

(13) proposed an extension to SINDy called E-SINDy as shown in figure 6. Ensembling means to sub-sample the data and generate n models by regressing each of the sub-samples. Ensembling was introduced to overcome some of the limitations of SINDy: specifically in its accuracy when the data is limited and noisy. Ensembling makes the regression more robust against outliers. The n models are combined and either the mean or median of the models is used in order to produce an optimal predictive model.

There are several classical ensembling methods introduced in (13). Ensembling techniques include bagging (bootstrap aggregation) (15) which takes the mean of the model, bragging (robust bagging) (30) which takes the median, and boosting (31). For E-SINDy, two types of ensemble model discovery methods are proposed: b(r)agging E-SINDy and library E-SINDy. Ensembling in PySINDy can be done with or without replacement.

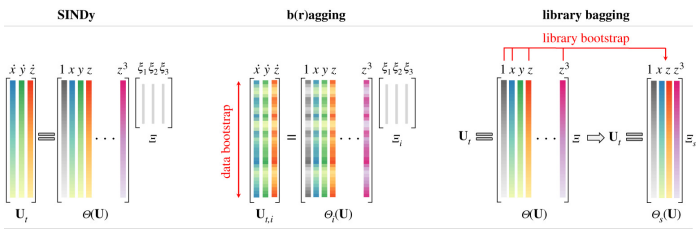


Figure 7: Schematic of SINDy and E-SINDy with b(r)agging and library bagging. Adapted from (6)

The different types of ensembling are shown in Figure 7.

B(r)agging E-SINDy utilises bootstrapping. Bootstrapping involves randomly drawing elements from your data with replacement and concatenating them into a new data set. A set of models is aggregated by taking the mean (for bagging) or median (for bragging) of the identified coefficients.

With regards to library bagging E-SINDy, this samples library terms instead of data pairs. We sample l out of library terms without replacement. Replacement does not affect the sparse regression problem when it comes to sampling library terms. The advantage of library E-SINDy is that a smaller library will improve model identification significantly. This is due to the complexity of the least-squares algorithm being $\mathcal{O}(ml^2)$, thus bagging with low l (smaller libraries) is better. As with Bagging SINDy, an ensemble of models and model coefficient inclusion probabilities are used to get the Library E-SINDy model.

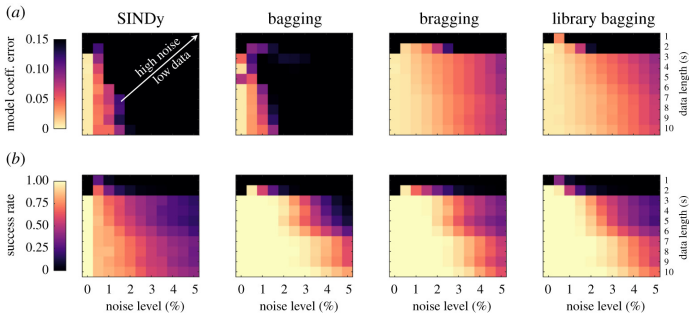


Figure 8: Comparison between different SINDy and ensemble SINDy models when noise level and data length changes on the Lorenz system. Ensembling is shown to improve accuracy of SINDy while needing less data and having more noise present. Adapted from E-SINDy(13)

In figure 8, ensembling is shown to help SINDy with model discovery, improving the accuracy while requiring less data. Bragging is shown to do better than bagging in the low data limit with little or no noise. This is due to the fact that outliers weigh more heavily in bagging (which takes the mean). Thus, bragging (which takes the median) is not as affected by outliers, meaning it is recommended to use bragging when data size is small.

4.2.6 Weak SINDy

Weak SINDy is a proposed extension of SINDy that does not have the same limitations as the original SINDy. It was first introduced in Messenger et al.(14) and extended in a later paper by the same authors(32), as an improvement on standard SINDy. Weak SINDy is able to select, from a large library, the correct linear, nonlinear, and spatial derivative terms, resulting in the identification of ODEs or PDEs from data (33). Weak SINDy identifies the weak formulation of the differential equation, meaning that high-order derivatives of noisy data are not identified in the model (high-order derivatives of noisy data will amplify noise). Only those terms that are most informative about the dynamics are selected as part of the discovered ODE or PDE.

The advantages of Weak SINDy as compared to standard SINDy are as follows. First, by considering the weak form of equations, Weak SINDy does not evaluate pointwise derivatives at all, which in standard SINDy causes significant accuracy issues. Furthermore, Weak SINDy uses the linear operator in its integration operation, which results in a weighted least squares framework that decreases the error rate as compared to ordinary least squares.

In Rudy et al.(33), Weak SINDy is used in the PDE-FIND algorithm to identify the dynamical system. The functionalities on PDE-FIND are built on three key concepts: (i) using over-complete libraries of candidate functions to represent the dynamics, (ii) utilising sparse regression to select a small number of terms, and (iii) employing the Pareto analysis to parsimoniously select the governing equations.

5 Methodology

This section seeks to explain how the project works and the design decisions made, while leaving the detailed explanation of features to their respective sections. The flow of my model is shown in figure 9. The code repository can be found here: https://github.com/ah19190/moving-obstacle-trajectory-prediction.

Algorithm 1 Model algorithm for csv file

import_csv_data
▷ Get start and end time for the while loop
start_time_index ← FIND_TIME_INDICES(t, t[0], MIN_WINDOW_SIZE)
start_time ← t[start_time_index]
end_time ← $t[-1]$
▷ Declare rmse_score
rmse_score ← 0
window_size ← MIN_WINDOW_SIZE
▷ This is the part where I fit and predict every PREDICTION_FREQUENCY seconds of data
while start_time ≤ end_time − PREDICTION_FREQUENCY **do**
    FIT(start_time, window_size)
    rmse_score_new
    rmse_score_new ← PREDICT(start_time, window_size)
    start_time ← start_time + PREDICTION_FREQUENCY
end while

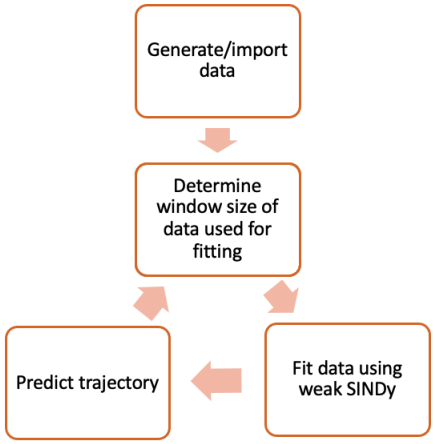


Figure 9: Diagram of the flow of my code. First, the data is either generated or imported from a csv file. Second, the model starts with a default window size of data to be passed into the data. Third, the most parsimonious dynamical model is fitted using weak SINDy. Fourth, the fitted model is used to simulate the future trajectory. Lastly, the accuracy of the prediction is used to inform the model whether to increase or decrease the window size for the next cycle.

5.1 Importing data

The only input the model requires is the coordinate data over a period of time of the object whose trajectory we want to predict. This data can either be generated and passed to SINDy as a numpy array or imported from a csv

file.

For the data generation option, the code currently generates the data using the specifications for the projectile motion equations, and saves it as numpy arrays.

For importing data, my project accepts csv files with the variables arranged in the following order: time, x-coordinate, y-coordinate and z-coordinate. The data is checked to ensure that time is strictly increasing, with any data which is not strictly increasing removed, before being separated into numpy arrays of coordinate data and time respectively.

At this step, there is the option to add noise to the data using mean square error, in order to test the accuracy of predictions under different noise conditions. While ensemble and Weak SINDy are extensions that are already designed to handle noisy data, a moving average filter is applied to smooth out variations or noise in the dataset. The filter calculates the average of a fixed window of consecutive data points as the window "moves" or "slides" through the data. The coordinate data and the smoothed noisy coordinate data can be visualised using a plot function as shown in figure 10.

(a) Trajectory data of Frisbee flight with 1% noise added. No moving average filter applied.

(b) Trajectory data of Frisbee flight with 1% noise added. Moving average filter with window = 15 applied.

Figure 10: Example of imported data with added noise and smoothing function

Three datasets are saved as individual files: the original coordinates , the coordinates with noise added and time datasets. These can then be accessed by the fit and predict scripts.

5.2 Determining window size using an elastic moving window

For an unknown moving object, we have no information on the dynamics beforehand. This means that we do not know whether the object changes directions, which means that the dynamical equations governing its movement changes as well. For example, when a UAV changes its motion from takeoff to a circular motion, the dynamics of its motion changes.

This means that not all of the past trajectory data might be relevant for SINDy to fit the data, and that we should use 'recent' data for fitting the model. The elastic moving window has two aspects: (i) the window moves along with the data so only the last window length of data is used for fitting, and (ii) based on the accuracy of the prediction, the window will shrink or expand. When the accuracy decreases, the window will shorten so that the next model is fitted so more of older data is not used for fitting. When accuracy increases, the model will lengthen the window so that more data is used for fit the data.

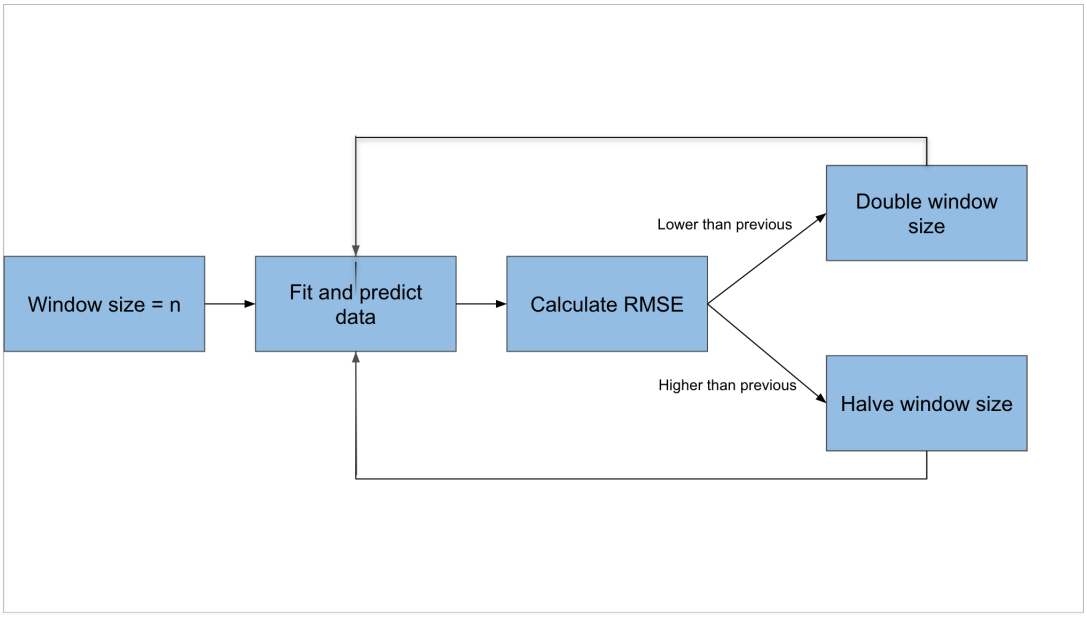


Figure 11: Diagram of the flow of window size algorithm.

As shown in figure 11, the model first starts with a default window size, which is set in my project to be the lower limit of the window size we want to take. For every prediction, only the window size of data is used to fit the model. We compare the accuracy of the predictions using the Root Mean Square Error (RMSE). If the RMSE decreases, it means that the current prediction is more accurate than the previous prediction. The window size will be doubled from the previous window size. If the RMSE increases, it means that the current prediction is less accurate than the previous prediction. The window will be halved from the previous window size. The window size is bounded by a minimum and maximum window size that is specified in the code and can be modified based on the time measurements of the data.

5.3 Fitting the data

This projects uses PySINDy (34; 35), which is a sparse regression package that has several implementations of SINDy, including standard SINDy, E-SINDy, and Weak SINDy.

During this step, we first define the weak PDE library and use that to select the sparsification hyperparameter using the model selection algorithm detailed in below in a separate section. The hyperparameter is passed into the SR3 optimizer. The model is then fitted once using the SR3 optimizer and weak PDE library.

In standard SINDy and Ensemble SINDy, the model only needs to be fitted once. However, this is not the case with Weak SINDy. A problem raised in the example for weak SINDy is that the model.predict functionality of the scikit-learn package used in PySINDy returns the prediction of the weak form of $\dot{x}$, not the original form of $\dot{x}$. Using the method described in the Weak SINDy example in PySINDy (34; 35), the workaround is to input the model coefficients from the weak form into the original(not weak) model.

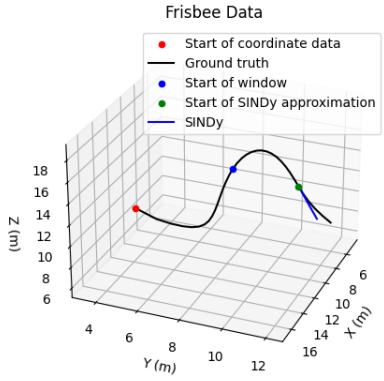
The model is fitted using a library of the non-weak candidate functions, and inputs the model coefficients from the weak form for fitting of the original model. This gives me the fitted weak SINDy model.

In this project, the bragging version of ensemble SINDy is used, which finds the median coefficients over the models (in the default setting, 20 models are fitted). In PySINDy, the simulate and predict functions use the coefficients of the last model that was fitted by default, so the median coefficients of the ensemble models must be calculated manually, and passed into the optimizer accordingly. This ensures that we are using the median model to simulate the trajectory, not the last fitted model. The median fitted model and an array of the ensemble coefficients are then saved as pickle files to be used in the predict step.

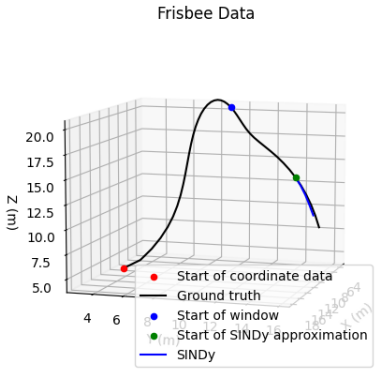
5.4 Prediction

In this step, the indexes of the window of data and start time of the prediction (which would be the last index of the window of data passed in) are found through a search of the time data. The indexes are then used to select the time array model will predict.

The prediction is simulated and plotted on a graph like in figure 12 showing the trajectory window passed into the model for fitting, the prediction, and the actual path of the object over the prediction time. There is an option to simulate the ensemble models and plot it on the graph as well.



(a) Despite the window being when the Frisbee changes from its initial upward trajectory and arcs downwards, weak SINDy is still able to predict the downward flight path



(b) In areas of the data where there is no huge shift in trajectory, the prediction is closer to the actual flight path.

Figure 12: Example of weak SINDy prediction on Frisbee. Blue to Green dot is the window of trajectory data used for fitting. Blue line shows weak SINDy prediction while black line shows the actual flight path of the Frisbee.

5.5 Features implemented

There are three key features implemented in my model, which will each be covered in individual sections:

1. Model Selection: Choosing sparsification parameter that balances accuracy and model complexity
2. Elastic moving window: Sliding window that moves along with the data, so that model fitting uses the most recent data. The window size decreases if prediction accuracy drops or increases if accuracy goes up.
3. Ensemble prediction: Using ensembling, multiple dynamical models are fitted to the window of data. All of the ensemble models used to predict multiple possible trajectories for the object. By default, only the median of the models is used as the prediction, but I have made available an option to show all the ensemble predictions.

6 Experimental Evaluation

6.1 Model Selection

A important aspect of identifying the dynamical equation using a sparsity promoting method such as SINDy is in choosing a sparsification hyperparameter that appropriately balances fit with model sparsity. If the hyperparameter is too large, important terms will be truncated off the model, and the error in modelling the trajectory will be high. If the hyperparameter is too small, the equation is not adequately sparsified, leading to the data exhibiting behaviour similar to overfitting.

Using a default threshold parameter is not sufficient, as different types of dynamical systems require different threshold values to identify the parsimonious model. Thus, we need a model selection algorithm is which can identify the parsimonious models, where the error is minimised by using the minimal number of terms to fit the data.

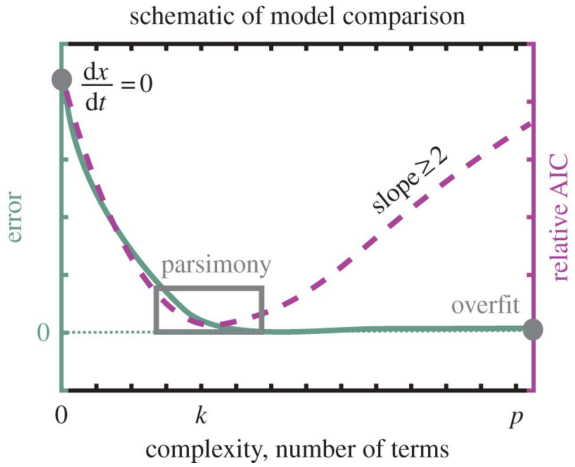


Figure 13: Diagram comparing Pareto analysis(green solid line) with AIC(dotted magenta line). Grey box shows the region of parsimony. Adapted from (36)

In the original SINDy papers (12) and other literature (37; 38), Pareto analysis is used to balance accuracy and model complexity. Pareto analysis is used to identify the "elbow" in the curve of accuracy vs complexity, called the Pareto front. However, it is not straightforward to interpret the Pareto analysis as it may not have a sharp elbow as shown in 13. Instead, there may be a region of parsimony - a range of values of the sparsity threshold where the root mean square error is the same (36). This means that Pareto analysis does not always provide a clear answer to the value of the optimal sparsification hyperparameter.

Since the model needs to handle trajectory data that has noise, parameter selection is important to ensure that we do not overfit to the data. Inspired by (39), I balance accuracy and model complexity by selecting parameters which minimize the Akaike information criterion (AIC) metric(36; 23).

The parameter selection comprises the hyperparameter threshold λ and nu parameter ν for the SR3 optimizer. The threshold determines the strength of regularization, while the nu determines the level of relaxation. Decreasing nu encourages w and v in equation 2 to be close, whereas increasing nu allows the regularized coefficients v to be farther from w.

The corrected AIC is given by

$$AIC_c = m \ln(RSS/m) + 2k + \frac{2(k+1)(k+2)}{m-k-2} \tag{9}$$

where RSS is the residual sum of squared errors.

$m = n \times d$ is the total number of measurements, k is the sparsity of the solution, while the last term is the correction for finite samples.

The AIC is used in the fitting and tuning the SR3 optimizer during the fitting step. The model selection procedure is as follows:

RMSE comparison		
Types of trajectory data	Weak SINDy with default hyperparameter	Weak SINDy with model selection
Projectile motion 0% noise	2.39	2.39
Projectile motion 2% noise	10.62	10.62
Frisbee 0% noise	0.46	0.46
Frisbee 2% noise	1.01	**0.67**
UAV takeoff	0.02*	**0.06**
UAV takeoff and circle	0.03*	**0.04**

Table 1: RMSE comparison of Weak SINDy models with and without model selection. The numbers with an asterisk '*' indicate cases where the model was unable to fit on one or more occasions, and returned models with no terms.

1. The window of trajectory data being used for fitting the model is split into a training and validation set. Here it is split in a 1:1 ratio, as the pysindy code requires the length of the training and validation dataset to be the same in order to fit the data properly.

2. The range of hyperparameters to be tested, training and validation set is passed into the SR3 tuning method.

3. The SR3 tuning method iterates through the different combination of threshold and ν to find the pairing with the minimum AIC and returns the pairing to be used in fitting step.

The user defines the range of hyperparameters in commons (or they can use the pre-set range provided) by providing a minimum and maximum hyperparameter, and the number of threshold values to be tested(which will define the interval between hyperparameters). For example, if you set the minimum to be 0.0 and the maximum to be 0.1, with number of threshold values 11, the values tested will be 0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10. A preset list of ν is included in the function as well as a default option, but the user can choose to specify it as well.

In table 1, no moving average filter is applied to smooth the noise in the data, in order to test each model's effectiveness in identifying parsimonious models with noiseless and noisy data. For the addition of noise, the random number generator was seeded to ensure that the results are reproducible. To produce the table above, the RMSE of each prediction is calculated, and the average of all the predictions for that particular trajectory data is taken as the RMSE.

The Weak SINDy with no model selection was unable to fit the model on more than one occasion, thus the lower RMSE returned is not indicative of a better performance as it fails to identify the equation and thus does not give a prediction of the object's trajectory.

On the Frisbee data, the RMSE of model selection is lower compared to without model selection in the presence of noise. When there is no noise, performance accuracy remains the same.

For the Gazebo data of different UAV motions, the weak SINDy with model selection was able to both fit the model and produce a low RMSE. Thus, model selection preserves the accuracy of the prediction by selecting the parsimonious model, and also adds robustness to the model, making it more able to adapt to different types of data and still make predictions.

6.2 Moving elastic window

Algorithm 2 Elastic window

```
   MIN_WINDOW_SIZE
2: MAX_WINDOW_SIZE
   rmse_score = 0                                              ▷ Current RMSE score
4: window_size ← MIN_WINDOW_SIZE                               ▷ Current window size
   rmse_score_new ← RMSE of prediction                         ▷ New RMSE score
6: if rmse_score_new > rmse_score and window_size > MIN_WINDOW_SIZE then
       window_size ← 0.75 × window_size
8:     rmse_score ← rmse_score_new
   else if rmse_score_new < rmse_score and window_size < MAX_WINDOW_SIZE then
10:    window_size ← 1.5 × window_size
       rmse_score ← rmse_score_new
12: else
                                                   ▷ Do not change if at max or min
14:    rmse_score ← rmse_score_new
   end if
```

An unknown moving object can make a sudden change in its movements, meaning the underlying dynamical system changes. When that happens, the model would fit data from both the old and new movements, and it would not be able to predict the future trajectory well.

This project presents a solution for that problem by only taking the most 'recent' data using a moving window. This way, the data from the older dynamical system is shifted out over time, and the model will fit the new motion.

Additionally, we can further adjust to when that sudden change in movement happens by making the window elastic. When a change in movement happens, the accuracy of the prediction will naturally decrease since the model has not adjusted yet. By shortening the window when the prediction accuracy drops, less or none of the data from the old dynamical system will be in the window. The next fit and prediction will then be to the new dynamical system. When the prediction accuracy of the model rises again, the window can be increased so that SINDy has more data to better fit the model.

In this project, the elastic window is bounded within a range which can be specified by the user. Starting with the lower bound, the window increases in length by 50% if RMSE decreases from one prediction to another. The window decreases by 25% of its current length if RMSE increases from one prediction to another. The exact percentage to increase or decrease the window length was decided based on testing various pairings to determine the one that best balances adaptability to change and accuracy. There is further scope for experimentation to determine a robust numerical approach to deciding how much to change the window.

In this work, the method implemented is to let the length window increases or decreases be changed by the user. They can choose to have a bigger change in window size if they want the model to be more adaptive to change. However, there will be a trade-off in accuracy.

Similar to table 1, no moving average filter is applied to smooth the noise in the data. The random number generator is seeded.

As expected, there are instances where the model cannot fit the data when there is no window for the Gazebo data. In the UAV takeoff and circle data, the motion of the UAV changes from an upward takeoff trajectory to a circular motion. Without a window, the data cannot fit the data when it has changes in motion, thus it returns a model with no terms and it does not make a prediction. The no window approach does better when there is no change in the dynamics in a noiseless data, such as the projectile motion 0% noise. This can be expected for data that follows a simple dynamical motion, as more data allows SINDy to better fit the data.

The fixed length moving window is able to adapt to the changes in dynamics, resulting in the model being able to fit the data. When there are changes in motion such as the UAV takeoff and circle data, the moving window will eventually not contain the data from the takeoff and only the circular motion, ensuring that the data can be fitted with. However, the accuracy is highly dependent on the type of data and the length of the moving window. If the window is large, the model will be less adaptable to sudden changes in motion. If the window is small, accuracy can be affected as there is less data to fit too, raising the possibility of the model behaving similarly to overfitting. Therefore, with a fixed moving window, the choice of the length of window is important and dependent on the user.

RMSE comparison between window types			
Types of trajectory data	No window	Fixed length moving window	Elastic moving window
Projectile motion 0% noise	**2.00**	2.39	2.39
Projectile motion 2% noise	24.4	12.63	**10.62**
Frisbee 0% noise	0.74	0.52	**0.46**
Frisbee 2% noise	1.06	1.20	**0.67**
UAV takeoff	0.01*	0.19	**0.06**
UAV takeoff and circle	0.02*	0.05	**0.02**

Table 2: RMSE comparison of Weak SINDy models with different types of windows. The numbers with an asterisk '*' indicate cases where the model was unable to fit on one or more occasions and returned models with no terms.

The elastic moving window performs better or as well as the fixed moving window across most of the data. Furthermore, it is able to fit the different types of data.

6.3 Ensemble SINDy prediction

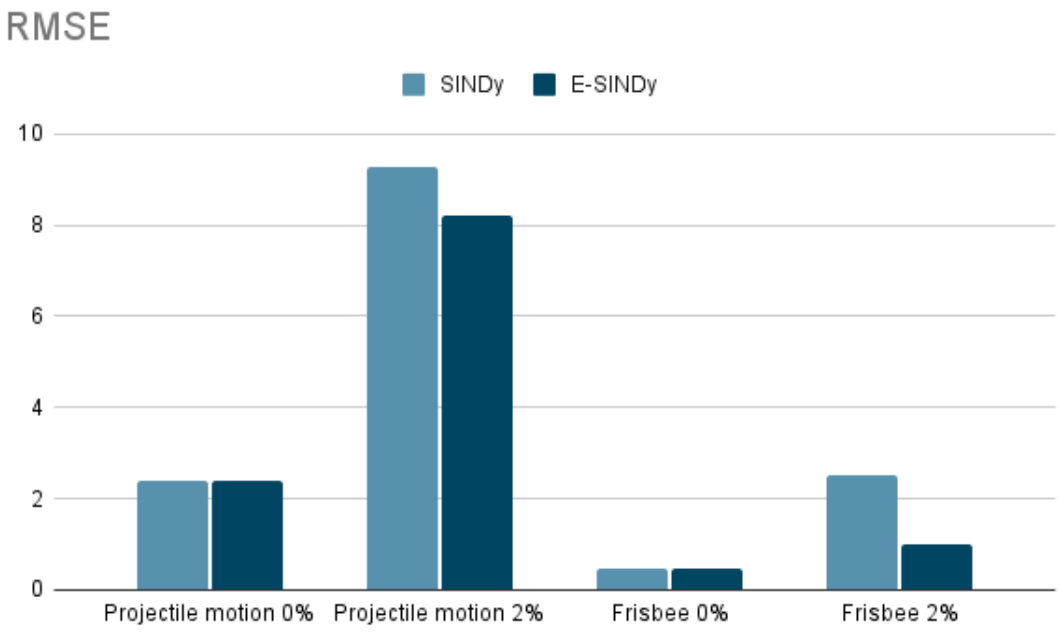
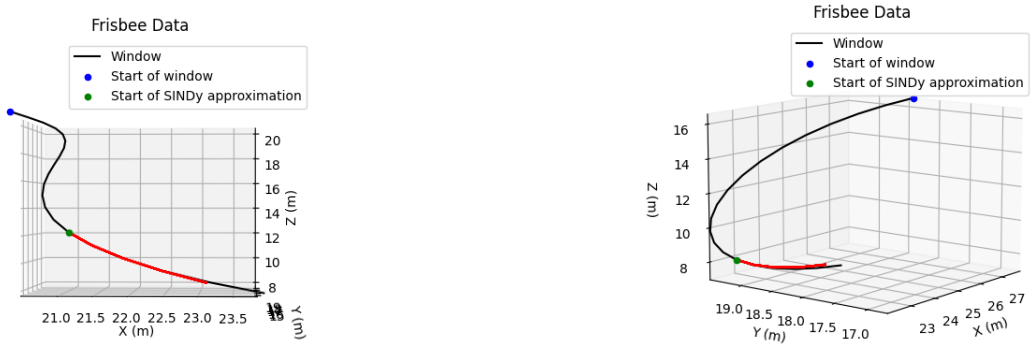


Figure 14: Column chart comparing the RMSE of standard SINDy to E-SINDy. E-SINDy has lower RMSE when the data is noisy, and the same RMSE when data has no noise.

According to (13), Ensemble SINDy demonstrates substantial improvements in accuracy when the data is noisy. This is backed up by the results in figure 14, where E-SINDy performs better than standard SINDy when using noisy data. Using ensembling in this project improves performance when dealing with noisy data.

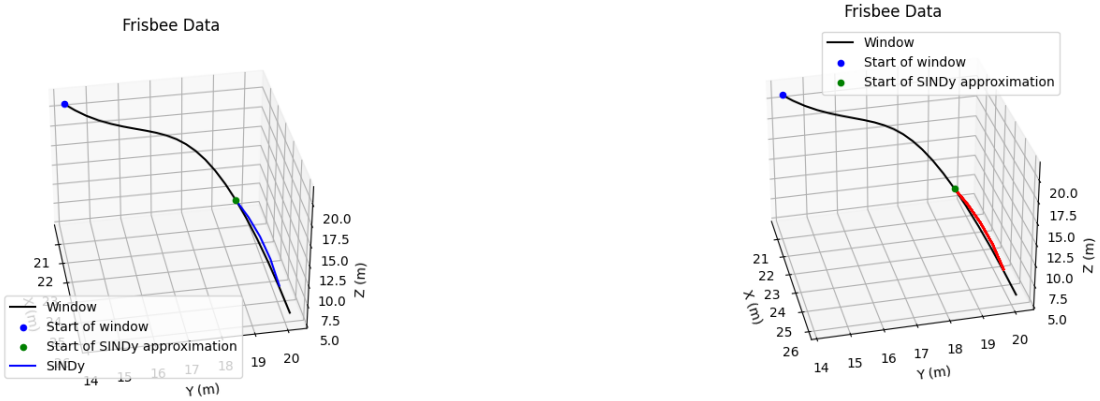
When using ensemble weak SINDy, the ensemble models are saved as well during the fitting step. In the prediction, we can simulate using all of the ensemble models to give us a bigger range of predictions. In the case of a well defined dynamics such as a Frisbee where there is no noise, the added range of prediction will be quite small



(a)

Figure 15: Predicting the flight trajectory of Frisbee using all the ensemble models instead of just the median model. The red zigzag line is the prediction of all the ensemble models (which is 20 in the default setting). Since the ensemble models are similar to one another, they end up being plotted quite close together to form a zigzag line

as seen in figure 15 as the ensemble models will be similar to one another, so the predictions will be similar as well. Also, there are only 20 ensemble models above, hence its limited range.



(a)

Figure 16: Comparing the accuracy of prediction of the flight trajectory of Frisbee using all the ensemble models instead of just the median model at 3% noise . The blue line is the prediction using only the median model and the red line shows the prediction using the ensemble models. The red line is closer to the actual trajectory line

When there is noise in the data, the ensemble range can provide a slight improvement over just taking the median as seen in figure 16. However, since the ensemble models are based on the same fitted data, the fitted models end up being quite similar to one another. This results in range of prediction being limited.

RMSE comparison between SINDy models			
Types of trajectory data	Standard SINDy	E-SINDy	Weak SINDy
Projectile motion 0% noise	2.39	2.39	2.39
Projectile motion 2% noise	9.29	8.20	10.62
Frisbee 0% noise	0.46	0.46	0.46
Frisbee 2% noise	2.53	1.01	**0.67**
UAV takeoff	0.004*	0.004*	**0.06**
UAV takeoff and circle	0.02*	0.02*	**0.02**

Table 3: RMSE comparison of SINDy models. The numbers with an asterisk '*' indicate cases where the model was unable to fit on one or more occasions and returned models with no terms.

6.4 Prediction accuracy

Table 3 proves that Weak SINDy is both more robust and has a higher accuracy when it comes to predicting different trajectories. Weak SINDy is the only one of the three SINDy methods tested that could fit the model for a UAV dataset generated using Gazebo. The table also shows how the approach developed over this work is able to handle different types of moving objects and data.

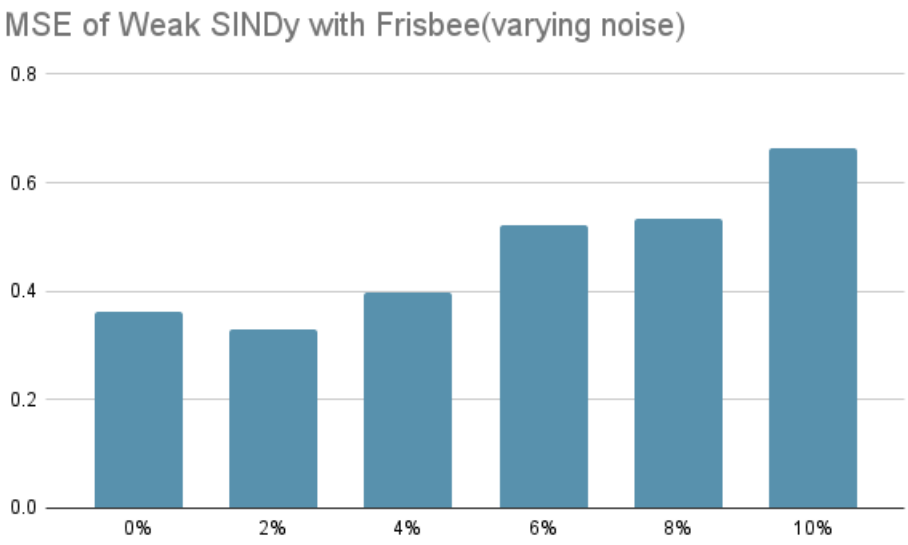


Figure 17: Column chart comparing the MSE of weak SINDy model with varying noise

Figure 17 shows that the accuracy of the prediction drops as the noise level in the data increases, which is the expected behaviour. While Weak SINDy is able to make predictions in higher noise levels, the noise level in real data could be higher than what Weak SINDy can handle. In that scenario, no model will be fitted and no prediction is made. This is a potential problem, especially if this work is used in a system where safety is critical and dependent on there always being an accurate prediction.

7 Conclusion

7.1 Limitations

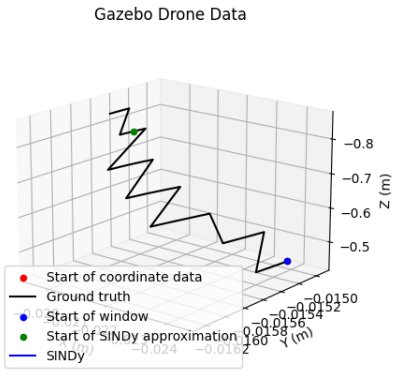
The work presented has certain assumptions that will prove to be limitations depending on the type of moving object it is modelling.

Firstly, this work assumes that the data is sufficient to fit the data, and that there are no multiple changes in dynamics within one window of data. The work would not fit a model if the data size is small and the direction the object is moving in changes very quickly, as illustrated in figure 18. In both scenarios in the figure, the takeoff motion of the UAV involves it being unstable and changing directions as it moves upwards. Ideally, the window would be able to shorten to handle this behaviour, but this happens over a short time frame of 0.5s, which is 17 data points for this dataset. If the window size decreases, the work would either have insufficient data to fit the model, or it will exhibit behaviour similar to overfitting to the data. Small data size and quick direction changes is more likely of high fidelity simulated data and real data.

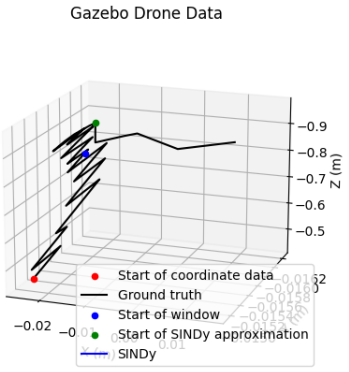
Next, the upper and lower bound of the window size needs to be specified, representing the upper and lower limits of how much time of data should the model take to fit the data. The range of hyperparameters to be tested also needs to be specified in the model. While it is possible to give a wide range and test more hyperparameters, this will have an increasing impact on complexity, as there are two for loops over the range of thresholds meaning complexity is $\mathcal{O}(n^2)$.

Moreover, motion data of real moving objects might have missing data (for example, x and y data present but z data is missing) or time steps which are not constant. The current approach takes into account that real data might not be in strictly increasing order of time due to errors in recording or other factors. It removes any data where time does not increase, as SINDy is not able to fit the data when the data given to it is not strictly increasing in terms of time. However, as of current implementation, there is no method to handle missing data.

Finally, the RMSE only takes into account the space component of error, not the time component. This means that a prediction that is completely straight but parallel to the actual trajectory, and one that oscillates around the actual trajectory could give the same RMSE. Hence, the RMSE does not give an idea on the type of error; it is only useful for comparing the accuracy between models.



(a) Start of Gazebo data, where UAV is taking off



(b) Next cycle of Gazebo data, where UAV is taking off

(c)

Figure 18: When there is limited data and noisy, weak SINDy model is still unable to fit the data and produce a prediction.

7.2 Discussion

This work shows that weak SINDy can accurately simulate future motion using only past coordinate data, with no prior knowledge of the environment or object dynamics. The proposed implementation using weak SINDy is able to tolerate noise and limited data size to make accurate predictions about the trajectory of the moving object.

We have also evaluated and shown how having variable data size depending on error metrics can improve prediction accuracy. Changes in motion are similar to data drift behaviour in time series forecasting problems. Our proposed implementation of a simple elastic moving window is effective in adapting to such behaviour, ensuring the robustness of the simulation while balancing accuracy. We show that the proposed implementation works on both (a) data with well-defined dynamics like projectile motion and Frisbee, and (b) high fidelity datasets where the motion changes.

One exciting future extension to this work would be to combine this project's prediction capabilities with MPC strategies, to produce a moving obstacle avoidance approach like the one using the SSA method(10), except that instead of SSA we utilise Weak SINDy. Another future extension would be for the Weak SINDy to increase the number of candidate functions or entirely different candidate libraries for fitting the data when the model accuracy is low, in order to adapt to non-linear dynamics better.

With regards to the elastic window, a possible future extension would be for each prediction, to calculate the accuracy of multiple window sizes and use that to update the window size for the next prediction. This could allow the prediction to be more robust when there is a change in object motion, but might be more computationally expensive as multiple simulations would need to be run for each step. Lastly, Fourier analysis or other related methods could be utilised for detecting data drift in the data, helping the model to decide when to shorten the elastic moving window. The current work only accounts for the space component when deciding on the elasticity of the window, but detecting data drift would consider the time component in order to create a more adaptive model.

References

[1] Jean-Claude Latombe. *Robot Motion Planning*. Springer New York, NY, Boston, MA, USA, 31 August 1991. ISBN 978-0-7923-9206-4. doi: https://doi.org/10.1007/978-1-4615-4022-9.

[2] John Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-03136-1.

[3] Lozano-Perez Tomas. Spatial planning: A configuration space approach — springerlink. 1990. doi: 10.1007/978-1-4613-8997-2_20. URL https://link.springer.com/chapter/10.1007/978-1-4613-8997-2_20.

[4] Paolo Fiorini and Z. Shiller. Robot motion planning among moving obstacles. 1995.

[5] Steven M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[6] M.G. Mohanan and Ambuja Salgoankar. A survey of robotic motion planning in dynamic environments - sciencedirect. 100, February 2018. URL https://www.sciencedirect.com/science/article/pii/S0921889017300313#b4.

[7] John Canny. *The complexity of robot motion planning*. MIT press, 1988.

[8] S. H. Tang, C. K. Ang, D. Nakhaeinia, B. Karasfi, and O. Motlagh. A reactive collision avoidance approach for mobile robot in dynamic environments. *Journal of Automation and Control Engineering*, 1(1):16–20, 2013. doi: 10.12720/joace.1.1.16-20.

[9] Jiangmin Chunyu, Zhihua Qu, E. Pollak, and M. Falash. A new reactive target-tracking control with obstacle avoidance in a dynamic environment. In - *2009 American Control Conference*, pages 3872–3877, 2009. ISBN 2378-5861. doi: 10.1109/ACC.2009.5160362. ID: 1.

[10] S. X. Wei, A. Dixit, S. Tomar, and J. W. Burdick. Moving obstacle avoidance: A data-driven risk-aware approach, 2023. ID: 1.

[11] Anushri Dixit, Lars Lindemanna, Skylar Wei, Matthew Cleaveland, George J. Pappas, and Joel W. Burdick. Adaptive conformal prediction for motion planning among dynamic agents. 1 Dec 2022. doi: https://doi.org/10.48550/arXiv.2212.00278.

[12] Steven L. Brunton, Joshua L. Proctor, and J. N. 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. URL https://doi.org/10.1073/pnas.1517384113. doi: 10.1073/pnas.1517384113; 31.

[13] U. Fasel, J. N. Kutz, B. W. Brunton, and S. L. Brunton. Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control — vol. 478, no. 2260. royal society. 478, 13 April 2022. doi: https://doi.org/10.1098/rspa.2021.0904. URL https://royalsocietypublishing.org/doi/10.1098/rspa.2021.0904.

[14] Daniel A. Messenger and David M. Bortz. Weak sindy: Galerkin-based data-driven model selection. *Multiscale Modeling and Simulation*, 19(3):1474–1497, 2021. doi: 10.1137/20M1343166. URL https://doi.org/10.1137/20M1343166. doi: 10.1137/20M1343166; 08.

[15] E. Kaiser, J. N. Kutz, and S. L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit — proceedings of the royal society a: Mathematical, physical and engineering sciences. *Proceedings of the royal society A*, 14 November 2018. doi: https://doi.org/10.1098/rspa.2018.0335. URL https://royalsocietypublishing.org/doi/full/10.1098/rspa.2018.0335.

[16] What is model predictive control? - matlab and simulink - mathworks united kingdom, 2023. URL https://uk.mathworks.com/help/mpc/gs/what-is-mpc.html.

[17] Brogan Caroline. Drones that patrol forests could monitor environmental and ecological changes — imperial news — imperial college london. URL https://www.imperial.ac.uk/news/207653/drones-that-patrol-forests-could-monitor/.

[18] J. Rothe, M. Strohmeier, and S. Montenegro. A concept for catching drones with a net carried by cooperative uavs. In - *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 126–132, 2019. ISBN 2374-3247. doi: 10.1109/SSRR.2019.8848973. ID: 1.

[19] Drago Bajc. Maximum ranges in ideal projectile motion—a generalization. *American Journal of Physics*, 58 (4):408–409, 1990. doi: 10.1119/1.16461. URL https://doi.org/10.1119/1.16461.

[20] S. K. Bose. Thoughts on projectile motion. *American Journal of Physics*, 53(2):175–176, 1985. doi: 10.1119/1.14106. URL https://doi.org/10.1119/1.14106.

[21] F. D. Medina. Looking at projectile motion from a different angle. *American Journal of Physics*, 46(12): 1282–1283, 1978. doi: 10.1119/1.11447. URL https://doi.org/10.1119/1.11447.

[22] Sarah Ann Hummel. *Frisbee flight simulation and throw biomechanics*. University of California, Davis, 2003.

[23] H. Akaike. A new look at the statistical model identification, 1974. ID: 1.

[24] P. Zheng, T. Askham, S. L. Brunton, J. N. Kutz, and A. Y. Aravkin. A unified framework for sparse relaxed regularized regression: Sr3, 2019. ID: 1.

[25] K. Champion, P. Zheng, A. Y. Aravkin, S. L. Brunton, and J. N. Kutz. A unified sparse optimization framework to learn parsimonious physics-informed models from data, 2020. ID: 1.

[26] Breiman Leo. Bagging predictors — springerlink. *Mach Learn*, 24:123–140, August 1996. doi: https://doi.org/10.1007/BF00058655. URL https://link.springer.com/article/10.1007/BF00058655#citeas.

[27] Peter Lukas Bühlmann. Bagging, subagging and bragging for improving some prediction algorithms. In *Research report/Seminar für Statistik, Eidgenössische Technische Hochschule (ETH)*, volume 113. Seminar für Statistik, Eidgenössische Technische Hochschule (ETH), Zürich, 2003.

[28] Nina Golyandina, Vladimir Nekrutkin, and Anatoly A. Zhigljavsky. *Analysis of time series structure: SSA and related techniques*. CRC press, 2001.

[29] Nina Golyandina and Anton Korobeynikov. Basic singular spectrum analysis and forecasting with r. *Computational Statistics and Data Analysis*, 71:934–954, 2014.

[30] Steven L. Brunton, Joshua L. Proctor, and J. N. 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. URL https://doi.org/10.1073/pnas.1517384113. doi: 10.1073/pnas.1517384113; 02.

[31] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[32] Daniel A. Messenger and David M. Bortz. Weak sindy for partial differential equations. *Journal of Computational Physics*, 443:110525, 2021. doi: 10.1016/j.jcp.2021.110525. URL https://www.sciencedirect.com/science/article/pii/S0021999121004204. ID: 272570.

[33] Rudy Rudy, Brunton Brunton, Proctor Proctor, and Kutz Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017. doi: 10.1126/sciadv.1602614. URL https://doi.org/10.1126/sciadv.1602614. doi: 10.1126/sciadv.1602614; 29.

[34] Brian M. de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton. Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):2104, /05/18 2020. doi: 10.21105/joss.02104. URL https://joss.theoj.org/papers/10.21105/joss.02104.

[35] Alan A. Kaptanoglu, Brian M. de Silva, Urban Fasel, Kadierdan Kaheman, Andy J. Goldschmidt, Jared Callaham, Charles B. Delahunt, Zachary G. Nicolaou, Kathleen Champion, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton. Pysindy: A comprehensive python package for robust sparse system identification. *Journal of Open Source Software*, 7(69):3994, /01/29 2022. doi: 10.21105/joss.03994. URL https://joss.theoj.org/papers/10.21105/joss.03994.

[36] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Model selection for dynamical systems via sparse regression and information criteria — proceedings of the royal society a: Mathematical, physical and engineering sciences. *Proc. R. Soc.*, 30 August 2017. doi: https://doi.org/10.1098/rspa.2017.0009. URL https://royalsocietypublishing.org/doi/10.1098/rspa.2017.0009.

[37] Wen-Xu Wang, Ying-Cheng Lai, and Celso Grebogi. Data based identification and prediction of nonlinear and complex dynamical systems. *Physics Reports*, 644:1–76, 2016. doi: 10.1016/j.physrep.2016.06.004. URL https://www.sciencedirect.com/science/article/pii/S037015731630134X. ID: 271542.

[38] Schmidt Schmidt and Lipson Lipson. Distilling free-form natural laws from experimental data. *Science*, 324 (5923):81–85, 2009. doi: 10.1126/science.1165893. URL https://doi.org/10.1126/science.1165893. doi: 10.1126/science.1165893; 03.

[39] Dimitris Bertsimas and Wes Gurnee. Learning sparse nonlinear dynamics via mixed-integer optimization. *Nonlinear Dynamics*, 111(7):6585–6604, 2023. doi: 10.1007/s11071-022-08178-9. URL https://doi.org/10.1007/s11071-022-08178-9. ID: Bertsimas2023.

8 User guide

8.1 Installation

To install and run the program, first clone the repository to your local computer.

Install the required packages from the relevant sites using your preferred installation method.

1. PySINDy
2. Dill
3. h5py
4. numpy
5. ipython
6. scikit-learn
7. matplotlib

Ensure that you are using python 3.8 and above.

8.2 Running the code

To run the ensemble Weak SINDy, navigate to the src folder and run in terminal:

```
python3 main.py
```

Or to run the projectile motion dataset instead:

```
python3 $main\_projectile\_motion.py$
```

Other versions of SINDy such as standard SINDy and E-SINDy can also be run by navigating to the corresponding folders and running the same commands as those above.