# A Robust Data-Driven Approach for Dynamics Model Identification in Trajectory Planning

Jiangqiu Chen[1,#] Minyu Zhang[1,#] Zhifei Yang[1,#] Linqing Xia[2,*]

***Abstract*—In this paper, we propose a data-driven modelling framework using a sparse regression technique to find the governing equations of dynamics systems. With this approach, the prior knowledge of features from simple structures can be used to deduce which on complex structures. The prior knowledge of single-pendulums, double-pendulums, and spherical pendulum enlightens the guess of the feature library for a 3-DOF manipulator. The feature library is sparsified with a fully autonomous machine learning algorithm composited of the L1-regularization and proportional filter. The training dataset with non-zero-mean Gaussian noise simulates real-world conditions and proves the system's robustness to the noise. Compared with the neural-network-based system identification method, this paper's technique can be promptly applied in dynamic trajectory planning. A simulation of the optimal trajectory planning for the obstacle-avoidance on the Lynxmotion robot is accomplished by optimizing the objective function constructed with energy and penalty function. Results of the simulation support that the estimated model works correctly. Comparison between the data-driven and the closed-form model evidences the reliability and robustness of this identification technique.**

***Keywords—dynamic system, system identification, machine learning, optimization, obstacle avoidance, trajectory generation***

## I. INTRODUCTION AND RELATED WORK

Since Newton's second law of motion, modeling a dynamic system has long been the basis of our analysis and control. For some simple system, like simple pendulum or mass spring damping system, the Newton-Euler iterative formula or the Lagrangian equation method plays the typical role to obtain dynamic models. However, for some advanced dynamic systems work in complicated scenario, those forward modeling methods becomes thorny or even impossible to achieve. For these more complex systems, we need to look for a more widespread or generalized approach for modelling.

With the advancement of machine learning and big data technology, the idea of obtaining models by measuring data is becoming a reality. Schmidt and Lipson [1] showed that it is possible to get the underlying structure of the dynamics model from the data. It uses the symbolic regression method, a regression method that does not require prior knowledge, and its search space is sparse. Therefore, if the underlying structure is complicated, the cost of realizing the model becomes significantly high.

In recent work, a data-driven sparse regression method was proposed to solve the difficulty of generating non-linear system dynamics from the measurement. Sparse identification of non-linear dynamics [2] (SINDy) emerged as an effective way to find the underlying structure of dynamics. To build a library of non-linear features, the cascading pivotal elemental functions could be established through the prior knowledge of the dynamic system. It shrinks the dimensionality of the library through sparse regression [3], whose results in an interpretable and parsimonious dynamics model. SINDy has some other variants, such as SINDy-PDE [4], SINDy-c [5], implicit-SINDy [6], PI-SINDy [7], abrupt-SINDy [8], modified-SINDy [9], and Lagrangian-SINDy [10]. They are extensions of the original SINDy, including implicit differential equations, partial differential equations, and differential equations from noisy data.

There are some other methods for dynamic system identification. These methods extract the governing equations from probabilistic machine learning(Gaussian process) [11][12], physics-informed neural network framework [13][14], and physics-informed generative adversarial networks [15].

Although the above methods have attained sparse modeling of some dynamics equations, these systems are usually work in a plane, such as double pendulums, the cart-pendulum. However, physical systems usually work in three dimensional spaces. After identifying dynamic systems, these models are usually used for trajectory planning and controlling. SINDy-c introduced external force and feedback to control the system after system identification. Nevertheless, position, velocity, and acceleration of these systems,are usually used to determine the objective function in planning and optimization tasks. For example, in the trajectory optimization for a manipulator, the minimum time or least energy trajectory is usually generated with the joint torque as the constraint. Lagrangian-SINDy [10] managed to identify the Lagrangian and the Hamiltonian equation, but the Lagrangian function cannot be directly used for practical tasks, although the Lagrangian function implicitly contains the force and torque information. Revealing the relationship between force/torque and kinematics parameters from the Lagrangian function is not a computationally simple task in real-time. Therefore, to facilitate planning and optimization issues, forces and torques need to be obtained directly. [30] provides a huge body of model learning in the MPC community for control tasks, it is approximated using a linear model. However, an accurate model is critical for the trajectory planning scenario. Model identification can be seen

# These authors contributed equally to this work.

1 Department of Engineering Mathematics, Faculty of Engineering, University of Bristol, Bristol, BS8 1QU, UK, {lh20406, mz17947, hv20465}@bristol.ac.uk.

2 Shanghai Electric Group Co., Ltd. Central Academe, Shanghai 200070, China, xialq2@shanghai-electric.com.

*Correspondence: Linqing Xia

as a compromise between complexity and accuracy. According to [31], the sparse nonlinear model sacrifices some speed but has better performance than linear models for MPC.

An actual manipulator is simulated to verify the data-driven model, and the optimised trajectory is solved based on the estimated torque equation. Saramago and Junior [16] use distance as a penalty function to generate obstacle avoidance trajectories. The genetic algorithm is applied by Sharman et al. [17] with a planar 3-DOF manipulator, which works outstandingly in energy optimization because the dynamics equation is simple.

Our work proposes a method inspired by SINDy to determine the torque equation directly for complicated systems. The main contributions of this work are:

**1)** Develop a general scheme inspired by SINDy to obtain the torque equation from the real-world environment measurement

**2)** Verify the robustness of the estimated model

**3)** Plan a trajectory for obstacle avoidance with minimal energy based on the global interior point method.

**4)** Test the reliability and robustness of the approach by comparing the data-driven and closed-form solution.

## II. METHODOLOGY

The flow chart in Fig.1 demonstrates how sparse model estimation method works.

### A. Noise Decomposition

The measurement of sensors regularly contain noises, which need to be decomposed to improve the precession. Many previous works offered various smoothing techniques, a novel approach is applied by estimating the noise signal as a latent variable relating the measurements and the real dynamics model. This method drives out the accurate noise signal from a small dataset as an individual signal rather than smoothing it. In this way, feature of the dynamic system can be retained to the maximum extent. We have the measurement $\boldsymbol{Y} = [\boldsymbol{y_1}, \cdots, \boldsymbol{y_n}]$, estimated noise $\boldsymbol{V} = [\boldsymbol{v_1}, \cdots, \boldsymbol{v_n}]$, estimated dynamics data $\boldsymbol{X} = [\boldsymbol{x_1}, \cdots, \boldsymbol{x_n}]$ where $\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{v} \in \mathbb{R}^{\boldsymbol{n}}$, and they are defined as,

$$\boldsymbol{y_i} = \boldsymbol{x_i} + \boldsymbol{v_i} \tag{1}$$

$$\boldsymbol{x_{i+1}} = \boldsymbol{f}(\boldsymbol{x_i}) = \boldsymbol{x_i} + \int_{t_i}^{t_{i+1}} \frac{\boldsymbol{d}}{\boldsymbol{dt}} \boldsymbol{x}(\boldsymbol{t}) \tag{2}$$

$$\boldsymbol{y_{i+1}} = \overbrace{\boldsymbol{f}\left(\underbrace{\boldsymbol{y_i} - \boldsymbol{v_i}}_{\boldsymbol{x_i}}\right)}^{\boldsymbol{x_{i+1}}} + \boldsymbol{v_{i+1}}. \tag{3}$$

So $\widehat{\boldsymbol{X}} = \boldsymbol{Y} - \widehat{\boldsymbol{V}}$, more detailed estimation approach can be founded in [19].

### B. Model-driven

The robotic manipulator's dynamics equation, used in this paper, is derived by a machine learning approach inspired by SINDY[2], which linearly regresses the features insides a customized library to fit the dynamics model. The general form is,

$$\dot{\boldsymbol{X}} = \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\Xi} + \boldsymbol{\eta}\boldsymbol{V}.$$

Where $\boldsymbol{\Xi} = [\boldsymbol{\zeta_1}, \cdots, \boldsymbol{\zeta_n}] \in \mathbb{R}^{\boldsymbol{n}}$ is an n-dimension weight coefficient corresponding to the features inside library $\boldsymbol{\Theta} \in \mathbb{R}^{\boldsymbol{m}\times\boldsymbol{n}}$ and indicates the correlation of feature terms to the model. $\boldsymbol{X} = [\boldsymbol{x_1}, \cdots, \boldsymbol{x_n}] \in \mathbb{R}^{\boldsymbol{n}}$ is the kinematic displacement of the model. $V$ is the noise, and $\boldsymbol{\eta}$ is the noise level; $V$ is removed after the noise decomposition. So, the dynamics equation is updated to,

$$\dot{\boldsymbol{X}} = \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\Xi}. \tag{4}$$

To handle the energy optimization on a manipulator, the approach in this work regress torque models from real-time kinematics angle-related feature $\boldsymbol{Q} = (\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \in \mathbb{R}$. The torque vector for actuating joint is represented by $\tau_{\mathrm{i}}$ can be derived by the Lagrangian equation as,

$$\boldsymbol{\tau_i}(\boldsymbol{t}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \frac{\boldsymbol{d}}{\boldsymbol{dt}}\frac{\boldsymbol{\partial L}}{\boldsymbol{\partial \dot{q}_i}} - \frac{\boldsymbol{\partial L}}{\boldsymbol{\partial q_i}}. \tag{5}$$

The Lagrangian equation is the linear combination of linear and non-linear functions of $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ which can be stated as,

$$\boldsymbol{L} = \boldsymbol{\Theta}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{\Xi}. \tag{6}$$

Based on equation (5), torque remains the form of the Lagrangian equation, and the related terms change as:

$$\frac{\boldsymbol{d}}{\boldsymbol{dt}}\frac{\boldsymbol{\partial L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\boldsymbol{\partial \dot{q}}} \rightarrow (\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \qquad \frac{\boldsymbol{\partial L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\boldsymbol{\partial q}} \rightarrow (\boldsymbol{q}, \dot{\boldsymbol{q}}).$$

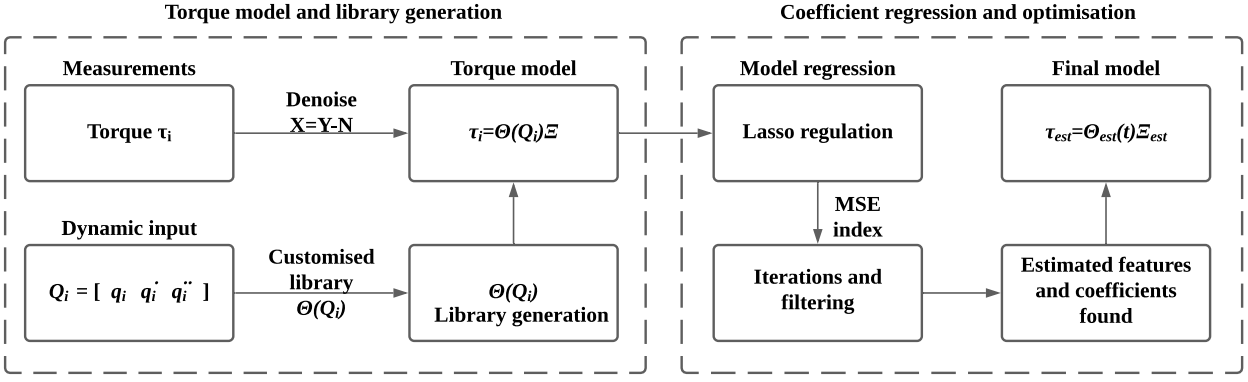


Fig.1 Block diagram showing the algorithm for sparse model estimation.

So, the torque model is then possible to be fitted for the $i^{th}$ joint based on $\boldsymbol{Q_i} = [\boldsymbol{q_i}, \dot{\boldsymbol{q}}_i, \ddot{\boldsymbol{q}}_i]$. Rearrange (4) to achieve the new relationship,

$$\boldsymbol{\tau_i(t)} = \boldsymbol{\Theta(Q_i)\Xi} \quad (6)$$

Measurements $\boldsymbol{\tau_i} \in \mathbb{R}^{\boldsymbol{m}}$ is the training data, also the input of this approach.

The library of linear and non-linear is constricted from previous knowledge, which means the library for complex system usually is a deduction from a simple structure. If a brand-new system structure is unknown, the library can be generated as the combination and permutation of non-linear monomial terms.

Many known optimization techniques can solve the coefficient $\boldsymbol{\Xi}$, and commonly regulation approaches include $Ridge, Lasso,$ L0 regulation [23, 24, 25], and the least-squares [26, 27], Bayesian optimization [28, 29] can verify the $\boldsymbol{\Xi}$ as well. If we aim to achieve a most sparse coefficients list, the Lasso regression is the solver for this algorithm. This could be written as a formal constrained equation,

$$\min_{\boldsymbol{\Xi}} \|\boldsymbol{\tau_i(t)} - \boldsymbol{\Theta}(\boldsymbol{q_i})\boldsymbol{\Xi}\|_2 + \boldsymbol{\alpha}\|\boldsymbol{\Xi}\|_1$$

$$\textbf{s.t. } \textbf{diag}(\boldsymbol{\Xi}) = \mathbf{0}.$$

The penalty coefficient $\alpha$ relating to the Lasso is updated with the iterations until the fitted model has the least error quantified by the MSE index compared to the estimated value $\hat{\tau}_i$ to the training data $\tau_i$.

$$\boldsymbol{\alpha} = \textbf{arg min } \boldsymbol{MSE}(\boldsymbol{\tau_i}, \hat{\boldsymbol{\tau}}_i)$$

### C. Feature selection

The Outputs of Lasso regression are sparse, which means some of the features has zero or equally tiny coefficients. A novel filter is introduced to eliminate all zero terms, and $\boldsymbol{\mathcal{N}}$ non-zero terms have the least contribution to the system with passing rate referred to as $r_p$.

$$\boldsymbol{\mathcal{N}} = \boldsymbol{(1 - r_p)n} \quad (7)$$

Where $n$ is the number of features that existed in the library, all features with zero coefficient and $\mathcal{N}$ left features with least coefficient are saved as a vector $\boldsymbol{RC} \in \backslash\mathbb{R}^{\boldsymbol{n}}$ and the coefficients of those features are saved as a vector $\boldsymbol{RF} \in \backslash\mathbb{R}^{\boldsymbol{n}}$. The new library is sparser and applied to the next regression.

This process sparsely reduces the number of features that have no contribution to the model fitting, so these eliminations do not affect the fitting accuracy. The sparsest feature library can be found until the filter starts to reduce the fitting precision, and the filtering algorithm can be written as,

$$\boldsymbol{MSE_{j+1} - MSE_j \geq 0} \Rightarrow$$

$$\boldsymbol{RF_j \notin \Theta_{j+1}, RC_j \notin \Xi_{j+1} \Rightarrow \Theta_{j+1}, \Xi_{j+1}}.$$

Where $\boldsymbol{j}$ is the number of iterations, and MSE is the Mean-Squared-Error between the estimation and the denoised data.

The output of the filter is the final estimation for the torque model described as,

$$\hat{\boldsymbol{\tau}}_{\boldsymbol{i}}\boldsymbol{(t)} = \widehat{\boldsymbol{\Theta}}\boldsymbol{(q_i)}\widehat{\boldsymbol{\Xi}}. \quad (8)$$

Where $\left(\hat{\boldsymbol{\tau}}_{\boldsymbol{i}}, \widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{\Xi}}\right)$ are the estimation results for the $\boldsymbol{i^{th}}$ joint of a manipulator by machine learning approach.

## III. Implementation

After identifying the dynamics relationships, a practical experiment (optimal trajectory planning) is implemented to verify that the model can be applied in practice. Optimization goal of the trajectory is based on energy minimization and collision free. The total energy can be computed by adding the product of the power at the discrete-time point and the time interval, and the power is the product of torque and the joint angular velocity $\dot{X}$. The obstacle is simplified as spheroidal. When the normal distance from the obstacle to the link is smaller than threshold $\lambda$, square of the difference between the distance and $\lambda$ is the penalty function. The closed-form torque formula is used as the baseline, and the trajectory is generated using the same method. By comparing the energy curve and total energy deviation of the two methods, we can verify whether the data-driven method achieves a feasible dynamics model.

Two complete pseudocodes for the model regression and dynamics trajectory planning are shown in Table. I and II.

### A. Model Statement

The 5-DOF Lynxmotion manipulator can be simplified to 3-DOF with revolute joints. Detailed geometric parameters and description can be found in [21]. The end-effector position in the simplified robot is derived from the Distal approach of the Denavit-Hartenberg convention.

$$\boldsymbol{P_3} = \begin{bmatrix} \boldsymbol{\cos q_1\,(l_3 \cos(q_2 + q_3) + l_2 \cos q_2)} \\ \boldsymbol{\sin q_1\,(l_3 \cos(q_2 + q_3) + l_2 \cos q_2)} \\ \boldsymbol{l_2 \sin q_2 + l_3 \sin(q_2 + q_3) + d_1} \end{bmatrix} \quad (9)$$

For $l_3 = 0$ and $q_3 = 0$, the position of the elbow can also be easily derived, and the equations are as follows:

$$\boldsymbol{P_2} = \begin{bmatrix} \boldsymbol{l_2 \cos q_1 \cos q_2} \\ \boldsymbol{l_2 \cos q_2 \sin q_1} \\ \boldsymbol{l_2 \sin q_2 + d_1} \end{bmatrix} \quad (10)$$

### B. Joint Space and Normalisation

Three quadratic polynomials related to time can formulate trajectory generated in joint space with $\boldsymbol{i} \in [1,3]$. The joint space angle and angular velocity at the initial and end positions can constrain the trajectory functions and reduce the polynomials' parameters.

$$\boldsymbol{q_i(t)} = \boldsymbol{a_{5(i-1)+1}\,t^4 + a_{5(i-1)+2}\,t^3 + a_{5(i-1)+3}\,t^2 + a_{5(i-1)+4}\,t + a_{5(i-1)+5}} \quad (11)$$

$$\boldsymbol{t_n} = \frac{\boldsymbol{t - t_{begin}}}{\boldsymbol{\sigma}}$$

$$\boldsymbol{\sigma = t_{end} - t_{begin}}$$

$$\boldsymbol{t = t_n \sigma + t_{begin}} \quad (12)$$

Moreover, the time in (11) can be normalized as $\boldsymbol{t_n} \in [0,1]$ to simplify the numerical energy integration from discrete power values and dynamics trajectory generation as in(12), $\boldsymbol{t}$ is the present moment, $\boldsymbol{t_{begin}}$ is the initial time, $\boldsymbol{t_{end}}$ is the final time. When the moving obstacle approaches the manipulator, it will dynamically re-plan the trajectory.

Therefore, the initial time, angle and angular velocity will also change in the new trajectory. The normalized time can unify the trajectory function format and simplify the calculation.

### C. Training Data Acquisition

The results of the analytical method will simulate the training data for this data-driven sparse regression method. Joint's required torque can be derived in terms of the Lagrangian equation for conservative energy. The velocity of the elbow and end effector can be achieved by derivative with respect to time. The Lagrangian equation described by L with respect to time, angle and angular velocity is equivalent to the difference between total kinetic energy and total potential energy in the following equation,

$$\boldsymbol{L(t,q,\dot{q})} = \frac{1}{2}\boldsymbol{m_2 v_2^2} + \frac{1}{2}\boldsymbol{m_3 v_3^2} + \frac{1}{2}\boldsymbol{I_1 \dot{q}_1{}^2} + \frac{1}{2}\boldsymbol{I_2 \dot{q}_2{}^2} + \frac{1}{2}\boldsymbol{I_3 \dot{q}_3{}^2} - \boldsymbol{m_2 g(P_{2z} - d_1)} - \boldsymbol{m_3 g(P_{3z} - d_1)}. \quad (13)$$

The robot dynamics equation in all the robot joints space can also be considered in the form of the physical description below,

$$\boldsymbol{\tau(t,q,\dot{q},\ddot{q})} = \boldsymbol{M(q)\,\ddot{q} + C(q,\dot{q})\dot{q} + G(q)}. \quad (14)$$

The total torque used as input to control the manipulator can be expressed in time, angle, angular velocity and angular acceleration. The symmetric matrix $M(q)$ describes inertia matrix; $C(q,\dot{q})$ with respect to angular velocity and angular acceleration defines centrifugal and Coriolis torque; The gravity force is represented by $G(q)$. Static friction, dynamic friction and viscous friction are ignored in this paper to simplify the simulation model.

### D. Measurement Error

In a compromise between model fit and calculation time, about 4000 data points train the model. Since it is impossible to avoid noise in real measurement, the training data is mixed with non-zero-mean Gaussian [19] distributed noise to improve the regression algorithm's robustness. The noise intensity is set between 0% and 10% because excessive noise will cause poor fitting. The actual measurement premises that the noise intensity is not extremely high, so excessive noise is not considered to verify the model identification algorithm.

### E. Library Generation based on prior knowledge

The Lagrangian function is a linear combination of non-linear functions $(\boldsymbol{X,\dot{X},\ddot{X}})$ related to kinematic parameters. Torque can be calculated by the Lagrangian function, so it can be inferred that it is also a linear combination of non-linear functions related to kinematic parameters. According to the work of SINDy-Lagrangian, it can be found that non-linear functions are composed of constant, trigonometric functions and polynomials as elementary functions, and the prior knowledge from simpler systems can inspire the identification of more complex systems. We derive the torque equations of the 1-DOF manipulator, the planar 2-DOF manipulator, and the spatial 2-DOF (the first and second shafts are perpendicular) manipulators. We can assume that the dynamics model of the 3-DOF manipulator has the following structure,

$$\boldsymbol{\Theta(X,\dot{X},\ddot{X})} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{X^{P_n}} & \boldsymbol{\dot{X}^{P_n}} & \boldsymbol{\ddot{X}^{P_n}} & \cdots \\ \boldsymbol{X^{P_n} sin\,(\sum_{i=1}^{n} P_n X_i)} & & \boldsymbol{X^{P_n} cos\,(\sum_{i=1}^{n} P_n X_i)} & & \end{bmatrix},$$

where $\boldsymbol{n \in Z^+}$ & $\boldsymbol{n < DOF}$.

Here, the higher polynomials are expressed as $X^{P_n}$, for example, $X^{P_2}$ denotes the quadratic nonlinearities of the state $X$, $\boldsymbol{X^{P_2}} = [\boldsymbol{X^2}_1 \quad \boldsymbol{X_1 X_2} \quad \cdots \quad \boldsymbol{X^2}_2 \quad \cdots \quad \boldsymbol{X^2}_n]$.

$\boldsymbol{sin\,(\sum_{i=1}^{n} P_n X_i)}$ denotes the nonlinearities of the sine function compound with polynomial functions, for example, $\boldsymbol{sin\,(\sum_{i=1}^{2} P_2 X_i)}$ is denoted as,

$$\boldsymbol{sin}\left(\sum_{i=1}^{2} \boldsymbol{P_2 X_i}\right) = \begin{bmatrix} \boldsymbol{sin\,(X_1)} & \boldsymbol{sin\,(X_2)} & \cdots \\ \boldsymbol{sin\,(2X_1 + X_2)} & \boldsymbol{sin\,(2X_1 + 2X_2)} & \end{bmatrix}.$$

The more categories of the library, the more likely it is to return to an accurate model but the slower the regression speed. Choose the size of the library according to the complexity of the applied system. In this work, the number of feature vectors for the initial library of 3-DOF manipulators is 75.

### F. Optimization of Hyperparameters for Model Selection

Foremost, specify that α is in a rough range, such as 10e-6~10e-4, with a step size of 10e-6. The model fitting is repeated for each discrete α value in the range.

The L1 regularization is performed every iteration to sparse the parameters, after that, the parameters are sorted, and the parameters with smaller absolute values are eliminated according to a certain proportion $fr$. MSE qualitative the fitting of the model, and the model with the smallest MSE is found by traversing the hyper-parameters within a specific range.

TABLE I. PSEUDOCODE OF TORQUE MODEL REGRESSION.

**Algorithm 1:** $\tau$ model regression

**Input:** Measurement($\tau$)
**Output:** Dynamics Model of $\tau$

**Function** `NoiseApproximation`($data, \lambda$)**:**
  $X\_smooth.T \times Noise\_matrix = data$;
  $Noise\_approxi = data - X\_smooth$;
  **return** $(Noise\_approxi, X\_smooth)$

**Preprocessing:**
  **i**.Library generation;
  **ii**.Training data generation;
  **iii**.Measurement noise generation;
  **iv**.Noise Approximation;

**repeat**
  L1 regulation;
  **for** $i = 1$ *to iterations* **do**
    $\mathcal{N}$= $(1 - fr)$·Coeff_Volume;
    **if** *Coeff*<*Coeff_sorted*[$\mathcal{N}$] **then**
      delete Coeff, Feature;
  Mean-Absolute-Percentage-error;
**until** *(best alpha found)*;
regulation by best alpha;

### G. Dynamic Avoidance with Energy Optimization

Before ascertaining total consumed energy as part of the objective function, absolute total power consumption $P(t_n)$ is derived by the sum of torques multiplied by angular velocity initially. It is hard to integrate a trigonometric function containing a polynomial inside, hence the numerical integration of power consumption is applied to approximate consumed total energy for actuating robot arm.

$$\boldsymbol{P(t_n)} = \left| \sum_{i=1}^{3} \boldsymbol{\tau_i \dot{q}_i} \right| \tag{15}$$

A vector relationship can demonstrate the distance from the elbow and the end effector to the moving obstacle in (16). $D_1$ represents the distance between the second link to the obstacle. The distance from the last link to the obstacle is denoted in $D_2$; $P_1$ and $P_{obs}$ are the distance of shoulder and obstacle, respectively.

$$\begin{cases} \boldsymbol{D_1(t_n)} = \left| \overrightarrow{\boldsymbol{P_1P_{obs}}} - \dfrac{\overrightarrow{\boldsymbol{P_1P_2}}}{\left|\overrightarrow{\boldsymbol{P_1P_2}}\right|} \overrightarrow{\boldsymbol{P_1P_{obs}}} \dfrac{\overrightarrow{\boldsymbol{P_1P_2}}}{\left|\overrightarrow{\boldsymbol{P_1P_2}}\right|} \right| \\ \boldsymbol{D_2(t_n)} = \left| \overrightarrow{\boldsymbol{P_2P_{obs}}} - \dfrac{\overrightarrow{\boldsymbol{P_2P_3}}}{\left|\overrightarrow{\boldsymbol{P_2P_3}}\right|} \overrightarrow{\boldsymbol{P_2P_{obs}}} \dfrac{\overrightarrow{\boldsymbol{P_2P_3}}}{\left|\overrightarrow{\boldsymbol{P_2P_3}}\right|} \right| \end{cases} \tag{16}$$

A penalty function $F(t_{n2})$ with respect to $t_{n2}$ is introduced to avoid crashing the obstacle in time. The objective function is the sum of approximate total energy and a penalty function.

$$\boldsymbol{F(t_n)} = \boldsymbol{\alpha} \begin{pmatrix} \boldsymbol{max}\{\boldsymbol{0}, -(\boldsymbol{D_1(t_n)} - \boldsymbol{\lambda})\}^{\boldsymbol{2}} + \\ \boldsymbol{max}\{\boldsymbol{0}, -(\boldsymbol{D_2(t_n)} - \boldsymbol{\lambda})\}^{\boldsymbol{2}} \end{pmatrix} \tag{17}$$

$$\begin{aligned} \boldsymbol{E} &= \sum_{i=1}^{N} |\boldsymbol{P(t_{n1})}| \frac{\boldsymbol{\sigma}}{\boldsymbol{N}} + \boldsymbol{F(t_{n2})} \\ &= \frac{\boldsymbol{\sigma}}{\boldsymbol{N}} \begin{pmatrix} \left|\boldsymbol{P}\left(\frac{\boldsymbol{1}}{\boldsymbol{N}}\right)\right| + \cdots + \left|\boldsymbol{P}\left(\frac{\boldsymbol{i}}{\boldsymbol{N}}\right)\right| + \\ \cdots + |\boldsymbol{P(1)}| \end{pmatrix} + \boldsymbol{F(t_{n2})} \end{aligned} \tag{18}$$

In the experiment, normalized time $t_{n1}$ should be divided into $N$ parts according to a fixed σ to calculate the discrete value of consumed power, while the sampling number of obstacle detection $K$ is more significant than $N$ in order to update the trajectory in real-time. When a moving obstacle is detected at the $k_{st}$ sampling point, $t_{n2}$ should be initialized into zero. For initialization, angle and angular velocity, σ and the remaining number of sampling at $k_{st}$ sampling point will be updated for the initial boundary condition in the remaining part of the trajectory.

The distance between the manipulator and the obstacle should be much larger than the length of the robot grasper; hence it can maintain a safe distance as threshold $\boldsymbol{\lambda}$ to avoid the moving obstacle. When the distance is larger than $\boldsymbol{\lambda}$, there is no penalty value. Otherwise, the penalty function will be activated to avoid obstacles. The objective function with the exterior penalty function will eventually converge when the penalty coefficient α increases in each iteration. Besides, the quadratic function can accelerate the speed of convergence.

The interior-point method (IPM)[20] can optimize the non-linear objective function because this non-linear optimization method is suitable for large-scale constrained problems. The geometric boundary non-linearly constrains the objective function at the start and end position. The parameters vector $\bar{a} \in \mathbb{R}^9$, of joint space function are optimized by this method, which can be applied to the torque equation to control the robot along the planned trajectory.

$$\bar{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}, \mathbf{q_{end}}} \boldsymbol{E}(\bar{\boldsymbol{a}}) \tag{19}$$

$$\boldsymbol{s.t.\ q(t_{end}) - q_{end} = 0}$$

TABLE II. PSEUDOCODE OF DYNAMICS TRAJECTORY GENERATION WITH MINIMUM ENERGY.

**Algorithm 2:** Dynamics trajectory generation with minimum energy

**Input:** Boundary Condition, Sample Volume, Obstacle Position
**Output:** Joint-Space parameters

**Function** `Objcetive function`$(BC, obstcale(x,y,z), \alpha, \lambda)$**:**
  $E(x) = \sum_{t_n=\frac{1}{N}}^{1} P(t_n) \cdot \delta t_n$;
  $p_i(x) = \alpha \cdot max(0, -(d_i - \lambda))^2$;
  $f(x) = E(x) + p_i(x)$;
  **return** $f(x)$
**Preprocessing:**
  **i**.Time normalization:
    $t_n = \frac{t - t_0}{t_f - t_0}, \sigma = t_f - t_0$;
  **ii**.Objective function;
Trajectory initialization;
**for** $i = 1$ *to sample volume* **do**
  **if** *(d1,d2)<Threshold* **then**
    Update (Boundary conditions, Sample volume);
    **for** *n=1 to iterations* **do**
      Optimization:$minf(x), s.t.ceq(x)$;
    $x \leftarrow min(f(x))$;
    $(NewPath,\ NewVelocity) \leftarrow x$;
    Path=Array[Path]+NewPath;
    Velocity=Array[Velocity]+NewVelocity;
Visualisation;

## IV. RESULTS AND DISCUSSION

### A. De-Noise

The denoise progress effectively wipes off the noise from the measurement after the non-mean-Gaussian noise is added to the theoretical training data. Fig.2 demonstrates one denoise example on the first joint at a 5% noise level. The standard deviation for the measurement data is 0.3603 N/m, and the standard deviation for the denoised data is 0.3599 N/m, which is the same as the theoretical data free from noise. The MSE between the no-noise data and the measurement is 0.000196 N/m. In comparison, the MSE between the non-noise and the denoised data is 0.000109. Those indices indicate that the de-noise progress has effectiveness on extracting the noise from the measurement.

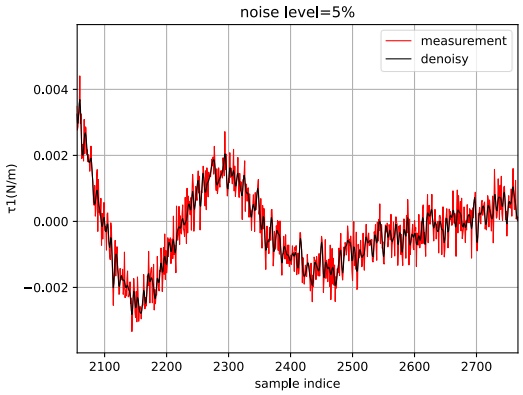


Fig.2 Denoise at 5% noise level for the first joint.

## B. *Model Estimation*

The best $\alpha$ parameters relating to the Lasso regulation for the three joints of the Lynxmotion robot are selected with iterations. The MAPE(Mean-Average-Percentage-Error) plot for $\alpha$ on each joint is shown correspondingly in Fig.3,4,5. The least error scenario achieved when the MAPE scores are minimal means the highest rate of the fitting. These points are also shown as the bottom point on the MAPE plot.

Subsequently, the estimated model is regressed at the best $\alpha$ values, which is selected from the $\alpha$ selection, 7.5e-7, 7e-5, 6e-5 corresponds to joint1 to 3. Afterwards, the fitting precisions at the selected $\alpha$ are demonstrated in Fig.6,7,8.

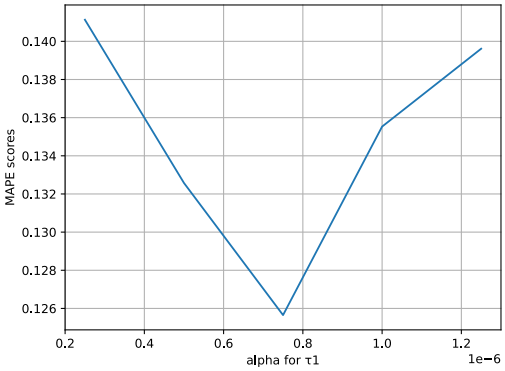


Fig.3 The minimum point on $\boldsymbol{\tau_1}$

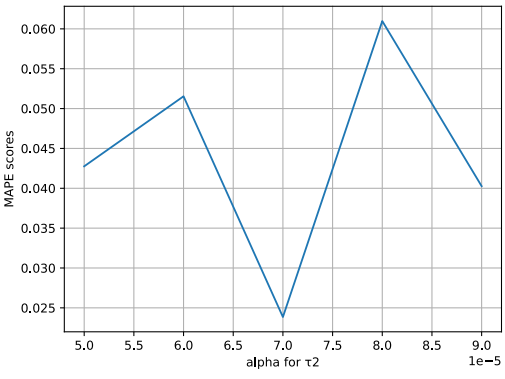


Fig.4 The minimum point on $\boldsymbol{\tau_2}$

*To ensure the accuracy of the predicted model, the torque data used for training is spliced by several pieces of measured values.

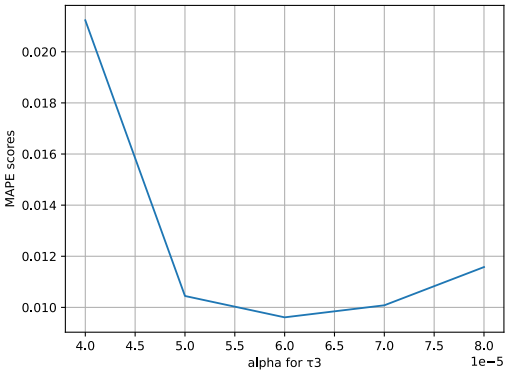


Fig.5 The minimum point on $\boldsymbol{\tau_3}$

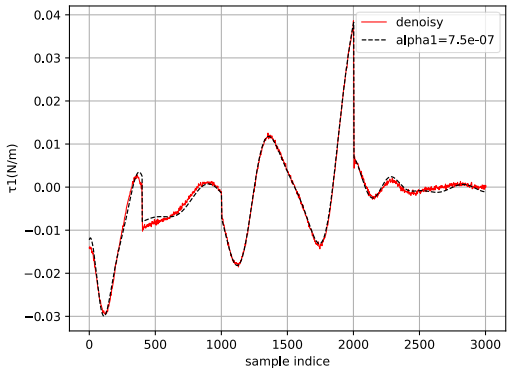


*Fig.6 Model fitting for $\boldsymbol{\tau_1}$.

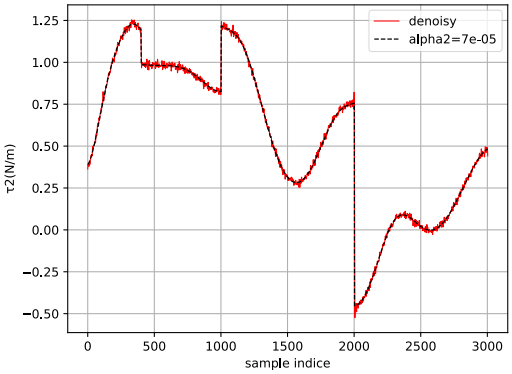


*Fig.7 Model fitting for $\boldsymbol{\tau_2}$.

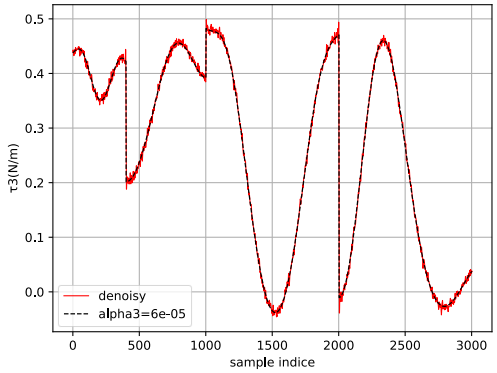


*Fig.8 Model fitting for $\boldsymbol{\tau_3}$.

The torque model for every joint has a perfect fitting precision proved by the MSE indices listed below. This

performance attests this estimated model is resisted to noise and maintains high reliability and robust in the noised environment. Potentially, this identification approach for an unknown model can be implemented further in investigation or control purposes in real-world conditions.

TABLE III. MSE VALIDATION

| Joint | MSE(N/m) |
|---|---|
| 1st | 0.0000000114 |
| 2nd | 0.0000031885 |
| 3rd | 0.0000010991 |

### C. *Robutness*

The robustness of this sparsification in model identification is proved to have good robustness in different physical model in this paper. This algorithm is verified with three independent models with different orders and libraries. The results are shown in the *TABLE IV*. For all the cases, the identified models almost restore the true models and the MSE indices between the real and predicted trajectory clue the goodness of fit.

### D. *Trajectory visualisation*

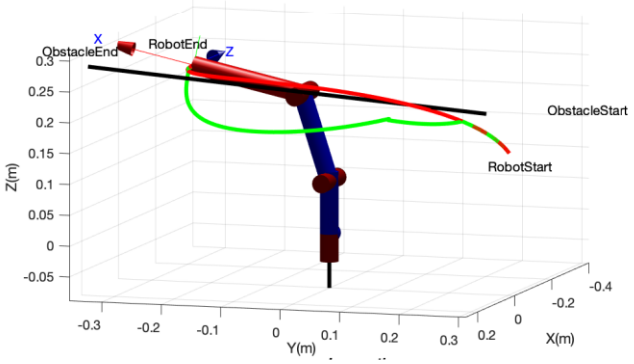


Fig.9 The visualisation of the estimated model for obstacle avoidance.

Fig.9 visualize the simulation of the Lynxmotion's trajectory controlled based on the predicted model for obstacle avoidance in a dynamic environment. The black line is the movement of the obstacle, which moves simultaneously with the robot. The start and the end point for the robot are preset; the beginning speed is desired, the red line is the energy-optimal trajectory without obstacles, while the green line updates the real-time trajectory to avoid crush onto the black obstacle.

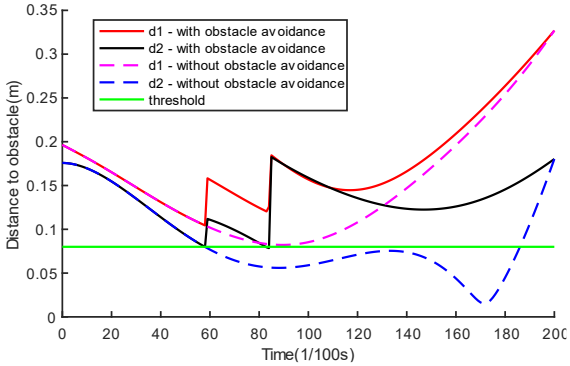


Fig.10 Distance from the links to the obstacle.

In Fig.10, the notation d1 and d2 are the distance to the second and third robot link, respectively. The penalty function (17) enables the manipulator to avoid the moving obstacle when the distance to the obstacle is less than the threshold. It is comprehensible to find when the avoidance function is on, the distance to the obstacle is always higher than the threshold, which means the penalty function and the dynamics model works with effectiveness.

The optimized power plot against time for the closed-form (analytical) and the model-estimation solution (Fig.11) shows the same global trend and has overall closed energy, whereas the latter consumes slightly more energy. The orange line represents the estimated model avoiding obstacle, and there is some bias on this line because the penalty function amplifies the position bias due to the estimation error. However, this error is not apparent when there are no avoidance requirements since the penalty function does not participate. These results proved from the energy aspect that the model estimation works expectedly and correctly.

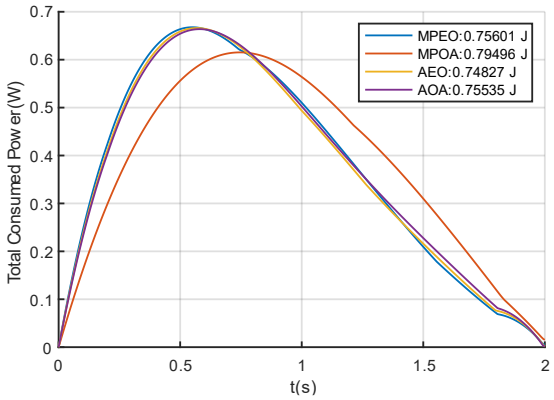


Fig.11 Comparison between model-estimation and closed-form solution.

## V. CONCLUSION AND FUTURE WORK

This paper states a data-driven approach for dynamics model identification* with the inspiration of SINDy, which learns a group of reality-simulated data (noise, moment of inertia included) to regress the estimated dynamics model. We use a three-dimensional manipulator to test the estimated model by planning an energy-optimal trajectory for moving-obstacle avoidance. Results verify that the estimated model works rarely similar to the expected performance, evidencing the approach's high fitting accuracy with good robustness to the noise.

The regression progress works expectedly when we have a fixed predict library since the simulated model is static and frictionless. For future work, selecting the initial feature library for a dynamic system is a challenging issue. A higher-order non-linear term of velocity and position exists in the real robot model when friction involved. The initial feature library must increase to contain higher-order terms. However, a large feature library requires a significant time consumption. Hence, how to select the initial library of order will be a potential work and a generalised feature library for dynamic systems also needs to be deliberate.

*https://github.com/minyuzhang98/Modell-identification-and-trajectory-planning

TABLE IV. DYNAMICS MODEL EXTRACTED FROM SIMULATION DATA AT 5% NOISE

| Model | True Model | Identified Model | Metrics |
|---|---|---|---|
| Mass Spring Damper | $\ddot{x} + 2\dot{x} + 100x$ | $\ddot{x} + 2.02\dot{x} + 100.67x$ | MSE:0.012 |
| Double pendulum | $3\ddot{\theta}_1 + 2\ddot{\theta}_2 \cos(\theta_1 - \theta_2) + 2\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + 29.4 \sin(\theta_1)$ | $2.99\ddot{\theta}_1 + 2.04\ddot{\theta}_2 \cos(\theta_1 - \theta_2) + 2.02\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + 29.47 \sin(\theta_1)$ | MSE:0.086 |
| | $2\ddot{\theta}_2 + \ddot{\theta}_1 \cos(\theta_1 - \theta_2) - \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + 9.8 \sin(\theta_2)$ | $2.03\ddot{\theta}_2 + 0.99\ddot{\theta}_1 \cos(\theta_1 - \theta_2) - 0.99\dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + 9.78 \sin(\theta_2)$ | MSE:0.028 |
| Spherical Pendulum | $2\ddot{\phi} \sin^2(\theta) + 4\dot{\phi}\dot{\theta} \cos(\theta) \sin(\theta)$ | $1.99\ddot{\phi} \sin^2(\theta) + 3.97\dot{\phi}\dot{\theta} \cos(\theta) \sin(\theta)$ | MSE:0.0002 |
| | $2\ddot{\theta} - 2\dot{\phi}^2 \sin(\theta) \cos(\theta) + 19.6 \sin(\theta)$ | $1.98\ddot{\theta} - 2.05\dot{\phi}^2 \sin(\theta) \cos(\theta) + 19.53 \sin(\theta)$ | MSE:0.0056 |


## ACKNOWLEDGEMENT

We would like to express our sincere gratitude for the support of Professor Chenguang Yang. We also appreciate the valuable discussion with Linqing Xia.


## REFERENCES


[1] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *science,* vol. 324, no. 5923, pp. 81-85, 2009.

[2] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of non-linear dynamical systems," *Proceedings of the national academy of sciences,* vol. 113, no. 15, pp. 3932-3937, 2016.

[3] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science Advances,* vol. 3, no. 4, p. e1602614, 2017.

[4] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological),* vol. 58, no. 1, pp. 267-288, 1996.

[5] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Sparse identification of non-linear dynamics with control (SINDYc)," *IFAC-PapersOnLine,* vol. 49, no. 18, pp. 710-715, 2016.

[6] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Inferring biological networks by sparse identification of non-linear dynamics," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications,* vol. 2, no. 1, pp. 52-63, 2016.

[7] K. Kaheman, J. N. Kutz, and S. L. Brunton, "SINDy-PI: a robust algorithm for parallel implicit sparse identification of non-linear dynamics," *Proceedings of the Royal Society A,* vol. 476, no. 2242, p. 20200279, 2020.

[8] M. Quade, M. Abel, J. Nathan Kutz, and S. L. Brunton, "Sparse identification of non-linear dynamics for rapid model recovery," *Chaos: An Interdisciplinary Journal of Nonlinear Science,* vol. 28, no. 6, p. 063116, 2018.

[9] K. Kaheman, S. L. Brunton, and J. N. Kutz, "Automatic differentiation to simultaneously identify non-linear dynamics and extract noise probability distributions from data," *arXiv preprint arXiv:2009.08810,* 2020.

[10] H. K. Chu and M. Hayashibe, "Discovering interpretable dynamics by sparsity promotion on energy and the lagrangian," *IEEE Robotics and Automation Letters,* vol. 5, no. 2, pp. 2154-2160, 2020.

[11] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Machine learning of linear differential equations using Gaussian processes," *Journal of Computational Physics,* vol. 348, pp. 683-693, 2017.

[12] M. Raissi and G. E. Karniadakis, "Hidden physics models: Machine learning of non-linear partial differential equations," *Journal of Computational Physics,* vol. 357, pp. 125-141, 2018.

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

[14] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *SIAM Review,* vol. 63, no. 1, pp. 208-228, 2021.

[15] L. Yang, D. Zhang, and G. E. Karniadakis, "Physics-informed generative adversarial networks for stochastic differential equations," *arXiv preprint arXiv:1811.02033,* 2018.

[16] S. F. Saramago and V. S. Junior, "Optimal trajectory planning of robot manipulators in the presence of moving obstacles," *Mechanism and Machine Theory,* vol. 35, no. 8, pp. 1079-1094, 2000.

[17] Sharma, Gouri Shankar, Mandeep Singh, and Tejinder Singh. "Optimisation of energy in robotic arm using genetic algorithm." International Journal of Computer Science and Technology 2.2 (2011): 315-317.

[18] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*. Springer, 2017.

[19] S. H. Rudy, J. N. Kutz, and S. L. Brunton, "Deep learning of dynamics and signal-noise decomposition with time-stepping constraints," *Journal of Computational Physics,* vol. 396, pp. 483-506, 2019.

[20] Y. Zhang, "Solving large-scale linear programs by interior-point methods under the MATLAB environment," *Optimisation Methods and Software,* vol. 10, no. 1, pp. 1-31, 1998.

[21] "AL5D Robot Arm Specifications," *Lynxmotion.com,* Online 2021. [Online]. Available: http://www.lynxmotion.com/driver.aspx?Topic=specs04.

[22] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research,* vol. 12, pp. 2825-2830, 2011.

[23] Hoerl, Arthur E., and Robert W. Kennard. "Ridge regression: applications to nonorthogonal problems." Technometrics 12.1 (1970): 69-82.

[24] R. Tibshirani, "Regression shrinkage and selection via the Lasso," Journal of the Royal Statistical Society: Series B (Methodological), vol. 58, no. 1, pp. 267–288, 1996.

[25] Huang, Jian, et al. "A constructive approach to l0 penalised regression." The Journal of Machine Learning Research 19.1 (2018): 403-439.

[26] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of non-linear dynamical systems," Proceedings of the national academy of sciences, vol. 113, no. 15, pp. 3932–3937, 2016.

[27] L. Zhang and H. Schaeffer, "On the convergence of the SINDy algorithm," Multiscale Modeling & Simulation, vol. 17, no. 3, pp. 948–972, 2019.

[28] S. Zhang and G. Lin, "Robust data-driven discovery of governing physical laws with error bars," Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 474, no. 2217, p. 20180305, 2018.

[29] W. Pan, Y. Yuan, J. Gonçalves, and G. Stan, "A sparse Bayesian approach to the identification of non-linear state-space systems," IEEE Transactions on Automatic Control, vol. 61, no. 1, pp. 182–187, January 2016.

[30] Hewing L, Wabersich KP, Menner M, Zeilinger MN. Learning-based model predictive control: Toward safe learning in control. Annual Review of Control, Robotics, and Autonomous Systems. 2020 May 3;3:269-96.

[31] Kaiser, Eurika, J. Nathan Kutz, and Steven L. Brunton. "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit." Proceedings of the Royal Society A 474.2219 (2018): 20180335.