Research Paper Reproduction: Sparse identification of nonlinear dynamics for model predictive control in the low-data limit

Kartikeya Mishra
M. Eng, Robotics
University of Maryland,
MD, USA
kmishra@umd.edu

Pratik Patel
M. Eng, Robotics
University of Maryland,
MD, USA
pratik94@umd.edu

***Index Terms*—Model predictive control, nonlinear dynamics, sparse identification of nonlinear dynamics (SINDY), system identification, control theory, machine learning**

***Abstract*—Using advances in data driven modelling, system models can be improved to great extent and Model Predictive Control can be implemented without much hurdles. In this paper author tries to overcome traditional machine learning models' drawbacks such as large data requirements, lack of generalizability, interpretable output models by using sparse identification of the system and applying that to the system with control. Author shows the method's usability to model system for MPC on varied strongly nonlinear systems which undergoes rapid changes only using low operating data. [1]**

I. Introduction

Advent of powerful processors and advances in machine learning algorithms has caused revolution in many fields that was purely theoretical and mathematical or empirical. This has propagated to controls as well. One of the ubiquitous methods used in the industry to control dynamical process is, Model-based control, Model Predictive Control.

Despite tremendous progress in the big data algorithms, their application on physical dynamical systems is lacking on a few fronts. For example, machine learning algorithms cannot easily implement physical constraints. In addition, trained model says a little about system's dependency on input parameters. Add to this the fact that Machine learning algorithm requires a lot of data train since they have lots of input features, it is a luxury for lots of systems. Also, they cannot be trained online on abruptly changing systems. Another limitation of Machine learning based models is, they suffer from poor generalization capabilities.

To answer these shortcomings, SINDy has been developed for dynamical system, which can provide a lot of insight about system's physical properties and requires little data. In this paper author attempts to apply SINDY to control non-linear dynamical system using Model predictive control. Basic idea is to train model and get physical insights into the system. And use this model of the system as input to Model predictive controller.

Model predictive control (MPC) takes advantage of system model to control strongly nonlinear systems. The reason for this is, MPC solves optimization problem as an open-loop controller for some time into the future and implements the solution for a time-step and again uses feedback to incorporate deviations or disturbances in the system. MPC can take care of multiple input multiple output system, systems with time delays, instability in the system. In addition, it is easy to implement physical input as well as output constraints, both soft and hard constraints.

Only draw back of the MPC is, it requires a fairly accurate model of the system. Traditionally system is modelled using experiments and extensive mathematical calculations. This has been a major roadblock to readily implementing MPC universally. There exists another method to get system model from data, for example, Eigen-system realization algorithm, subspace identification methods. But the limitations of these methods are, they cannot provide output that is easily interpretable by humans.

This paper concentrates on solving model identification problem with minimal data and as rapidly as possible. This is particularly useful when a strongly nonlinear system undergoes major change. This can cause system behaviour to change. And there often is not enough time for automatic control to perform trial and error to bring back system to stability. In this case, author suggests to solve for system dynamics using SINDy and using the output of SINDy to get model, which then will be used for Model predictive control.

II. SINDY-MPC framework

The SINDY - MPC framework is to control system in two steps, first SINDY will perform online training of the model which will eventually be used to control the system.

1) Identifying governing equation of nonlinear system with control.

This paper is an attempt to model control system along with system's dynamics using behaviour data of the system. Theme of the paper is to identify most crucial terms. Most system are dependent upon very few terms as opposed to most general machine learning model seem to emphasis. In general, machine learning model will identify and weight almost all the input features. Whereas SINDy is an attempt to identify most important terms from the system at the cost of slight accuracy.

This paper uses Sparse Identification of Non-linear dynamics (SINDy) method and implements that on a system with control. SINDy identifies few active terms of governing equation using system data. For this, sindy uses a library of functions built by system expert who have some knowledge of what system could depend on. This method can also be equated to creating feature crosses in popular machine learning. Once the library is built, SINDy will identify active terms from the library.

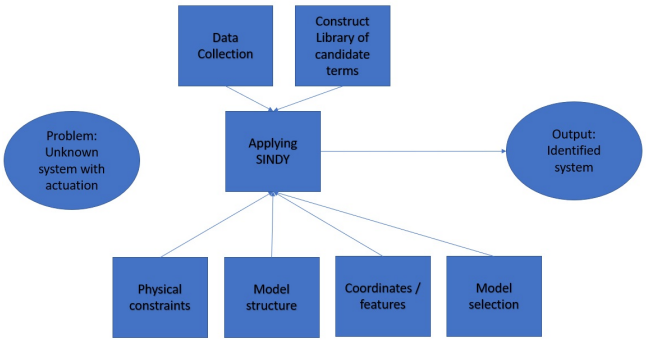
Here SINDy is extended to include actuation. Hence function library will also be extended to include u1, u2, u3,... Since system can be nonlinear, function library can also include nonlinear feature crosses, i.e. nonlinear functions in x and u.

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_m \end{bmatrix}$$

$$U = \begin{bmatrix} u_1 & u_2 & \dots & u_m \end{bmatrix}$$

Since this is a time series analysis, we will measure m measurements at different times and arrange them in matrix format according to above equation.

Library: it is important to choose right functions to include in the library. SINDy approximates the system behaviour using these functions. And if the right function is not present, it is going to approximate the behaviour using the combination of present functions. Common strategy is to start from polynomial functions and include trigonometric functions. One can also include functions using knowledge of the system physics.



$$\dot{X} = \xi * \theta^T(X, U)$$

where ξ is coefficient matrix found by SINDYc.
Time-derivative of system state can be measured directly or can be calculated using numerical difference.
This equation is then solved for optimization as explained in the following sections.

2) Finding discrete-time dynamics.

SINDY paper is extended here to include controls in discrete-time dynamics. It was shown in the SINDY paper that $x_{k+1} = F(x_k)$.
Here, instead of computing derivatives, author simply uses X with furthering one time-step. Hence,

$$X^{'} = \begin{bmatrix} x_2 & x_3 & \dots & x_{m-1} \end{bmatrix}$$

Hence dynamics may be written as,

$$X^{'} = \Xi * \theta^T(X, U)$$

and the regression problem for discrete time dynamics becomes...

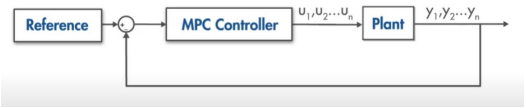
$$\xi_k = \arg\min_{\xi_k} \|X_k^{'} - \xi_k * \theta^T(X, U)\| + \lambda * \|\xi_k\|$$

This optimization may be solved by Lasso model or sequentially thresholded least squares procedure. The goal is to solve regression problem using a small number of features. This will inevitably create a compromise between number of variables vs accuracy. We employ the least absolute shrinkage and selection operator (lasso) method for this goal. The lasso performs both variable selection and regularization to optimize between accuracy and a smaller number of variables. The lasso combines least squares method and l1 norm regularization.

III. Model Predictive Control

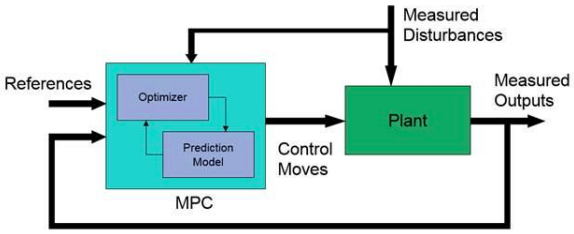
Model Predictive Control is a feedback control algorithm that uses system model to make predictions about the system trajectory and performs optimization based on system characteristics for the inputs. Model Predictive

Control (MPC) is the extended version of optimum control in the sense that it calculates optimization at each time step.
MPC can handle multi-input multi-output systems. And it can also take constraints into consideration while optimizing the inputs.



Controller tracks reference trajectory as close as possible. And performs calculations to obtain inputs for next few time steps. (Although it does not apply all the calculated inputs.) But these inputs for the future are used to perform optimization calculations. Then once into the next-step, it performs these calculations again. That's why MPC is also called receding horizon control.

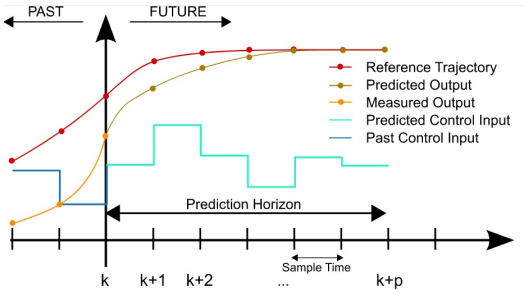
MPC has the capability to make predictions about system till certain time period into the future. It is called Prediction Horizon.
MPC can control up to certain time-steps into future. This period into future is called Control horizon. The length of control horizon is determined by amount of computing resources available for the controller Since it calculates optimization at each of the time-step, i.e. online optimization problem. Sample time is the time gap at which system measurements are taken.



reference: Mathworks.
MPC will assign cost to each of the input and change in the input through its control horizon. It will then select the path with least least cost. This condition ensures smooth functioning of the system.

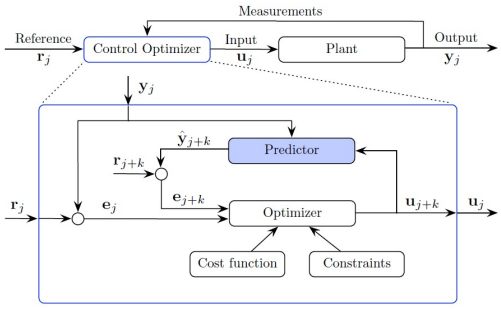
Once calculation is done and optimal input scheme is decided, it is only applied to only the first time-step. And system might not be where it was predicted using the model previously, it takes state measurements and computes the whole optimization problem again. The state optimization can be done using Kalman filter if not full-state measurement is not available.

For this reason, it would be waste of computing resources if control horizon is extended unnecessarily. In addition, this is the reason why MPC gives good result even for strongly non-linear systems even with fairly poor models.



reference: Wikipedia.

IV. MPC IN REGARDS TO SINDYC



The receding horizon problem in a sense in open-loop optimization problem where controller determines sequence of inputs for control horizon such that cost of input is minimized over prediction horizon. Control horizon is usually smaller than or equal to prediction horizon. In case of control horizon being small, input is considered constant beyond control horizon time-steps.

cost optimization at each time-step is given by:

$$\min_{u(.|x_j)} J(x_j) = \min_{u(.|x_j)} [A + B + Z]$$

where

$$A = \|X_{j+m_p} - X_{m_p}\| * Q_{mp}$$

$$B = \sum_{k=0}^{m_p-1} \|X_{j+k} - X_k\| * Q$$

and

$$Z = \sum_{k=1}^{m_c-1} (\|u_{j+k}\| * R_u + \|\Delta u_{j+k}\| * R_{\Delta u})$$

where Δu and u are constrained.

powThis equation is solved using LASSO or sequentially thresholded least squares method. The second term in this equation promotes sparsity. That is selected according to Pareto optimal model that balances accuracy with sparsity.

For our paper implementation, we assume that full state measurements are available. The cost function J optimizes over input, change of input and deviation from reference trajectory.

Here $Q, Q_{m_p}, R_u, R_{\Delta u}$ are weight matrices.

Here system governing equation is given by SINDYc model in discrete-time manner. Author adapts two stage model. First he identifies the system using data and this system model is then used in MPC.

V. Lorentz Chaotic System:

Chaotic systems are defined by simple governing equations but with large dependencies on initial conditions. i.e. Two chaotic systems governed by same equations will have vastly different trajectories if their initial conditions are different even by a small margin. These systems are completely deterministic but still largely unpredictable.

Lorentz system is a system of O.D.Es studied by Mathematicians of the same name. We will try to model Lorentz system. It is a simplified mathematical model for atmospheric convection. And the model is defined by three O.D.Es known as Lorentz equations.

We will be studying the Lorentz system with control input. i.e. we will attempt to stabilize Lorentz system about a fixed reference point.

VI. Model Parameters and Assumptions:

Please note that we are trying to stabilize the system w.r.t to a fixed set point, 10, as opposed to paper which tries to control system around the weakly unstable fixed points in 3D. $(\pm\sqrt{72}, \pm\sqrt{72}, 27)$.

VII. Implementation

Lorenz system is constructed for getting the simulated data which will act as the ground truth for validation. Differential equations were written and solved by ode45 in Matlab to get the output shown in figure 1. Also, known by its famous name "The Butterfly Effect". Lorenz system's initial parameters were:

$$x = 0, y = 1, z = 20 \tag{1}$$

And, its chaos parameters were:

$$\sigma = 10, \rho = 28, \beta = \frac{8}{3} \tag{2}$$

Now, we assume that we don't know the model. And, only have the **X = [x, y, z]** data are available to us. From these state variables, **[dX = dx, dy, dz]** are calculated straightforwardly. Note that the δt = 0.001 has been taken for 100 seconds. With this data, we calculate the permutation of all the state variables up to order 1. See figure 2 Each column in the figure

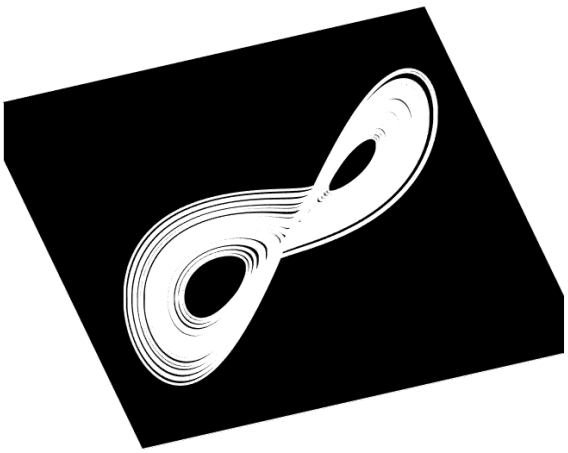


Fig. 1. Lorenz System

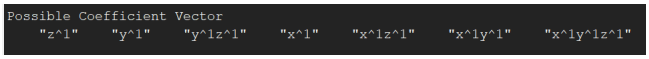


Fig. 2. Possible Coefficient Vectors

2 is a vector. And, the outputs [dx, dy and dz] is some form of a combination of these state variables. [2] Let's say all permutations of state variables is denoted by the Matrix, **P**. Now, we need the compute the relationship between dX and X, let's say **S. S** is actually a sparsity matrix that gives the relationship between dX and X.

$$dX = X * S \tag{3}$$

To calculate this Sparsity Matrix we use a method called Lasso Regression [3] given by the formula in figure 3: In our case, y

$$\beta_{\text{lasso}} := \arg\min_{\beta} \frac{1}{2}\|y - X^T\beta\|_2^2 + \lambda\|\beta\|_1,$$

Fig. 3. Lasso formula

is dX, and β = S. And, we need to minimize this for the value of β. Once this Sparse Matrix is calculated, we get some non-linear dependent state variables close to what the system thinks it is with the given data. In our case we get: see figure 4 Now, we take these dependent variables and create a Simulink model of the system in figure 5 With the above lorenz system, an Model Predictive Block was initialized taking Simulink SINDy generated model as its predictive model(also known as the Plant) [4]. The input constraints have been given [-50,50]. So, the MPC input signal never overshoots these boundaries. The horizon is 10s. And the control horizon is 2s with a time step of 0.01s. Note, the MPC input signal is only added to the dx

```
---------------
dx coefficients
---------------
 | y^1 | -x^1
---------------
dy coefficients
---------------
 | y^1 | -y^1z^1 | x^1 | -x^1z^1
---------------
dz coefficients
---------------
 | -z^1 | x^1y^1
```

Fig. 4. Non-linear Dependent State Variables generated from SINDy

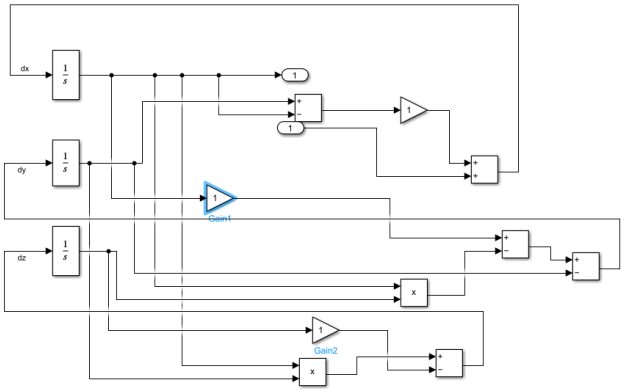


Fig. 5. Simulink Model of the SINDy Generated system. Note all gain blocks of chaos parameters are 1. Because the system has no knowledge of the chaos parameters of the system

variable. And, we try to reach a reference signal of a constant value of 10. See fig 6

$$dx = y(t) - x(t) + u(t) \tag{4}$$

And finally, this is the final output which we can see MPC

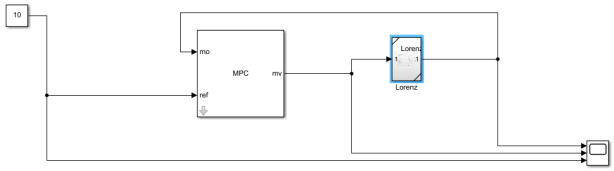


Fig. 6. MPC block added to SINDy Lorenz System

generates the input to match the reference output. See Figure 7. Note, the yellow line represents the chaotic x value of the lorenz system, an MPC input by the Controller is generated in the blue line which tries to match the reference output of constant 10 which is the red line.

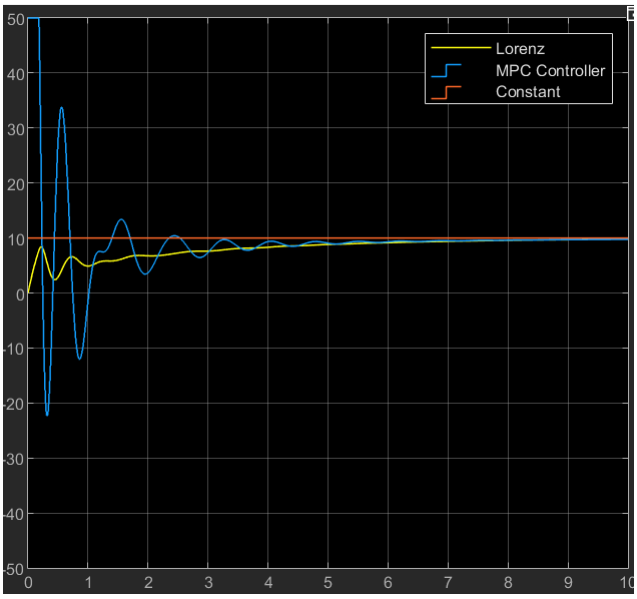


Fig. 7. Final output of the MPC Input Signal to reach reference output signal

VIII. Conclusion

An overview of the whole process is given in figure 8. In brief, governing general equations for a non-linear dynamic system are discovered using the sparse-regression Lasso Model on the state variables data. These equations become the plant for the Model Predictive Control to match a reference output and a corresponding input signal is generated with some hard constraints on the input signal.

IX. Deviations

We have implemented code to demonstrate understanding of SINDYc. (Application of SINDY on MPC). And we are deviating from original paper on number of topics that are listed below with explanations:

1. Instead of applying SINDYc to all the nonlinear system author has implemented, we are only implementing it on one system. (We asked for permission from the professor regarding this and he had approved for it.)
Although we initially thought of implementing SINDY + MPC for F8 Crusader aircraft settled on Lorentz Chaotic system. Please note that both are strongly nonlinear system (with only difference F8 system having higher order polynomial nonlinearity.) We first thought of implementing it on Lorentz system and then do the same for F8 system, but we ran out of time.

2. Not implementing DMDc and Neural Network for comparing their performance with SINDYc: We asked for professor's permission to not implement these methods, since both are state-of-the-art techniques with their own nuances. And professor did approve this exception as well.
3. Author tries to show superiority of SINDYc over DMDc and NN over varying amounts training data and time taken to

train them. We did not perform training using varying size of training data. Instead we aimed to actually train the controller to control the chaotic system. We have shown model of the system and graph of controller performance.
4. Controller reference point is different. (Explained earlier in VI section: Model parameters and assumptions.)

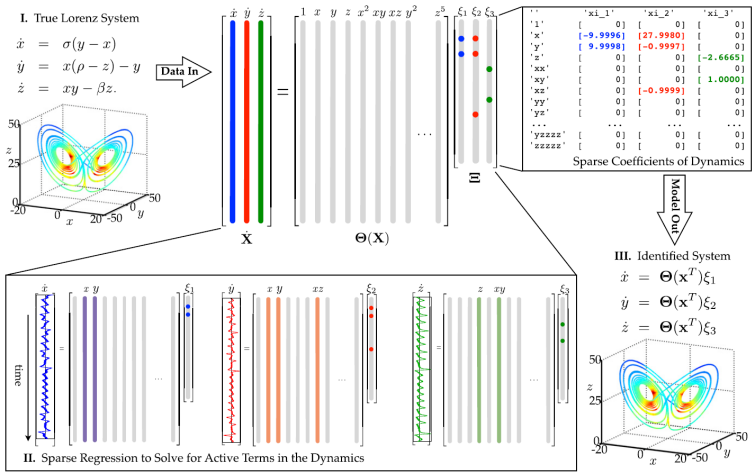


Fig. 8. Schematic of the SINDy algorithm, demonstrated on the Lorenz equations. Data are collected from the system, including a time history of the states and derivatives; Next, a library of nonlinear functions of the states is constructed. This nonlinear feature the library is used to find the dependent coefficients by sparse regression

REFERENCES

[1] 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*, 474(2219):20180335, nov 2018.

[2] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[3] Tibshirani R. Wainwright M. Hastie, T. *Statistical Learning with Sparsity: The Lasso and Generalizations (1st ed.).* 2015.

[4] Taku Takahama and Daisuke Akasaka. Model predictive control approach to design practical adaptive cruise control for traffic jam. *International Journal of Automotive Engineering*, 9(3):99–104, 2018.