Planar Pushing Task using SINDy Model

Madhav Rawal, Aditya Paranjape
{madhav, adityaap}@umich.edu

***Abstract*—Abstract - In this project we use the SINDy model to accomplish a planar pushing task. SINDy is custom deep autoencoder network to discover a reduced coordinate system where the dynamics are sparsely represented. This allows for simultaneous learning of governing equations and associated coordinate systems, combining the strengths of deep neural networks and sparse identification of nonlinear dynamics. We collect the data of the manipulator in the simulation environment. This data is passed through the SINDy Autoencoder to discover the dynamics and predict the next states. We show a comprehensive analysis of the SINDy model with different hyperparameters, dynamics methods, and a comparison with the E2C model.**

I. Introduction

Advances in sparse regression are allowing us to identify the structure and parameters of nonlinear systems from data, resulting in models with the fewest terms necessary for describing the dynamics. To achieve this, an effective coordinate system is needed. SINDy is custom deep autoencoder network to discover a reduced coordinate system where the dynamics are sparsely represented. This allows for simultaneous learning of governing equations and associated coordinate systems, combining the strengths of deep neural networks and sparse identification of nonlinear dynamics. The discovery of coordinates and models are given equal importance in this method. The motivation of using SINDy is that it can discover dynamics in latent space based on the data collected. In this project we use the SINDy model to accomplish a Planar Pushing Task using images of the goal and start positions. The versatility of SINDy model makes it a very good choice to use when the environment in which the robot is going to operate does not change. In such scenarios SINDy model could outperform traditional techniques.

II. Background

Neural networks (NNs) have become increasingly popular for studying complex dynamical systems. While NNs have traditionally been successful in image classification and speech recognition, they have also been applied to a wide range of scientific and engineering problems. In particular, deep learning techniques have been used to improve the solution of dynamical systems with known equations and to understand and predict the behavior of complex systems with unknown equations. Several methods have been developed to train NNs to predict dynamics, including time-lagged autoencoders, recurrent architectures like LSTMs, and reservoir computing. Additionally, deep learning has been used to discover coordinates for Koopman analysis, a theory that seeks to linearize nonlinear dynamics. Despite their widespread use, NNs face several challenges, including generalization, extrapolation, and interpretation. The ability to extrapolate, and therefore generalize, is a known weakness of NNs and is particularly relevant in the context of dynamical systems and forecasting. Additionally, NN models can be difficult to interpret due to their complicated architectures and large number of parameters. However, NN methods still have the potential to learn general and interpretable dynamical models if they are properly constrained or regularized. For example, deep learning has been used for parameter estimation of partial differential equations and for discovering linear embeddings. With continued development and refinement of NN techniques, they hold promise for advancing the study of complex dynamical systems in the future.

III. Implementation

Our aim is to learn the dynamics of the robot arm and accomplish a planar pushing task. We started with the aim of collecting data of the environment with random actions, then performed data processing. We used data in the SINDy autoencoder to discover the coordinates and a sparse matrix to represent the dynamics. Further, we use a MPPI controller in the latent space to plan the trajectory of the block to the goal state.

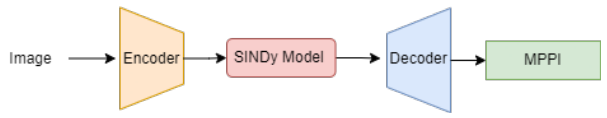

Fig. 1: Flowchart of the Algorithm

A. Environmental Setup

In this scenario, we have a robot arm that is operating within a gym environment. The robot arm's actions are defined by three parameters: the location on the edge where it will push, the angle at which it will push, and the length of its push. However, the state space of the environment is represented by a 32x32 image captured from an overhead viewpoint. This means that the robot arm must use its sensors to interpret the visual information it receives from the image to determine its actions within the environment. The image provides a snapshot of the state of the environment at a particular moment in time, and the robot arm must analyze the image to make decisions about where and how to push. The action space of the robot arm is represented in the fig3, which shows the range of possible pushing locations, angles, and lengths. The fig2 shows the planner pushing environment, in which the green area is the target pose for the object.

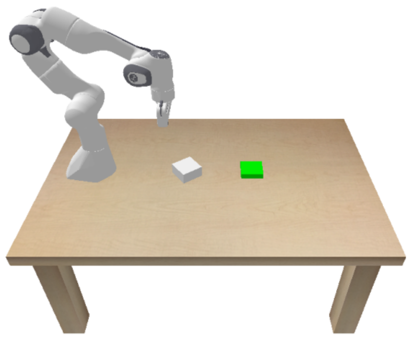

Fig. 2: Pushing Environment

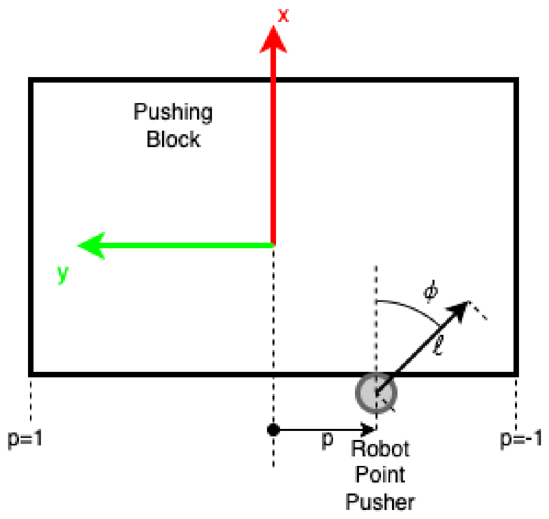

Fig. 3: Action Space

Each action $\mathbf{u} = \begin{bmatrix} p & \phi & \ell \end{bmatrix}^\top \in \mathbb{R}^3$ is composed by:

$p \in [-1, 1]$: pushing location along the lower block edge.

$\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ pushing angle.

$\ell \in [0, 1]$ pushing length as a fraction of the maximum pushing length. The maximum pushing length is is 0.1m

B. Data Collection and Processing

To train our robot arm in a gym environment, we perform various trajectories and collect data in the form of images. To do this, we randomly initialize the object start pose and uniformly sample actions within the limits of the action space. By collecting data for a specified number of trajectories and trajectory length, we can capture a diverse range of possible scenarios and actions.

To prepare the data for training, we normalize the dataset to ensure its consistency and reliability. We then create a PyTorch Dataset and keep the number of steps flexible for single or multi-step analysis. This allows us to analyze the data at different levels of granularity and gain a deeper understanding of how the robot arm operates within the environment. By using this approach, we can identify areas for improvement and optimize the robot's performance, enabling it to operate more effectively in a wide range of environments and scenarios.

C. SINDy Autoencoder

This approach uses a combination of a SINDy (Sparse Identification of Nonlinear Dynamics) model and a deep autoencoder network to discover intrinsic coordinates and develop an associated parsimonious nonlinear dynamical model. The collected data is passed through an Encoder to create a latent space, where the SINDy model is used to learn a matrix of coefficients. A library of polynomial functions is created, which is a function of the latent state coordinates. The next latent state is predicted by multiplying the library of functions with the coefficient matrix. The aim is to learn a sparse coefficient matrix, so the approach includes a sequential thresholding process to replace all the numbers below a threshold value with zero. This helps to achieve the goal of having a sparse coefficient matrix with the least number of active coefficients needed to represent the dynamics of the unknown system.

The image input is passed through the encoder (fig4) and a latent space of 16 dimensions is created. The SINDy model is used in this latent space to discover the dynamics and predit the next latent state. These latent states are decoded (fig5) to the original state space

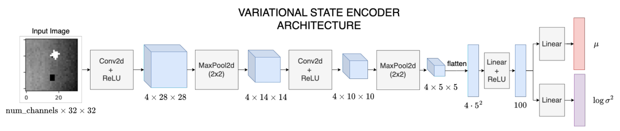

Fig. 4: Encoder Architecture

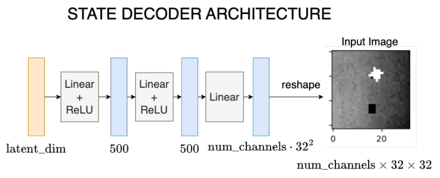

Fig. 5: Decoder Architecture

To train the model for a sparse coefficient matrix and the least number of active coefficients, the SINDy loss function consists of a sum of different components. The reconstruction loss is calculated based on the input and the decoded latent states, while the SINDy loss in states(‘x’) and the SINDy loss in latent states(‘z’) are used to identify the nonlinear dynamics of the system. Additionally, a regularization term is included to ensure that the model does not overfit the data. The loss function used for this approach can be seen in the provided equation.

$$L = \|x - \hat{x}\|_2 + \lambda_1 \|x_{t+1} - \hat{x_{t+1}}\|_2 + \lambda_2 \|z_{t+1} - \hat{z_{t+1}}\|_2 + \lambda_3 \|\epsilon\|_1 \tag{1}$$

By training the model with this loss function, we can develop a parsimonious nonlinear dynamical model that can effectively represent the unknown system’s behavior. This approach has the potential to significantly reduce the complexity of modeling nonlinear dynamical systems, making it possible to analyze and optimize these systems in a wide range of applications.

We used two different approaches to formulate the SINDy dynamics, the Absolute SINDy model and the Residual SINDy model. In the Absolute dynamics we learn the matrix to directly predict the next states and in the Residual model we learn the difference between the current and the next state. Below we can see a general equation of the dyanmics. The function here is our SINDy model.

Absolute Dynamics -

$$\hat{\mathbf{z}}_{t+1} = f(\mathbf{z}_t, \mathbf{u}_t)$$

Residual Dynamics -

$$\hat{\mathbf{z}}_{t+1} = \mathbf{z}_t + \Delta\mathbf{z}_t = \mathbf{z}_t + f(\mathbf{z}_t, \mathbf{u}_t)$$

D. *Controller*

We use the learned dynamics model to formulate a control problem in latent space. We use a MPPI to propagate the dynamics in the latent space. The cost function can be formulated in latent space or in the image space as well. We are comparing how close states are to the goal in the chosen state space.

IV. Results and Discussion

The experimental setup we used is as described in the environmental setup section. As we discussed earlier there are two possible SINDy dynamics model that can be formulated. We have shown a comprehensive analysis of both the models. We have presented a analysis of the effect of changing the polynomial order. We have compared their reconstructions for single and multi step dynamics. The distinctiveness of SINDy arises from discovering sparse dynamics, hence we have compared the sparsity of the learned matrix for both methods. Based on the comparison we conclude that the Absolute SINDy model performs better than the Residual model. We further compare the Absolute SINDy model with the simplified E2C model.

A. *Absolute and Residual Dynamics*

The SINDy model was trained to learn two types of dynamics: absolute and residual. In the absolute model, the next state was predicted directly from the input of latent state and action concatenated. This model achieved a loss of 0.06 and 19 active coefficients. In the residual model, the model learned the difference between the next state and the current state from the input of latent state and action concatenated. This model achieved a loss of 0.05 and 0 active coefficients. Both these models were trained on 1600 epochs with a learning rate of 0.001 and had a threshold of 0.05. Fig6 and fig7 show the loss curve and the active coefficients for the absolute dynamics model and fig8, fi69 loss curve and the active coefficients for the residual dynamics model. Fig11 shows the reconstruction of the images from the two algorithms.

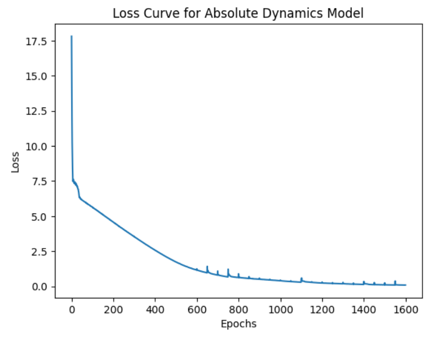

Fig. 6: Loss curve for Absolute Dynamics

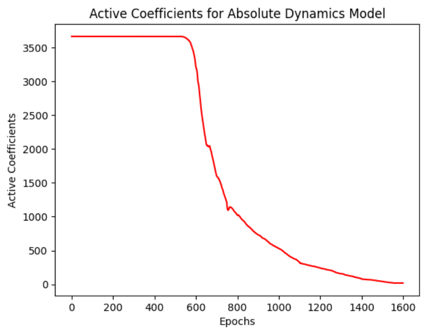

Fig. 7: Active Coefficients in Absolute Dynamics

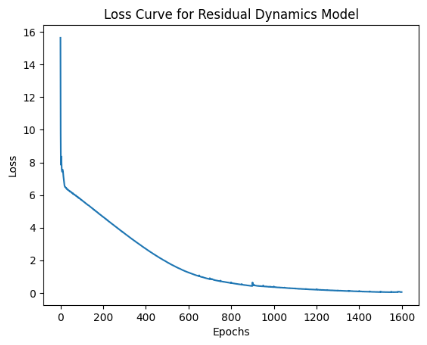

Fig. 8: Loss Curve for Residual Dynamics

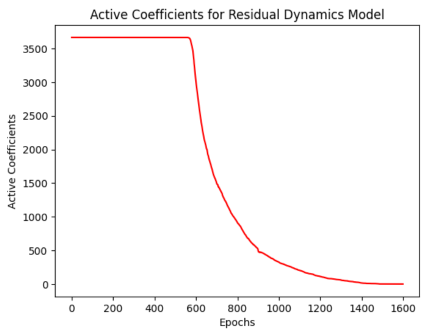

Fig. 9: Active Coefficients in Residual Dynamics

It can be seen that the reconstructions from the absolute dynamics capture the dynamics in both multi step and single step relatively well. The residual model has 0 active coefficients. This means that the residual model has not learnt any dynamics and only learnt to reconstruct the images. This can be seen clearly in Fig11 as it has clear single step dynamics predictions but the multi-step predictions are completely out of order. We suspect that as the residual dynamics SINDy model does not have to learn the dynamics of the state and instead learn the difference between two latent states, it fails to capture the minute differences between the two. There may be subtle differences that might need very low thresholds for any active coefficients to learn the dynamics. We keep this idea for the future scope of the project.

The library functions included are 0 order, 1 order, 2 order, and sine of the latent vector. Fig10 shows the sparse matrix learnt by the absolute dynamics model for polynomial order 2. The highlighted terms in the image show the coefficients above the threshold that influence the dynamics of the system. We can see that from our defined library of functions, the SINDy model used the elements with a power of one and a few sine terms to define the dynamics of the system.

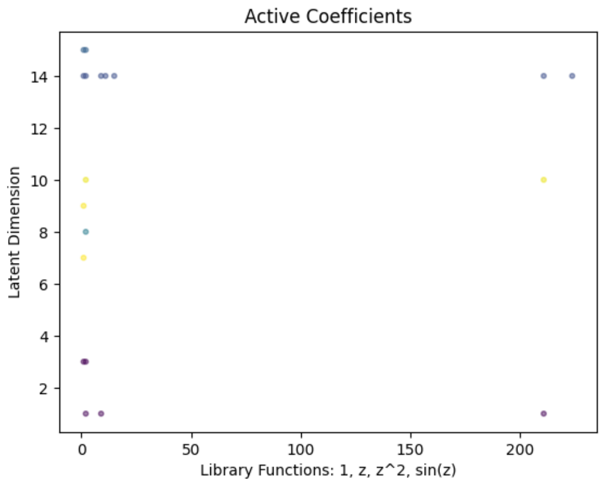

Fig. 10: Absolute Dynamics Sparse Matrix Learnt

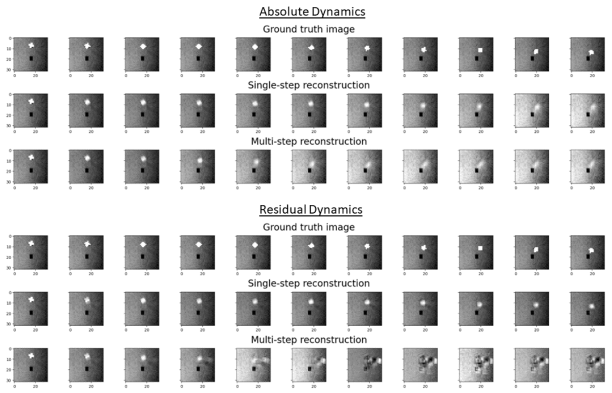
Absolute Dynamics
Ground truth image
Single-step reconstruction
Multi-step reconstruction
Residual Dynamics
Ground truth image
Single-step reconstruction
Multi-step reconstruction

Fig. 11: Reconstructions from Absolute and Residual dynamics for single step dynamics

B. E2C comparison with SINDy Model

Based on the comparison between the results of the Absolute SINDy model and the simplified E2C framework from fig12, it can be observed that the reconstructed images for the single step SINDy model and the

E2C framework are quite similar. However, for multi-step reconstructions, the E2C framework outperforms the Absolute SINDy model in terms of reproducing better results. The reason we suspect for this is that the SINDy model predicts the next states continuously for the specified number of steps, which leads to the accumulation of errors over all the steps. Consequently, the loss begins at a very high value and this error accumulation persists over the next steps, resulting in a final loss value that is higher than that of the E2C framework.

To further investigate this, we ran the multi-step Absolute SINDy model for 2, 3, and 4 steps. Only when the number of steps was set to 2, we were able to achieve a reasonable loss and reconstructions. However, even with this improvement, the reconstructions produced by the multi-step Absolute SINDy model are not as good as those reproduced by the multi-step E2C framework. The multi step model of E2C can correctly reconstruct the single step as well as the multi step reconstructions. The similar effect is seen in the multi step SINDy model.

To evaluate the performance of the two methods, we conducted a planar pushing task using the single step dynamics for both models. We measured the number of times the models were able to reach the goals out of a total of 10 attempts. We found that the SINDy single step model reached the goal all 10 times, while the E2C model reached the goal 8 out of 10 times. Additionally, we observed that on average, the E2C model required two steps less than the SINDy model to reach the goal.

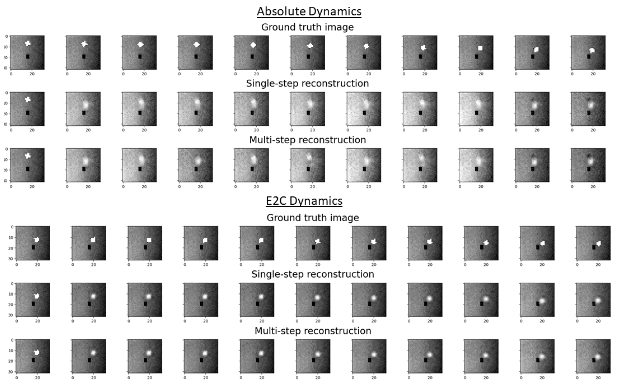

Fig. 12: Reconstructions from Absolute Dynamics Model and E2C Model

C. *Analysis of Polynomial order*

We ran the Absolute SINDy model with single step dynamics for polynomial of orders 1,2, and 3. We calculated the the sparsity percentage of the learned matrix for the polynomial orders. The percentage of the active coefficients for 1st order came out to be 1.2%, 2nd order is 0.5% and for 3rd order polynomial is 3.6%. Further we also analysed the planar pushing task for all the polynomial orders and found that each one almost takes similar number of steps to reach the goal. Based on the analysis we decided to chose the polynomial with order 2. The figures show the sparsity of the matrices for different polynomial orders.

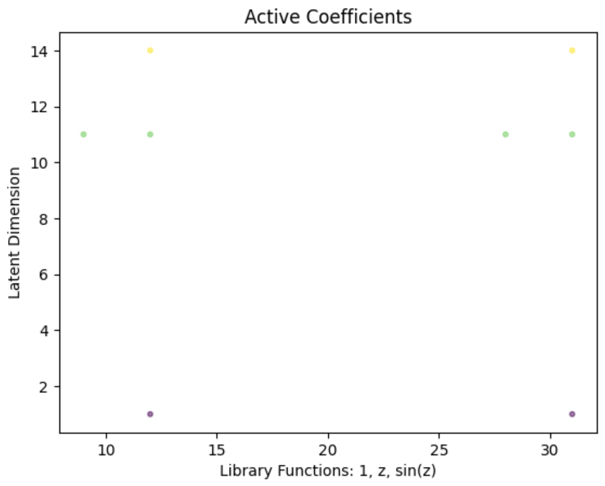

Fig. 13: Sparse Matrix with polynomial order 1

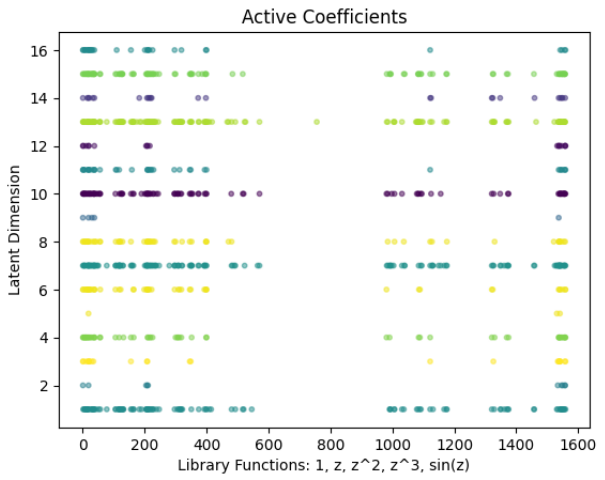

Fig. 14: Sparse Matrix with polynomial order 3

V. Conclusion and Future Scope

We have conducted experiments on the SINDy model, which is a discovery model used for identifying the underlying dynamics of a system. Our findings indicate that the SINDy model performs better with absolute dynamics rather than residual dynamics. This suggests that the model is better suited for systems that have clear and well-defined dynamics, as opposed to systems with complex and noisy data.

We have also observed that the SINDy model performs similarly to the E2C model, which is a well-known method for learning the dynamics of a system. However, we found that the SINDy model has limitations in processing multiple steps of a system.

To improve the SINDy model's performance, we propose several future research directions. Firstly, we could fine-tune the SINDy model to work with residual dynamics, which would enable it to handle more complex and noisy data. Secondly, we suggest incorporating collision avoidance into the controller to make the model more robust and practical. Lastly, we could include dynamics to push the block from all directions, which would further enhance the model's accuracy and reliability.

Overall, these future research directions have the potential to create a more advanced and accurate SINDy model that can handle more complex systems with multiple steps and collision avoidance, making it more applicable in real-world scenarios.

References

[1] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *Proceedings of the National Academy of Sciences*, vol. 116, no. 45, pp. 22 445–22 451, 2019.

[2] S. L. Brunton, J. 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*, vol. 113, no. 15, pp. 3932–3937, 2016.

[3] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," *Advances in neural information processing systems*, vol. 28, 2015.