Technical Report

A Signature-Based Approach to System Identification and Prediction of Controlled Dynamical Systems

Vladislav Snytko
Maastricht University
vvs.snytko@gmail.com
v.snytko@student.maastrichtuniversity.nl

Supervisor: George Stepaniants

October 18, 2025

1 Introduction

Control theory concerns the problem of selecting input trajectories for a system to drive its output toward desired values, with the ultimate goal of being able to control any nonlinear dynamical system. In practice, however, the equations governing a system's dynamics are often unknown, and the only available information consists of data — samples of inputs and their corresponding outputs. The necessity of obtaining a representation of a system before controlling it lies at the heart of the system identification component of the twofold control task. In this report, we present our results on exploring a novel signature-based method for system identification and, ultimately, for the control of dynamical systems.

An autonomous controlled dynamical system can be expressed as

$$\dot{x}(t) = f(x(t), u(t)) \tag{1}$$

where $x(t) \in \mathbb{R}^m$, $u(t) \in \mathbb{R}^n$, $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ and are all, generally, nonlinear functions. The method we are about to present requires the system to be *control-affine*, meaning that f should be linear in inputs $u(t)$. This is not a significant limitation, since Equation (1) can be rewritten in the control-affine form through coordinate lifting $\tilde{x} = (x, u), \tilde{u} = (\dot{u}, 1)$ resulting in the system

$$\dot{\tilde{x}} = \begin{bmatrix} f(\tilde{x}) \\ \dot{u} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n\times m} & f(\tilde{x}) \\ I_{m\times m} & \mathbf{0}_{m\times 1} \end{bmatrix} \tilde{u} := F(\tilde{x})\tilde{u}. \tag{2}$$

Now, the signature $S_{[a,b]}(u)$ of a trajectory u is an operator that returns an infinite series of numbers $S^{(i_1,\dots,i_k)}_{[a,b]}(u)$ for all $1 \le i_1, ..., i_k \le m$ and $k \ge 0$ given by the iterated integrals (Fermanian et al. [2021])

$$S^{(i_1,\dots,i_k)}_{[a,b]}(u) = \int_{a<t_k<b} \cdots \int_{a<t_1<t_2} \dot{u}_{i_1}(t_1) \dots \dot{u}_{i_k}(t_k) dt_1 \dots dt_k. \tag{3}$$

Formally, the signature is represented as follows

$$S_{[a,b]}(u) = (1, S^{(1)}_{[a,b]}(u), \dots, S^{(m)}_{[a,b]}(u), S^{(1,1)}_{[a,b]}(u), \dots, S^{(m,m)}_{[a,b]}(u), \dots). \tag{4}$$

What is interesting for our discussion is that the signature can be used to write out the solution for the Equation (1) (Fermanian et al. [2021]), where it accepts the control trajectory and combines with coefficients dependent on x_0

$$x(t) = x_0 + \sum_{k=1} \frac{1}{k!} \sum_{1\le i_1,\dots,i_k\le n} S^{(i_1,\dots,i_k)}_{[0,t]}(U) \cdot F_{i_1} \star \cdots \star F_{i_k}(x_0), \quad U(t) = \int_0^t u(\tau) d\tau \tag{5}$$

where $\star$ denotes differential product

$$f \star g = \sum_{i=1}^{n} \partial_i g \cdot f_i. \tag{6}$$

In practice we should use a finite expansion, so we reserve $S^K_{[a,b]}(u)$ to denote truncated series Equation (4) where the last term is $S^{K\times(n)}_{[a,b]}$, $K \times n = (n, \dots, n)$, where n is copied K times. With this we can write out the model succinctly

$$\widehat{x}(t) = S^K_{[0,t]}(U) G_K(x_0) \tag{7}$$

where $S^K_{[0,t]}(U) \in \mathbb{R}^{\frac{n^{K+1}-1}{n-1}}$ and $G_K : \mathbb{R}^m \to \mathbb{R}^{\frac{n^{K+1}-1}{n-1}\times m}$ is a stacked vector of differential products.

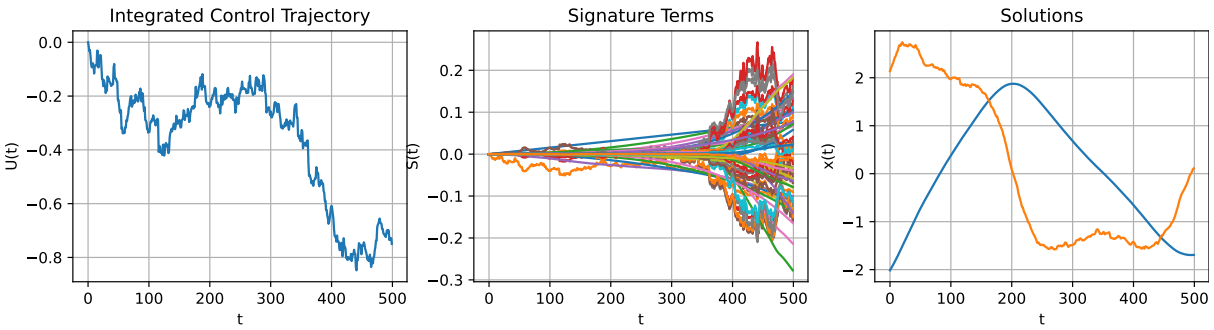


Figure 1: Example solution for Duffing oscillator and corresponding signature terms

The model we present is worth studying because it possesses several advantages, which we outline below. **First**, it is straightforward to interpret: one can directly observe the feature trajectories (signature terms) that are combined to produce predictions, and the exact function that computes them, Equation (4), is explicitly known. An example solution for the Duffing oscillator and the corresponding signature terms are given in Figure 1. This level of interpretability stands in contrast to the common approach in dynamical systems of training a neural network that maps inputs directly to outputs (see Lewis et al. [1998] and Perrusquía and Yu [2021]). Such models are typically black boxes, as the transformations applied to the inputs during the forward pass are not transparent. **Second**, the signature transform model is expressive enough to represent any dynamical system, unlike, for instance, Spectral State Space Models (see Agarwal et al. [2024]), which require additional architectural complexities to capture nonlinear dynamics. **Third**, the signature provides a simpler nonlinear transformation than that used in other predictive methods, such as reservoir computing (see te Vrugt [2024]), since it admits a closed-form expression for $S^K_{[0,t]}(U)$.

The aim of this report is to (i) summarize the methodology and experiments, and (ii) provide a record of the results for future reference. All experiments were implemented in Python, and the complete source code is publicly available online[1] .

2 Methodology

2.1 Signature Representation

We represent each control trajectory $u(t)$ through its truncated signature expansion $S^K_{[0,t]}(U)$, where $U(t) = \int_0^t u(\tau)d\tau$. The signature terms serve as the building blocks of the predictive model. The signatures are computed with *signax* library in Python (Tong [2022]). The default $K = 5$, if not mentioned otherwise.

2.2 Coefficient Matrix

The solution is modeled as a linear combination of signature terms with coefficients determined by a matrix $A(x_0)$, which depends on the initial condition x_0.

2.3 Error metric

Error metric between target trajectories and predictions that we are going to use in the rest of the report is Normalized Root Mean Square Error (NRMSE), which is normalized by the range of the target trajectory

$$E(y, \hat{y}) = 100 * \frac{\sum_{i=0}^{l}(y_i - \hat{y}_i)^2}{l(max(\hat{y}) - min(\hat{y}))} \tag{8}$$

[1] `https://github.com/kurolesica0211/signature-based-prediction`

where $y, \hat{y} \in \mathbb{R}^{l \times m}$ are predicted and target trajectories correspondingly with l as the number of samples and m dimensionality of the state.

2.4 Linear Regression Baseline

To validate the feasibility of the approach, we implemented a linear regression model to learn $A(x_0)$ for a fixed x_0 across multiple control trajectories. The objective is to observe a decrease in the test error as the number of trajectories used in the regression increases. Such a decrease would indicate the convergence of the coefficients A toward values that accurately model the dynamics for different input trajectories.

2.5 Neural Network Extension

Building on this, we trained a feedforward neural network to predict $A(x_0)$ directly from x_0. The predicted coefficients were then combined with the signature trajectories to generate a prediction of the solution, thereby completing the system identification and prediction pipeline.

2.6 Dynamical Systems and Control Generation

All the results in the report are tested on two dynamical systems. The first is a cubic polynomial system

$$\dot{x}(t) = x(t)u_1(t) + x^3(t)u_2(t). \tag{9}$$

In control-affine form it can be written as follows

$$\dot{x}(t) = \begin{bmatrix} x(t) & x^3(t) \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \tag{10}$$

The control trajectories $u(t) = [u_1(t), u_2(t)]^T$ generated for that system were largely negative and scaled down by 1000 to avoid blowing up solutions and allow for more interesting dynamics rather than simple fast convergence to zero. The second system is controlled Duffing oscillator

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = u(t) \tag{11}$$

where $\alpha = -1.0$, $\beta = 1.0$ and $\delta = 0.3$. The control-affine form is

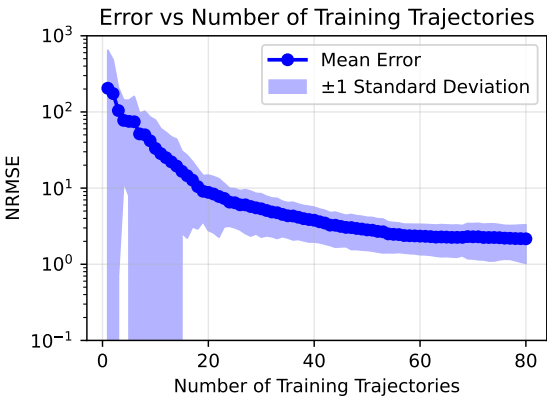
$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \end{bmatrix} = \begin{bmatrix} y(t) & 0 \\ -\delta y(t) - \alpha x(t) - \beta x^3(t) & 1 \end{bmatrix} \begin{bmatrix} 1 \\ u(t) \end{bmatrix} \tag{12}$$

The input trajectories for both systems were generated as sums of $\xi cos(\zeta t)$ and $\xi sin(\zeta t)$ with ξ and ζ drawn from normal distribution.

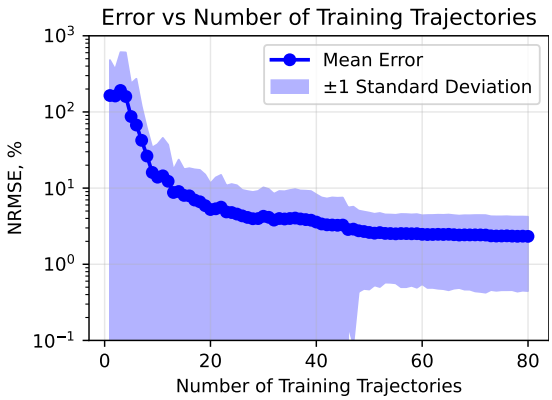
3 Experiments and Results

3.1 Linear Regression Validation

Here, we describe the experimental setup and the results of the linear regression. For both systems, the dataset consisted of 80 different initial conditions sampled from a uniform distribution over the interval $[-5; 5]$, and 160 distinct control trajectories, 2 trajectories for each example of x_0. The dataset was split evenly into training and test sets, each containing 1600 input–output pairs. Note that the training and test sets shared the same 80 initial conditions; only the input trajectories differed. The control signals were sampled over the interval $[0; 1]$ with $\Delta t = 0.01$ for the polynomial system, and over $[0; 0.5]$ with the same step size for the Duffing oscillator. The



(a) Cubic polynomial system



(b) Duffing oscillator

Figure 2: Error against the number of training trajectories

resulting solutions were integrated using a fifth-order Runge–Kutta solver. Then we performed linear regression for $A(x_0)$ from

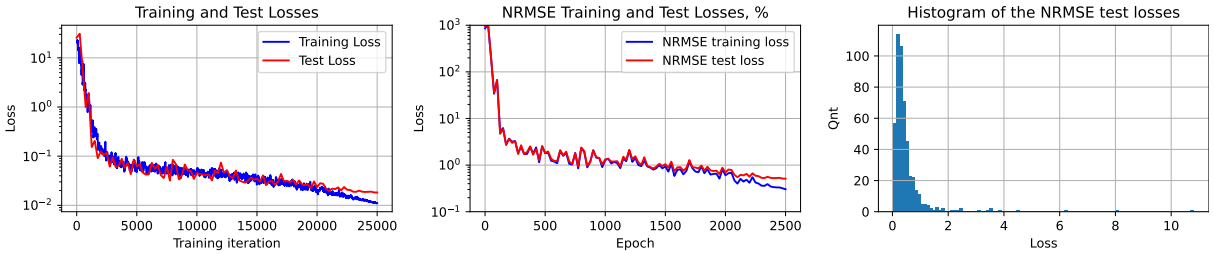
$$\begin{bmatrix} S^K_{[0,t]}(U^1) \\ S^K_{[0,t]}(U^2) \\ \vdots \\ S^K_{[0,t]}(U^i) \end{bmatrix} A(x_0) = \begin{bmatrix} x^1(t) \\ x^2(t) \\ \vdots \\ x^i(t) \end{bmatrix} \tag{13}$$

where i is the number of different control trajectories used in the regression. The obtained coefficients $A(x_0)$ were used on the test set for the corresponding initial conditions and the average NRMSE error across state dimension and all x_0's was calculated. The plot of log_{10} of the mean errors for increasing number of input trajectories used in the regression is given in Figure 2. As expected from the theory, the error decreases, because the coefficients $A(x_0)$ generalize to such that can model dynamics in combination with multiple control trajectories. This result confirms that in principle we are able to do system identification and prediction through the signature transform model.

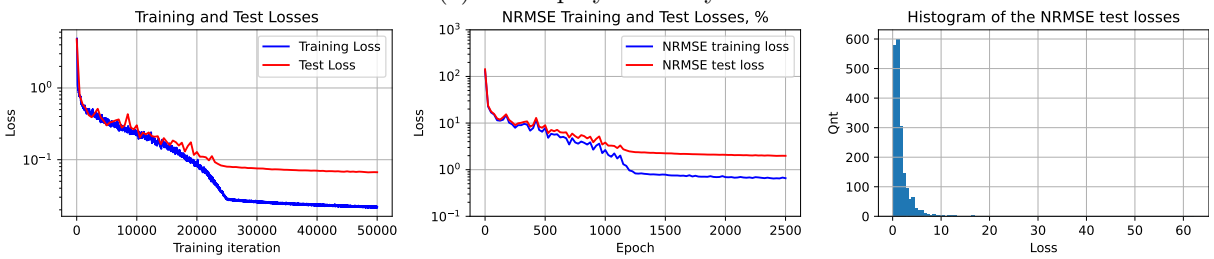
3.2 Neural Network Training

Using linear regression, we demonstrated that it is possible to learn the function $A(x_0)$. Building on this result, we are now ready to train a neural network that maps any initial condition to the corresponding coefficient matrix for a given system — not only those x_0 values seen during training, as in the case of linear regression. This completes the prediction method based on the signature transform model: given any x_0 and $u(t)$, one can predict $x(t)$ using the known signature transform and the coefficient mapping $A(x_0)$.

In practice, however, direct learning of this mapping often fails to produce satisfactory results, as the model's approximation error tends to increase over time (Scampicchio and Zeilinger [2024]). This makes long-horizon prediction challenging and hinders the ability to learn a stable approximation of $A(x_0)$. To enable reliable long-term predictions while still learning the mapping, we trained the neural network on a sequence of short-term predictions. The process works as follows: start from the initial condition x_0 of the target trajectory; generate a short-term prediction of the output; then take the last predicted point as the new x_0 and repeat the procedure. By concatenating these short-term predictions, a long-term trajectory can be reconstructed and compared to the target. This approach also makes the neural network more robust if one wishes to extend the prediction chain beyond the time horizon used during training.



(a) Cubic polynomial system



(b) Duffing oscillator

Figure 3: Summary of training the neural networks

We now move to the specifics of the implementation. We employ feedforward neural networks that take $x_0 \in \mathbb{R}^m$ as input and output $vec(A) \in \mathbb{R}^{m\frac{n^{K+1}-1}{n-1}}$, which is then reshaped into $A(x_0) \in \mathbb{R}^{\frac{n^{K+1}-1}{n-1}\times m}$. Since obtaining the true coefficients from the differential products, as defined in Equation (5), would require explicit knowledge of the system's governing equations—which is rarely feasible in practice—we do not train the network on target coefficient values. Instead, we train it based on how well each predicted coefficient matrix reproduces the solution for a given control trajectory. Our loss function used in training for both systems is Root Mean Square Error (RMSE) normalized by standard deviation of the target solution and given by

$$L(y,\hat{y}) = \sqrt{\frac{\sum_{i=0}^{l}(y_i-\hat{y}_i)^2}{\sum_{i=0}^{l}(y_i-\mu_y)^2}}, \quad \mu_y = \frac{\sum_{i=0}^{l} y_i}{l}. \tag{14}$$

Along with the loss we track the RMSE normalized by the range as given previously. Speaking about the network architectures, the network for the polynomial system consists of 5 linear layers with constant hidden size 256 each followed by the GELU activation function. The network for the Duffing oscillator consists of 6 such linear-GELU layers of the same constant hidden size with an addition of LayerNorm before the output layer. LayerNorm ensures that the outputs do not scale arbitrarily large when the network is at early stages of training, the problem that was absent when training for the polynomial system. The output layers in both networks were initialized at 0 to start the training from a reasonably bounded loss value.

The dataset for the polynomial system consists of 500 training examples (input–output pairs) and an equal number of test examples, sampled from the time interval $[0; 10]$ with $\Delta t = 0.01$. For the Duffing oscillator, larger training and test sets of 1000 examples each were used, sampled from a shorter time interval $[0; 5]$ with the same Δt, as the neural network for this system proved more difficult to train. For both systems, the sampled trajectories were divided into 10 segments each to enable chaining of short-term predictions, as described previously. An important implementation detail is that for the Duffing oscillator, we whitened the matrix of stacked signature trajectories so that all eigenvectors corresponded to the same eigenvalues. Additionally, we applied global per-feature rescaling: for each signature feature, the average norm across the entire training set was computed, and each feature was subsequently divided by its corresponding average norm.

The training in both cases was done in batches of 50 examples with Adam optimizer and linear schedule of the learning rate, from $3 * 10^{-4}$ down to 10^{-6} at the end of the training for the polynomial system, and from 10^{-2} to 10^{-4} for the Duffing oscillator.

The results of training are shown in Figure 3. The first image in a row shows how log_{10} of the internal neural network loss given in Equation (14) progresses with the number of training iterations (one epoch corresponds to 20 training iterations), the second image features the dynamics of the NRMSE error against the number of epochs, and the third images gives histogram of the amount of test trajectories against NRMSE errors in the end of the training. For the polynomial system the training finishes with the average NRMSE error of 0.51 on test set, the Duffing oscillator network reaches 1.98 on the same metric. We consider NRMSE error around 1.0 on test set a success.

4 Discussion

We have demonstrated that the signature transform model is a promising tool for system identification and prediction, and this result paves the way for further research. First, it is now possible to explore the control of dynamical systems using this model. The approach provides a relatively straightforward formulation of the open-loop control problem, which reduces to applying an appropriate optimization method to determine U from Equation (5). Closed-loop control can then be implemented using the Model Predictive Control framework.

Another important direction for future work is reducing the scaling of the model size. In its standard formulation, the dimensionality of the signature grows exponentially with K, the parameter governing the accuracy of the approximation in Equation (5). Consequently, the model quickly becomes computationally infeasible even for small K, leaving limited room for improving performance once the neural network's capacity is reached. This makes it valuable to search for reformulations of the model that can improve its scaling properties.

As a long-term goal, it would also be of great interest to identify other models that factorize the influence of the initial conditions and control on a system's dynamics. Such models combine high interpretability with relatively low computational cost for both training and inference.

5 Conclusion

In conclusion, we have shown that despite its novelty the signature-based approach to dynamics prediction is highly promising, deserving further research to mitigate its drawbacks and explore its control capabilities.

Code Availability

The repository containing the implementation and experimental scripts is available at: `https://github.com/kurolesica0211/signature-based-prediction`.

References

Adeline Fermanian, Pierre Marion, Jean-Philippe Vert, and Gérard Biau. Framing rnn as a kernel method: A neural ode approach. *Advances in Neural Information Processing Systems*, 34:3121–3134, 2021.

F. W. Lewis, S. Jagannathan, and A. Yesildirak. *Neural Network Control of Robot Manipulators and Non-Linear Systems*. CRC Press, 1st edition, 1998. doi: 10.1201/9781003062714. URL `https://doi.org/10.1201/9781003062714`.

Adolfo Perrusquía and Wen Yu. Identification and optimal control of nonlinear systems using recurrent neural networks and reinforcement learning: An overview. *Neurocomputing*, 438: 145–154, 2021. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.01.096. URL https://www.sciencedirect.com/science/article/pii/S0925231221001788.

Naman Agarwal, Daniel Suo, Xinyi Chen, and Elad Hazan. Spectral state space models, 2024. URL https://arxiv.org/abs/2312.06837.

Michael te Vrugt. An introduction to reservoir computing, 2024. URL https://arxiv.org/abs/2412.13212.

Anh Tong. signax, 2022. URL https://github.com/anh-tong/signax.

Anna Scampicchio and Melanie N Zeilinger. Data-driven control of input-affine systems: the role of the signature transform. *arXiv preprint arXiv:2409.05685*, 2024.