Koopman based Data-driven Simulation and Control

Yingzhao Lian†, Renzi Wang†, and Colin N. Jones⋆

Abstract—Sparked by the Willems’ fundamental lemma, a class of data-driven control methods has been developed for LTI system. At the same time, the Koopman operator theory attempts cast a nonlinear control problem into a standard linear one albeit infinite dimensional. Motivated by these two ideas, a data-driven control scheme for nonlinear systems is proposed in this work. The proposed scheme is compatible with most differential regressor enabling an offline learning. In particular, the model uncertainty is considered, enabling a novel data-driven simulation framework based on Wasserstein distance. Numerical experiments are performed with Bayesian neural networks to show the effectiveness of both the proposed control and simulation scheme.

Index Terms—Predictive control, data-driven control, Koopman operator

I. INTRODUCTION

RECENT trend of the digitalization motivates the research interest of data-driven control [1] because of the wide access of data collected by the sensors. Instead of resorting to a first-principle model, the collected data are used either to identify a model [2] or to construct a controller directly. The former approach is compatible with most control theory and therefore results in successful applications [3], [4]. Featuring a controller design without any intermediate stage, the latter data-driven scheme attract more research interests and finds successful application in linear systems [5] and iterative control [6], [7]. It worth mentioning that in the community of reinforcement learning [8], the learning schemes can also be categorized as model-based methods and model-free methods.

This work is developed on the basis of the Willems’ fundamental lemma and the Koopman operator theory. In particular, the Willems’ fundamental lemma characterizes the responses of deterministic linear time invariant (LTI) systems with measured trajectories under reasonable assumptions of controllability and persistent excitation. Based on the data-driven prediction enabled this lemma, predictive control scheme has been developed [9], [10]. Beyond the predictive control, the Willems’ fundamental lemma has also been adopted in feedback controller design [11], [12]. Within the LTI framework, the Willems’ fundamental lemma is further extend to incorporate measurement noise [13], [14] and process noise [11].

At the same time, a significant collection of works has tried to extend the applications of the Willems’ fundamental lemma to nonlinear systems. In [15], an extension to Hammerstein systems and Wiener systems is proposed based on an a priori knowledge of basis function. [16]–[18] attempt to apply the Willems’ fundamental lemma to the class of polynomial system. As pointed out in [17], the necessary and sufficient condition in the fundamental lemma is broken. An promising viewpoint regarding the quotient space is proposed in [19], which clusters trajectories into equivalent class and extend the Willems’ fundamental lemma into the reproducing kernel Hilbert space. The functional space viewpoint in [19] motivates us to apply the Willems’ fundamental lemma in the function space. In particular, the Koopman operator theory is used.

The Koopman operator theory is first introduced in the study of forward-complete autonomous systems [20], [21], which is a linear composite operator even when the system is nonlinear. [22], [23] later introduced the applications of the Koopman operator in controller and observer design, followed by a wide range of research ranging from the model reduction [24] to the global optimal control [25]. Even though most algorithms based on the Koopman operator have numerical implementations that are similar to those studied in [15]–[18], the Koopman operator establishes a totally different theoretical framework. In particular, the Koopman operator corresponds to a Heisenberg picture which models the evolution of the observable, while other aforementioned methods model the evolution of the state, corresponding to a Schrödinger picture [26]. The Koopman operator theory can alleviate the theoretical issue in studying the lifting function without resorting to the quotient space, and enables convergence analysis as that has done in [27].

The key component of a Koopman operator based method is the learning of the eigenfunctions or the lifting functions lying within the subspace spanned by the eigenfunctions. A standard framework of extended dynamic mode decomposition (EDMD) spans the lifting functions with a dictionary of basis function [28], which suffers from the curse of dimensionality. To overcome this challenge, [29], [30] applies kernel method to learn the Koopman operator in a non-parametric way, which is still not scalable to large dataset. In [31], [32], the lifting functions are approximated by neural networks. However, these aforementioned methods mainly consider one-step forward prediction either due to the formulation of the
learning problem or due to numerical stability, which results in a relatively inaccurate long-term prediction. A link between the Koopman operator and the subspace identification is observed in [33], which enables a learning scheme of long-term prediction. However, this method still suffers from the lack of scalability and the numerical limitation of the subspace identification [34].

In this work, we propose to incorporate the learning of a Koopman operator into the framework of the Willems’ fundamental lemma. The applications of the Koopman operator in the Willems’ fundamental lemma are mentioned in [9], [15] but have not been detailed. In this work, we show that by maximizing the linearity of an finite order approximation, a Koopman operator can be learned based on the sensitivity analysis of a parametric programming problem. The proposed learning scheme is capable of uncertainty quantification, where a new objective function is derived to account for the uncertainty. Meanwhile, we propose a control scheme that solves a bi-level optimization problem by a transformation into to a single level structure. The bi-level optimization formulation has been discussed in both [9], [35], where a bi-level problem is relaxed to a multi-objective problem.

The remainder of this paper is organized as follows: the preliminary knowledge is introduced in Section II, after which the framework in Section III along with presenting the numerical simulation results of prediction and control in Section V. We explain the proposed control scheme in both [9], [35], where a bi-level problem is relaxed to a single level optimization problem by a transformation into to a single level structure. The bi-level optimization problem is solved in [9], [35], where a bi-level problem is relaxed to a multi-objective problem. Meanwhile, the subscript \( k \) := \( w \) is reserved to denote the data collected offline. The superscript \( d \) is used to denote optimal solution of an optimization problem. \( \otimes \) denotes the Kronecker product.

## NOTATIONS

\( \| x \|_p \) indicates the \( \ell_p \) norm of vector \( x \) and \( \| x \|_Q := x^\top Q x \) is the weighted norm with \( Q \) being positive semi-definite. \( \| A \|_F \) and \( \| A \|_1 \) denote the Frobenius norm and the nuclear norm of the matrix \( A \) respectively. \( N \sim (\mu, \Sigma) \) is a Gaussian distribution with mean value \( \mu \) and covariance matrix \( \Sigma \). We use \( \mathbb{Z}_{\geq 0} \) to represent a non-negative integer. \( \mathbf{w} := \{ w_k \}_{k=a} \) is a sequence of signal \( \{ w_a, \ldots, w_k \} \) indexed by \( k \). Specifically, the boldface is used to denote a sequence while the lightface denotes a measurement, e.g. \( \mathbf{w} \) and \( w_k \). Meanwhile, the subscript \( d \) is reserved to denote the data collected offline. The superscript * is used to denote optimal solution of an optimization problem. \( \otimes \) denotes the Kronecker product.

## II. PRELIMINARY

In this section, the Willems’ fundamental lemma and the Koopman operator theory will be introduced. Then, the sensitivity analysis of a parametric optimization, which is the enabler of the learning, is discussed.

### A. Willems’ Fundamental Lemma

Given a sequence of measurements \( \{ w_k \}_{k=0}^{T-1} \), its Hankel matrix of depth \( L \) is defined as

\[
H_L(\mathbf{w}) := \begin{bmatrix}
w_0 & w_1 & \ldots & w_{T-L} \\
w_1 & w_2 & \ldots & w_{T-L+1} \\
\vdots & \vdots & \ddots & \vdots \\
w_{L-1} & w_{L} & \ldots & w_{T-1}
\end{bmatrix}.
\]

Regarding a Hankel matrix \( H_L(\mathbf{w}) \), the signal sequence \( \mathbf{w} \) is persistently exciting of order \( L \) if \( H_L(\mathbf{w}) \) is full row rank. The Willems’ fundamental lemma utilizes the Hankel matrices to characterize the response of the following deterministic linear time invariant (LTI) system, dubbed \( \mathcal{B}(A, B, C, D) \),

\[
x_{k+1} = Ax_k + Bu_k,
\]

\[
y_k = Cx_k + Du_k,
\]

where \( A \in \mathbb{R}^{n_x \times n_x}, B \in \mathbb{R}^{n_x \times n_u}, C \in \mathbb{R}^{n_y \times n_x}, D \in \mathbb{R}^{n_y \times n_u} \) parametrize the system dynamics and the order of this system is denoted by \( O(\mathcal{B}(A, B, C, D)) := n_x \). The Willems’ fundamental lemma is concluded as

**Lemma 1:** ( [36, Theorem 1], [12, Lemma 2]) Consider a controllable and observable system \( \mathcal{B} \), if the input sequence \( \mathbf{u}_d = \{ u_{d,k} \}_{k=0}^{T_{\text{ini}}-1} \) is persistently exciting of order \( O(\mathcal{B}(A, B, C, D)) + L \), then

1. Any L-step input/output trajectory of system \( \mathcal{B} \) can be expressed as

\[
\begin{bmatrix}
H_L(\mathbf{u}_d) \\
H_L(\mathbf{y}_d)
\end{bmatrix} g = \begin{bmatrix} u \\ y \end{bmatrix},
\]

2. Any linear combination of the columns of the Hankel matrices, that is

\[
\begin{bmatrix}
H_L(\mathbf{u}_d) \\
H_L(\mathbf{y}_d)
\end{bmatrix} g
\]

is a L-step input/output trajectory of \( \mathcal{B} \).

This lemma enables data-driven simulation and control [9], [37]. To make an N-step prediction, the Hankel matrices composed of offline data is partitioned as

\[
\begin{bmatrix}
U_p \\
U_f
\end{bmatrix} := H_{T_{\text{ini}}+N}(\mathbf{u}_d),
\]

\[
\begin{bmatrix}
Y_p \\
Y_f
\end{bmatrix} := H_{T_{\text{ini}}+N}(\mathbf{y}_d),
\]

where the first \( T_{\text{ini}} \) row blocks are used to construct \( U_p, Y_p \) while the remaining row blocks is assigned to \( U_f, Y_f \). In the remainder of this paper, \( n_e \) is reserved to denote the number of columns in the Hankel matrix. In particular, \( T_{\text{ini}} \) is chosen to ensure the uniqueness of prediction and the rank of the observability matrix

\[
\mathcal{O}_{T_{\text{ini}}}(A, C) := \begin{bmatrix} C^T & (CA)^T & \ldots & (CA^{T_{\text{ini}}-1})^T \end{bmatrix}^T
\]

is of rank \( O(\mathcal{B}(A, B, C, D)) = n_x \) [37]. Without measurement noise, the N-step output prediction \( \mathbf{y} \) is defined by

\[
\begin{bmatrix}
U_p \\
Y_p \\
U_f
\end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{ini}} \\ \mathbf{y}_{\text{ini}} \\ \mathbf{u} \end{bmatrix},
\]

where \( \mathbf{u}_{\text{ini}} \) and \( \mathbf{y}_{\text{ini}} \) are \( T_{\text{ini}} \)-step previous measurements of the inputs and the outputs. Accordingly, \( \mathbf{y} \) is the N-step response driven by input sequence \( \mathbf{u} \). Built on this prediction
scheme, the data-enabled predictive control (DeepPC) [9] is
\[
\min_{g, \sigma_g, u, y} \left( \sum_{k=0}^{N-1} \| y_k - r_{t+k} \|_Q + \| u_k \|_R^2 \right)
+ \lambda_g \| g \|_1 + \lambda_y \| \sigma_y \|_1
\]
s.t.
\[
\begin{bmatrix}
U_p \\
U_f
\end{bmatrix} g = \begin{bmatrix}
u_{\text{ini}} \\
y_{\text{ini}}
\end{bmatrix} + \sigma_y
\]
\[
u_k \in \mathcal{U}, \forall k \in 0, \ldots, N - 1
\]
\[
y_k \in \mathcal{Y}, \forall k \in 0, \ldots, N - 1
\],

where \(Q\) and \(R\) are the weight penalizing outputs and inputs respectively and \(\sigma_g\) is introduced to deal with measurement noise. \(r \in \mathbb{R}^{pN}\) is the reference trajectory and \(\mathcal{U}, \mathcal{Y}\) are the feasible sets of inputs and outputs. \(||g||_1\) and \(||\sigma_y||_1\) are regularization terms. \(\lambda_y, \lambda_g \in \mathbb{R}_{>0}\) are regularization parameters. These regularization terms have been interpreted under distributionally robust optimization framework [38] and maximal likelihood framework [14].

**Remark:** When a long input-output sequence is not available, the Hankel matrix \(H_L(\mathbf{w})\) can be replaced by a mosaic Hankel matrix [39]. Given \(M\) trajectories:
\[
\mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_M],
\]
where each trajectory \(\mathbf{w}_i = (u_{i,1}, \ldots, u_{i,T})\), \(u_{i,k} \in \mathbb{R}^q\) the mosaic Hankel matrix is defined as:
\[
H_L(\mathbf{w}) = [H_L(\mathbf{w}_1), \ldots, H_L(\mathbf{w}_M)].
\]

**B. Koopman Operator**

Given a discrete-time autonomous system
\[
x_{k+1} = f(x_k),
\]
where \(f\) models the nonlinear dynamics, a Koopman operator is a composite operator
\[
\mathcal{K} \psi := \psi \circ f,
\]
which \(\psi : \mathbb{R}^{nx} \to \mathbb{R}\) is called observable. Unlike standard state space model, the Koopman operator models the evolution of a function driven by system dynamics \(f\) and its existence is guaranteed for forward-complete system [40]. As the Koopman operator is an operator on a function space, \(\mathcal{K}\) is in general infinite-dimensional, but critically it is linear even when the dynamics \(F\) are non-linear and as such, an observable \(\phi\) is an eigenfunction associated with the eigenvalue \(\lambda \in \mathbb{C}\) if \(\mathcal{K} \phi = \lambda \phi\). From this we can see that the eigenfunctions (or linear combinations of the eigenfunctions) evolve linearly along the trajectories of our nonlinear system [5]

\[
\phi(x_{k+1}) = \phi(f(x_k)) = (\mathcal{K} \phi)(x_k) = \lambda \phi(x_k).
\]

Given a collection of eigenfunctions \(\{\phi_i\}_{i=1}^{n_\phi}\), any observable lying within the span of these eigenfunctions can be decomposed into \(\psi = \sum_i c_i(\psi)\phi_i\), where \(c_k(\psi)\) is called the Koopman modes of \(\psi\). Then, we have
\[
\mathcal{K} \psi = \sum_i c_i(\psi)\lambda_i \phi_i,
\]
with \(\lambda_i\) denoting the eigenvalue of \(\phi_i\).

In the sequel, the subscript \(a\) is used to denote the components corresponding to a system with control inputs. Given a nonlinear dynamics with control input
\[
x_{k+1} = f_u(x_k, u_k),
\]
the Koopman operator can be defined in different ways [22], [41], [42]. In this work, we consider the framework in [22]. More specifically, denote the infinite control sequence \(u := \{u_k\}_{k=0}^\infty \in l(\mathcal{U})\), where \(l(\mathcal{U})\) represents the space of all control sequence. The augmented state is
\[
\chi = \begin{bmatrix} x \\ u \end{bmatrix},
\]
upon which the system dynamic is augmented as
\[
F(\chi_k) := \begin{bmatrix} f_u(x_k, u_k(0)) \\ Su_k \end{bmatrix},
\]
where \(S\) is the left shift operator with \(Su(i) := u(i + 1)\) and \(u(0)\) is the evaluation of the \(i\)-th element of \(u\). In this setup, \(u\) can be considered as a sequence of mappings from index \(i\) to actual output \(u_i\). It is noteworthy to point out that this dynamical system [7] is infinite dimensional but autonomous. Hence, the aforementioned definition of the Koopman operator can be applied directly and the corresponding eigenfunctions are assumed to be spanned by the following dictionary of basis functions.

\[
\{ \phi_u(x, u)^{(n_u)_1}_{i=1} : \phi_{u,1}(x), \ldots, \phi_{u,n_u}(x), u(0) \}.
\]

If the evolution of this dictionary of basis functions is closed under the system dynamics, then we have
\[
z_{k+1} = A z_k + B u_k(0),
\]
\[
u_k(0) = u_k(1),
\]
where \(z := [\phi_{u,1}(x_k), \ldots, \phi_{u,n_u}(x_k)]\) and \(A, B\) captures the Koopman operator. Similarly, any functions within the span of these basis functions can be recovered by the Koopman mode as
\[
\psi_u(x, u(0)) = c_u^T \begin{bmatrix} z \\ u(0) \end{bmatrix},
\]
with \(c := [c_{u,1}, c_{u,2}, \ldots, c_{u,n_u+n_u}]\) the vector of Koopman modes. In particular, we are interested in the Koopman mode of the identity functions evaluated on the system outputs. Assumed that we have \(n_y\) outputs, the evolution of the \(i\)-th output is
\[
I_{y,i}(y_k) := y_{k,i}.
\]
With a bit of abuse of notation, the Koopman modes decomposition of outputs evaluation is
\[
y_k = \begin{bmatrix} I_{y_1}(y_k) \\ I_{y_2}(y_k) \\ \vdots \\ I_{y,n_y}(y_k) \end{bmatrix} = \begin{bmatrix} c_{u,1} \\ c_{u,2} \\ \vdots \\ c_{u,n_y} \end{bmatrix} \begin{bmatrix} z \\ u(0) \end{bmatrix} := C_u \begin{bmatrix} z \\ u(0) \end{bmatrix},
\]
where \(C_u\) stacks the Koopman modes of the output evaluations.
C. Differential Parametric Optimization

Sensitivity analysis investigates the smoothness of a parametric optimization problem, where the implicit function theorem [43] is applied to the KKT system. This idea has been applied to deep learning [44] and reinforcement learning [45]. Though the solution map is barely differentiable, the optimal value function is smoother than the solution map [46], which is the only tool used in this work. In general, the continuity of a general convex optimization problem is guaranteed by the uniform boundedness [47, Theorem 1.17], while a general nonlinear parametric optimization problem guarantees a lower semicontinuous value function under the assumption of local compactness [48].

For the sake of clarity, we elaborate this derivative by a standard quadratic programming (QP), please refer to [49] for a general conic form. We use subscript $q$ to avoid confusion. Consider the a parametric QP, $Q(e_q) := e_q \to z_q^*$ with parameters $\{Q_q, q_0, H_q, h_q, E_q\}$ and $Q_q$ positive definite:

$$\min \frac{1}{2} z_q^T Q_q z_q + q_q^T z_q$$

s.t. $H z_q \leq h_q, E z_q = e_q$  \hspace{1cm} (11)

The KKT conditions for the QP are:

$$Q z^* + q_q + H_q^T \lambda^* + E_q^T \nu^* = 0$$

$$\text{diag}(\lambda^*) (H_q z^* - h_q) = 0$$

$$E z^* - e_q = 0$$

$$\lambda^*, \nu^*, \lambda^*$$ are the optimal primal and dual variables, $\text{diag}(x)$ builds a diagonal matrix composed of $x$. Then the derivatives of KKT conditions can be computed as:

$$\begin{bmatrix} Q_q & H_q^T \\ D(\lambda^*) \lambda^* & E_q^T \\ E_q \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} dz_q \\ d\lambda^* \\ dE_q \\ 0 \end{bmatrix} = \begin{bmatrix} dQ_q z^* + dq_q + dH_q \lambda^* + dE_q^T \nu^* \\ \text{diag}(\lambda^*) \text{diag}(H_q z^* - h_q) \\ \text{diag}(\lambda^*) \text{diag}(\lambda^*) \\ dE_q z^* - de_q \end{bmatrix}$$

The derivatives of $z^*$ with respect to the parameters $(Q_q, q_0, H_q, h_q, E_q)$ and the function input $f$ are given by the solution to the linear system defined in Equation (13). For example, the solution $dz$ gives the result of $\frac{\partial z^*}{\partial Q_q}$ if we set $dQ_q = I$ and the differentials of other parameters to 0. The gradient of optimal value $L(z^*)$ with respect to $Q$ is calculated accordingly as $\frac{\partial L(z^*)}{\partial Q_q} = \frac{\partial z^*}{\partial Q_q}$.

III. Koopman based Data-driven Prediction

In this section, the fundamental lemma is first introduced in the Koopman operator theory, which enables a training scheme by minimizing the prediction error with respect to the training dataset. The stochastic prediction scheme is thereby introduced to show the compatibility of probabilistic models, such as Bayesian neural networks and Gaussian processes.

A. Koopman Operator with the Fundamental Lemma

As discussed in Section II the key component of a Koopman operator is the eigenfunctions or the linear subspace containing the eigenfunctions. Therefore, the learning of an Koopman operator is equivalent to find functions whose evolution of the function evaluation behaves like a linear system. Following corollary is the enabler of the proposed learning scheme

**Corollary 1:** An dynamical system of order $n_x$ can be parametrized as a linear system if and only if the Fundamental lemma holds.

**Proof:** Necessary condition holds by Lemma 1 and the sufficient condition holds by the definition of linear systems.

As discussed in Section II-B the outputs evaluations are assumed to be spanned by the following basis functions:

$$\{\phi_{u_i}(x, u)\}_{i=1}^{n_u} := \{\phi_{u_1}(x), \ldots, \phi_{u_{n_u}}(x), u(0)^T\}$$

Then given a sequence of state evolution $x_k$ with its corresponding inputs-output sequence $u_k, y_k$, whose inputs are persistently excited of order $n_u - n_q + L$, Corollary 1 implies that $\{\phi_{u_i}(x, u)\}_{i=1}^{n_u + L}$ is the desired collection basis functions if and only if $\forall x \in \mathbb{R}^{n_x}$, the outputs sequence $y$ driven by $u$, there exist $g \in \mathbb{R}^{n_y}$:

$$\begin{bmatrix} Z \\ H_L(u_k) \\ H_L(y_k) \end{bmatrix} = \begin{bmatrix} z \\ u \\ y \end{bmatrix},$$

where $z := [\phi_{u_1}(x), \ldots, \phi_{u_{n_u}}(x)]^T$ and

$$Z := \begin{bmatrix} \phi_{u_1}(x_0) & \phi_{u_1}(x_1) & \cdots & \phi_{u_1}(x_{n_y}) \\ \vdots & \ddots & \vdots & \vdots \\ \phi_{u_{n_u}}(x_0) & \phi_{u_{n_u}}(x_1) & \cdots & \phi_{u_{n_u}}(x_{n_y}) \end{bmatrix}$$

B. Leaning the Koopman Basis Functions

Due to the previous discussion, the learning of a Koopman operator is converted to learn basis functions that maximizes the satisfaction of the condition. In practice the underlying state for the nonlinear system is not necessarily measured, we therefore make the following assumption

**Assumption 1:** $x_k$ is measurable with respect to the previous $T_{ini}$ step input-output sequence $\{u_i, y_i\}_{i=k-T_{ini}+1}^k$. This assumption implies that $x_k$ can be determined from $u_i, y_i$ and therefore has similar utilization as the matrices $U_p, Y_p$ in problem 4 and 8. Assumed that we have a sequence of input-output data $u_d := \{u_{d,i}\}_{i=0}^{n_u}$ and $y_d := \{y_{d,i}\}_{i=0}^{n_d}$ consisting $n_d$ measurements, each of them is partitioned into two subsets, including $u_{d,:} := \{u_{d,i}\}_{i=0}^{n_d}$, $y_{d,:} := \{y_{d,i}\}_{i=0}^{n_d}$, $u_{d,:} := \{u_{d,i}\}_{i=n_d+1}^{n_d}$ and $y_{d,:} := \{y_{d,i}\}_{i=n_d+1}^{n_d}$, $n_{d,t} = n_d + T_{ini} + L - 1$ is the number of datapoints in the first two sets. The subsets with subscript $d,t$ are used to build the Hankel matrices characterizing the Koopman operator while the remaining two subsets are used to learn the basis functions.

Regarding the Assumption 1 a differentiable learner is used to learn the basis functions, dubbed $\{\phi_{u, \theta}\}_{i=1}^{n_u}$, whose parameters are denoted by $\theta$. Neural networks [50] and
Gaussian process [51] are recommended learners that have strong representation power. In particular, inducing variables can be considered as trainable parameters for a Gaussian process, please refer to [52], [53] for more details. Enforcing the condition (14) for L-step sequences, learning problem is formulated as follows:

$$
\min_{\theta} \sum_{i=n_c}^{n_d} \lambda_l (\| \theta \|_2^2 + \lambda_y \| Zg - z \|_2^2)
$$

s.t.

$$
\lambda_l = \arg \min_{\theta} P(\mathbf{u}_{d,i}, \mathbf{y}_{d,i}),
$$

and

$$
P(\mathbf{u}_{d,i}, \mathbf{y}_{d,i}) := \lambda_g \| g \|_2^2 + \lambda_y \| Zg - z \|_2^2
$$

s.t.

$$
H_L(\mathbf{u}_{d,i}) = \mathbf{u}_{d,i}.
$$

$$
z = \phi_u, \theta(\{u_k, y_k\}_{k=1}^{l+T_{ini}-1}).
$$

In particular, \( \mathbf{u}_{d,i} := [u_i, \ldots, u_{i+L+T_{ini}-1}] \) and \( \mathbf{y}_{d,i} := [y_i, \ldots, y_{i+L+T_{ini}-1}], l \geq n_d, l + 1 \) are sequences of inputs and outputs of length \( L + T_{ini} \). The matrix \( Z \) is the evaluation of the basis functions

$$
Z := [\phi_u, \theta(\{u_1, y_1\}_{k=1}^{T_{ini}}), \ldots, \phi_u, \theta(\{u_1, y_1\}_{k=n_d-1})].
$$

The constraint in the learning problem (15) is actually a prediction problem similar to (3) and, therefore, \( l(\cdot) \) penalizes the prediction error. As one may notice, there are two relaxations in the learning problem (15).

1) To recover an output evaluation, an infinite set of basis functions may be required. This learning problem learns a finite order approximation of these probably infinite set.

2) The condition (14) is required to be satisfied for any states, however, the learning problem relax this condition to a set of sampled states. Therefore, the training set \( \mathbf{u}_{d,i} \) and \( \mathbf{y}_{d,i} \) should be large enough to represent the condition (14).

Figure 1 shows the flow of the learning problem, where the dashed line indicates the direction of back-propagation. More specifically, the prediction problem is considered as a parametric optimization problem whose differentiation is discussed in Section II-C.

Finally, we end this subsection by further listing the benefits of the proposed scheme.

- Unlike general EDMD methods, which learns the matrices \( A, B, C_u \) in (8) and (10), the proposed scheme get rid of the learning of the parameters. Learning of \( A, B, C_u \) is ill-conditioned because the solution is not unique.

- The learning problem optimizes a multi-step forward prediction, meanwhile the utilization of the Willems’ fundamental lemma guarantees a good numerical stability in the training scheme, which is a key challenge in training the fully-connected recurrent neural network [54].

- The proposed scheme is scalable and can be parallelized.

- Unlike other nonlinear extension of the Willems’ fundamental lemma, the proposed scheme does not require nonlinear mapping of future inputs and outputs in the prediction problem.

### C. Stochastic Prediction

As shown in the Section III-B, the prediction problem plays a key role in the proposed scheme. If the chosen learner is deterministic, then the predicted output sequence \( \tilde{y} \) driven by inputs \( \bar{u} \) is calculated by

$$
\tilde{y} = H_L(\mathbf{y}_{d,i}) \hat{g}
$$

$$
\hat{g} = \arg \min_{\theta} \lambda_g \| g \|_2^2 + \lambda_y \| Zg - z \|_2^2
$$

s.t.

$$
H_L(\mathbf{u}_{d,i}) = \mathbf{u}_{d,i}.
$$

$$
z = \phi_u, \theta(\{u_{d,i}, \mathbf{y}_{d,i}\}_{k=1}^{l+T_{ini}-1}).
$$

which mainly results in overfitting. Probabilistic learner is one solution to avoid overfitting, such as the aforementioned Gaussian process and the Bayesian neural networks. The output of a probabilistic learner is a distribution but not a deterministic point. In this section, we will show how this distributional outputs from a probabilistic learner can be used for prediction, which is also essential for the training of the Koopman operator. Two methods will be discussed, one is based on Monte-Carlo sampling while another one generates prediction by bounding the Wasserstein distance.

#### 1) Monte-Carlo Prediction

Assuming the distribution of the probabilistic learner is \( \mathbb{P}_{\phi_u} \), a Monte-Carlo method is applied to calculate the distribution of the prediction. In particular, the matrix \( Z \) in problem (16) is sampled from \( \mathbb{P}_{\phi_u} \), which gives a sample from the output distribution. By Monte-Carlo method, the output distribution can be approximated by the sampled outputs.

Meanwhile, the loss function in the learning problem (15) is modified to expected cost. In conclusion, we have the following learning problem

$$
\min_{\theta} \sum_{i=n_c}^{n_d} \mathbb{E} l(\mathbf{y}_{d,i} - H_L(\mathbf{y}_{d,i})g_i)
$$

s.t.

$$
g_i \sim \arg \min_{\theta} P(\mathbf{u}_{d,i}, \mathbf{y}_{d,i}),
$$

and

$$
P(\mathbf{u}_{d,i}, \mathbf{y}_{d,i}) := \lambda_g \| g \|_2^2 + \lambda_y \| Zg - z \|_2^2
$$

s.t.

$$
H_L(\mathbf{u}_{d,i}) = \mathbf{u}_{d,i}.
$$

$$
z \sim \mathbb{P}_{\phi_u, \theta}(\{u_k, y_k\}_{k=1}^{l+T_{ini}-1}).
$$

The \( k \)-th column of the matrix \( Z \) follows the distribution \( \mathbb{P}_{\phi_u, \theta}(\{u_k, y_k\}_{k=1}^{l+T_{ini}-1}) \). The gradient of this learning problem is also approximated by a Monte-Carlo method.

#### 2) Wasserstein Distanced based Prediction

Intuitively, the regularization term \( \| Zg - z \|_2^2 \) in the prediction problem (16) can be considered as the distance between the mean of \( Zg \) and \( z \). To formulate a more rigor scheme based on a probabilistic learner, we propose to minimize the Wasserstein distance between \( Zg \) and \( z \). Above all, the entry of the probabilistic learner output is approximated by a Gaussian distribution. Specifically, the \( i \)-th column of the matrix \( Z \) is approximated
by \( \mathcal{N}(\mu_{i,l}, \Sigma_{i,l}) \), whose covariance matrix is diagonal
\[
\Sigma_{i,l} = \begin{bmatrix}
\sigma^2_{i,1,l} & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \sigma^2_{i,n_{o,l}}
\end{bmatrix},
\]
and we denotes the vector composed of the diagonal elements as \( \sigma^2_{i,l} \). Accordingly, the \( i \)-th element of vector \( z \) is approximated by \( \mathcal{N}(\mu_{i}, \sigma_{z}) \) and we denote \( z \sim \mathcal{N}(\mu_{i}, \sigma_{z}) \).

**Remark 1:** • It is noteworthy that if an Gaussian process is used to learn the basis function, no approximation is required as the output is already Gaussian.
• To better satisfy the condition of a diagonal covariance matrix \( \Sigma_{i,l} \), it is recommended to replace the Hankel matrices with Page matrices. In comparison with definition in [I], a depth \( L \) Page matrix of a sequence \( \mathbf{w} \) is defined as
\[
\Psi_{L}(\mathbf{w}) := \begin{bmatrix}
w_0 & w_L & \cdots & w_{(M-1)L} \\
w_1 & w_{L+1} & \cdots & w_{(M-1)L+1} \\
\vdots & \vdots & \ddots & \vdots \\
w_{L-1} & w_{L+1} & \cdots & w_{ML-1}
\end{bmatrix}.
\]

Based on the approximation, \( Zg \) turns out to be a Gaussian distribution, which is denoted by \( Zg \sim \mathcal{N}(\mu_{Zg}, \Sigma_{Zg}) \) for compactness. To enable a prediction scheme, we conclude the following lemma

**Lemma 2:** The second Wasserstein distance between \( Zg \) and \( z \) is bounded by
\[
W_{2}^{2}(Zg, z) \leq \|\mu_{Zg} - \mu_{z}\|_{2}^{2} + \|\Sigma_{Zg} - \Sigma_{z}\|_{*}.
\]

**Proof:** The second Wasserstein distance between two Gaussian distribution [55] is
\[
W_{2}^{2}(\mathcal{N}(\mu_{Zg}, \Sigma_{Zg}), \mathcal{N}(\mu_{z}, \Sigma_{z})) = \|\mu_{Zg} - \mu_{z}\|_{2}^{2} + \text{Tr}(\Sigma_{Zg} + \Sigma_{z}) - 2\text{Tr}(\Sigma_{Zg} \Sigma_{z}^{1/2}),
\]
where \( \mu_{Zg} = \sum_{i=1}^{n_{c}} \mu_{i}g_{i} \) with \( g_{i} \) being the \( i \)-th entry of \( g \). The covariance matrix of \( Zg \) is calculated as \( \Sigma_{Zg} = (g^{T} \otimes I)\text{Cov}(\text{vec}(Z))(g \otimes I) \), with
\[
\text{Cov}(\text{vec}(Z)) = \begin{bmatrix}
\Sigma_{1,l} & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \Sigma_{n_{o,l}}
\end{bmatrix},
\]
therefore, we conclude
\[
\Sigma_{Zg} = \sum_{i=1}^{n_{c}} g_{2}^{2} \Sigma_{i,l}.
\]

Since \( \Sigma_{z} \) is diagonal, \( \Sigma_{Zg} \Sigma_{z} = \Sigma_{z} \Sigma_{Zg} \), (18) can be reformulated as:
\[
\begin{align*}
W_{2}^{2}(\mathcal{N}(\mu_{Zg}, \Sigma_{Zg}), \mathcal{N}(\mu_{z}, \Sigma_{z})) &= \|\mu_{Zg} - \mu_{z}\|_{2}^{2} + \text{Tr}(\Sigma_{Zg} + \Sigma_{z} - 2(\Sigma_{Zg} \Sigma_{z})^{1/2}) \\
&= \|\mu_{Zg} - \mu_{z}\|_{2}^{2} + \text{Tr}((\Sigma_{Zg}^{1/2} - \Sigma_{z}^{1/2})(\Sigma_{Zg}^{1/2} - \Sigma_{z}^{1/2})) \\
&= \|\mu_{Zg} - \mu_{z}\|_{2}^{2} + \|\Sigma_{Zg}^{1/2} - \Sigma_{z}^{1/2}\|_{F}^{2},
\end{align*}
\]
where \( \|\cdot\|_{F} \) denotes the Frobenius norm. This objective function has a clear interpretation. The first term quantifies the distance between the mean of these two Gaussian distribution, while the second measures the discrepancy between the covariance matrices. If the derived \( \Sigma_{Zg} \) in (19) is substituted in (20), the evaluation of the resulting metric is numerically ill-conditioned. The Frobenius norm is therefore further relaxed with the Powers-Størmer’s inequality [56]:
\[
2\text{Tr}(A^{T}B^{1-\alpha}) \geq \text{Tr}(A + B - |A - B|), 0 \leq \alpha \leq 1
\]
(21)

A, B are positive semidefinite and \( |A| \) is the positive square root of the matrix \( A^{T}A \), we have:
\[
\begin{align*}
W_{2}^{2}(\mathcal{N}(\mu_{Zg}, \Sigma_{Zg}), \mathcal{N}(\mu_{z}, \Sigma_{z})) & \leq \|\mu_{Zg} - \mu_{z}\|_{2}^{2} + \|\Sigma_{Zg} - \Sigma_{z}\|_{*}, \\
&= \|\mu_{Zg} - \mu_{z}\|_{2}^{2} + \|\Sigma_{Zg} - \Sigma_{z}\|_{*},
\end{align*}
\]
with \( \|\cdot\|_{*} \) denoting the nuclear norm.

Based on this lemma, the prediction problem with a probabilistic learner is reformulated as
\[
\begin{align*}
\hat{y} &= H_{L}(y_{d,l})\hat{g} \\
\hat{g} &= \arg\min_{g} \|\mu_{Zg} - \mu_{z}\|_{2}^{2} + \|\Sigma_{Zg} - \Sigma_{z}\|_{*} \\
\text{s.t. } H_{L}(u_{d,l})g &= \tilde{u} \\
z &= \phi_{u,\theta}(\tilde{u}, \tilde{y}_{p}),
\end{align*}
\]

**Remark 2:** The upper bound (17) is non-smooth, where the absolute value evaluation is ill-conditioned around 0 [57, Chapter 3]. The absolute value is smoothed by an approach similar to a Huber loss [47, Chapter 2], which is defined as:
\[
L_{\delta}(a) = \begin{cases}
\frac{1}{2}a^{2} & \text{for } |a| < \delta \\
\delta(|a| - \frac{1}{2}\delta) & \text{otherwise}
\end{cases}
\]

The evaluation of the \( i \)-th diagonal elements in \( \|\Sigma_{Zg} - \Sigma_{z}\|_{*} \) is then reformulated as
\[
\begin{cases}
|\Sigma_{Zg} - \Sigma_{z}|_{ii}^{2} & \text{if } |\Sigma_{Zg} - \Sigma_{z}|_{ii} < \delta \\
\frac{1}{2}(\sigma^{2}_{Zg,i} - \sigma^{2}_{z,i})^{2} & \text{otherwise}
\end{cases}
\]
\[
\text{otherwise}
\]

\[
\begin{cases}
\frac{1}{2}(\sigma^{2}_{Zg,i} - \sigma^{2}_{z,i})^{2} & \text{if } |\Sigma_{Zg} - \Sigma_{z}|_{ii} < \delta \\
\delta(|\Sigma_{Zg} - \Sigma_{z}|_{ii} - \frac{1}{2}\delta) & \text{otherwise}
\end{cases}
\]
IV. KOOPMAN-BASED DATA-DRIVEN PREDICTIVE CONTROL

The DeePC framework \cite{the2018deePC} can be extended into nonlinear systems by integrating the equality constraints with \cite{the2018deePC}. However, DeePC formulation suffers from the problem that the prediction step interweaves with the control step. In another word, the algorithm may use a non-optimal prediction result for control. When the penalty factor \(\lambda_L\) is insufficient large and the system is initialized away from the reference, the algorithm tends to compensate the difference with a relative large \(\sigma_y\), which will result in control failure. To tackle this problem, we propose a bi-level programming formulation \cite{58}, where the prediction step is independent from control.

\[
\min_{u, y, g} \left( \sum_{k=0}^{N-1} \left\| y_k - r_{t+k} \right\|_Q + \left\| u_k \right\|^2_R \right) \\
\text{subject to } u_k \in \mathcal{U}, \forall k \in 0, \ldots, N-1 \\
y_k \in \mathcal{Y}, \forall k \in 0, \ldots, N-1 \\
Y_f y = y \\
\text{for some } g \in \Phi(u) \\
\Phi(u) = \arg \min_g \lambda_y \left\| g \right\|^2_Q + \lambda_g \left\| Zg - z \right\|^2_2 \\
\text{subject to } \begin{bmatrix} U_p \\ U_f \end{bmatrix} g = \begin{bmatrix} u_{ini} \\ u \end{bmatrix} \\
\text{with parameter } z = \phi_u(u_{ini}, y_{ini}) \\
\tag{23a}
\]

\[
2g^\top + 2(Zg - z)^\top Z + \mu_1 U_p + \mu_2 U_f = 0 \\
u_k \in \mathcal{U}, \forall k \in 0, \ldots, N-1 \\
x_k \in \mathcal{X}, \forall k \in 0, \ldots, N-1 \\
\tag{23b}
\]

The bi-level problem introduces a hierarchical structure where the upper level problem \eqref{23a} indicates the control step and the lower level problem \eqref{23b} functions as the prediction step. Note that compared to DeePC \cite{the2018deePC}, in \eqref{23b} the squares of the two norm are used so that the objective function of the lower level problem remains smooth.

A usually used approach to solve a bi-level problem is to transform it into a single level problem. Applying optimality conditions and introducing optimal value function are the two main categories of transformation approaches if the bi-level problem fulfills certain conditions \cite{58, Chapter 5}. Here we present the result by replacing the lower level problem \eqref{23b} with its KKT conditions:

\[
\min_{g, u, x, \mu_1, \mu_2} \left( \sum_{k=0}^{N-1} \left\| x_k - r_{t+k} \right\|_Q + \left\| u_k \right\|^2_R \right) \\
\text{subject to } \begin{bmatrix} U_p \\ U_f \end{bmatrix} g = \begin{bmatrix} u_{ini} \\ u \end{bmatrix} \\
2g^\top + 2(Zg - z)^\top Z + \mu_1 U_p + \mu_2 U_f = 0 \\
u_k \in \mathcal{U}, \forall k \in 0, \ldots, N-1 \\
x_k \in \mathcal{X}, \forall k \in 0, \ldots, N-1 \\
\tag{24}
\]

This equivalent single level problem is solvable by many optimization toolboxes.

Remark 3: Solving a bi-level problem is in general NP-hard. To the best of our knowledge, there is no valid approach to solve a general bi-level optimization problem where the lower level problem is non-convex. It is worth mentioning that the wasserstein distanced based prediction problem presented in Section II-C is non-convex. We therefore leave the control formulation integrated with wasserstein distanced based prediction as a future work.

The algorithm is summarized as follows:

\begin{algorithm}
\caption{Koopman based DeePC}
\begin{algorithmic}[1]
\For{$t = T_{ini}, \ldots$}
\State Set $z = \phi_u(u_{ini}, y_{ini})$ ;
\State Solve \eqref{24} to obtain an optimal input sequence $u^*(0)$ ;
\State Set $u_t = u^*(0)$ ;
\State Apply $u_t$ to the system and measure $y_t$
\EndFor
\end{algorithmic}
\end{algorithm}

where $u^*(0)$ denotes the first elements of $u^*$.

V. SIMULATION RESULTS

In this section, the prediction results on a Van der Pol oscillator based on Monte-Carlo prediction and wasserstein distanced based prediction are firstly illustrated. Then a numerical experiment of controlling a bilinear motor is presented. We finally demonstrate the potential of our proposed scheme in the large-scale problems with an example of controlling the nonlinear Korteweg-de Vries equation. The source code of the numerical examples can be accessed through \url{https://github.com/RencciIW/DataDrivenControlCode}.

A. Stochastic prediction

We show the results of prediction of trajectories from a Van der Pol oscillator

\[
\dot{x} = \left[ x_2 \right] - \mu (1 - x_1^2) x_1 - x_2 + u
\]

with \(\mu = 1\). We train a 5-layer network \((2, 12, 22, 12\text{ and }12)\) with 1100 data points sampled from 100 random trajectories generated by Van der Pol oscillator. Each layer excluding the input layer is added with a dropout layer. Choose ReLU as activation function for each hidden layer and Adam as optimizer with learning rate \(10^{-3}\). Set the dropout rate equals to 0.2. The code is implemented with PyTorch \cite{59}. From each of 100 trajectories, we sample 3 trajectory fragments to formulate the Hankel matrix. For a better comparison with following results, in test phase, we choose 3 trajectory fragments from each of 24 trajectories to formulate the Hankel matrix with $T_{ini} = 1$ and $N = 10$. Test the trained network on 50 data points sampled from trajectories, which are independent from the data used for training and hankel matrix formulation. Forward the same data 120 times to the network and compute the mean value and standard derivation of the prediction. The results are shown below. The light blue color indicates two times the standard derivation.

B. Prediction based on Wasserstein distance

We test the new loss function with Van der Pol data. From each of 24 different trajectories, choose one trajectory fragment to formulate the Page matrix. We firstly lift the
data for Page matrix construction with the dropout neural network we trained from last subsection. The mean value and
the standard deviation of the lifted data are computed for estimation. Then send the data to prediction problem to obtain
an optimizer $g^*$. Predict the future trajectory with $x = X_f g^*$, $X_f$ is the Hankel matrix block for prediction.

We compute the mean squared error (MSE) of the prediction based on proposed Wasserstein loss and original quadratic
loss at the 9th time step. The comparison is summarized in following table:

|                      | $x_1$ | $x_2$ |
|----------------------|-------|-------|
| Wasserstein loss     | 0.0817| 0.1132|
| Original loss        | 3.8707| 1.3909|

**TABLE I**: MSE comparison between prediction result computed with different loss functions

It is clear that the Wasserstein loss outperforms the original quadratic loss when uncertainty of lifting function is considered.

**C. Control with Koopman-based DeePC**

1) Control of a bilinear motor: We firstly compare the control algorithm with the algorithm Koopman operator-based MPC controller (K-MPC) proposed in [22] by controlling a bilinear model of a DC motor. [60]

\[
\begin{align*}
\dot{x}_1 &= -(R_a/L_a)x_1 - (k_m/L_a)x_2 u + u_a/L_a \\
\dot{x}_2 &= -(B/J)x_2 + (k_m/J)x_1 u - \tau_1/J \\
y &= x_2 
\end{align*}
\]

where $x_1$ is the rotor current, $x_2$ the angular velocity an the control input $u$ is the stator current and the output $y$ is the angular velocity. The parameters are $L_a = 0.314, R_a = 12.345, k_m = 0.253, J = 0.00441, B = 0.00732, \tau_1 = 1.47, u_a = 60$. The physical constraints on the control input are $u \in [-1, 1]$.

We use 40 trajectories with time horizon 0.25s to construct a mosaic Hankel matrix. All trajectories are randomly initialized
on the unit box $[-1, 1]^2$. The control input obeys to a uniform distribution over the interval $[-1, 1]$. Choose 40 thin plate spline radial basis function with centers selected randomly with uniform distribution over $[-1, 1]^3$ as lifting functions.

Fig. 2: Prediction result using Koopman operator learned by MC dropout

Fig. 3: Prediction result using loss function derived from Wasserstein distance
Since the system states are not directly measurable, we choose the number of delays \( n_d = 1 \). We define \( C = [1, 0, \ldots, 0] \), \( Q = Q_{N_p} = 10 \), \( R = 0.01 \). The prediction horizon \( N = 10 \), which implies 0.1s. Since the system is linear in the lifting space, choose \( T_{ini} = 1 \).

We define \( C = [1, 0, \ldots, 0] \), \( Q = 10 \), \( R = 0.01 \). The prediction horizon \( N = 10 \), which implies 0.1s. Since the system is linear in the lifting space, choose \( T_{ini} = 1 \). The reference is designed as \( r(t) = 0.5 \cos(2\pi t/3) \). Introduce constraints on output \( y \in [-0.4, 0.4] \).

We simulate for 3s and compare the result with a model-based method K-MPC proposed in [22].

\[ \frac{\partial y(t, x)}{\partial t} + y(t, x) \frac{\partial y(t, x)}{\partial x} + \frac{\partial^3 y(t, x)}{\partial x^3} = u(t, x) \]

where \( y(t, x) \) is the unknown function and \( u(t, x) \) the control input. \( x \in [-\pi, \pi] \). The space is descretized into 128 points and the time step \( \Delta t = 0.02s \). The input is assumed to be of the form \( u(t, x) = \sum_{i=1}^{3} u_i(t)v_i(x) \) where \( v_i \) consists of 3 spacial basis functions: \( v_1(x) = e^{-25(x-\pi/2)^2} \) with \( c_1 = -\pi/2 \), \( c_2 = 0 \), \( c_3 = \pi/2 \). The input is constrained to \([-1, 1]\). We initial the system by convexly combining 3 fixed spatial profiles \( y_0^1 = e^{-(x-\pi/2)^2} \), \( y_0^2 = -\sin^2(x/2) \), \( y_0^3 = e^{-(x+\pi/2)^2} \). We choose the states itself, the elementwise square of the state, the elementwise product of the states with its periodic shift as the lifting functions. \( Q \) is the identity matrix and \( R \) is zero matrix. The prediction horizon \( N = 5 \), which implies 0.1s, \( T_{ini} \) remains equal to 1. Formulate the Hankel matrix with 63 trajectories, each of which is simulated for 0.5s.

Despite of large dimension, the algorithm is still capable of computing the optimum input in an acceptable time and tracking the reference.

**VI. Conclusion**

In this work, we extend a data-driven predictive control method into nonlinear systems. The underlying idea is to lift the system with Koopman operator into infinite dimensional space where the system evolves linearly along the nonlinear system trajectories. Approximation of nonlinear lifting functions based on a purely data-driven framework is proposed, along with considerations on the uncertainty of the approximation, which enabling a novel data-driven simulation scheme based on wasserstein distance.
[54] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” IEEE transactions on neural networks, vol. 5, no. 2, pp. 157–166, 1994.

[55] C. Villani, Optimal transport: old and new. Springer Science & Business Media, 2008, vol. 338.

[56] R. T. Powers and E. Størmer, “Free states of the canonical anticommutation relations,” Communications in Mathematical Physics, vol. 16, no. 1, pp. 1–33, 1970.

[57] A. Beck, First-order methods in optimization. SIAM, 2017.

[58] S. Dempe, Foundations of bilevel programming. Springer Science & Business Media, 2002.

[59] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.

[60] S. Daniel-berhe and H. Unbehauen, “Parameter estimation of the nonlinear dynamics of a thyristor driven dc-motor experimental set-up using hmf-method,” IFAC Proceedings Volumes, vol. 30, no. 11, pp. 203 – 208, 1997, iFAC Symposium on System Identification (SYSID’97), Kitakyushu, Fukuoka, Japan, 8-11 July 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667017428473

[61] R. M. Miura, “The korteweg–devries equation: A survey of results,” SIAM review, vol. 18, no. 3, pp. 412–459, 1976.