Learning of Causal Observable Functions for Koopman-DFL Lifting Linearization of Nonlinear Controlled Systems and Its Application to Excavation Automation*

Nicholas S. Selby[1] and H. Harry Asada[2]

***Abstract*—Effective and causal observable functions for low-order lifting linearization of nonlinear controlled systems are learned from data by using neural networks. While Koopman operator theory allows us to represent a nonlinear system as a linear system in an infinite-dimensional space of observables, exact linearization is guaranteed only for autonomous systems with no input, and finding effective observable functions for approximation with a low-order linear system remains an open question. Dual-Faceted Linearization uses a set of effective observables for low-order lifting linearization, but the method requires knowledge of the physical structure of the nonlinear system. Here, a data-driven method is presented for generating a set of nonlinear observable functions that can accurately approximate a nonlinear control system to a low-order linear control system. A caveat in using data of measured variables as observables is that the measured variables may contain input to the system, which incurs a causality contradiction when lifting the system, i.e. taking derivatives of the observables. The current work presents a method for eliminating such anti-causal components of the observables and lifting the system using only causal observables. The method is applied to excavation automation, a complex nonlinear dynamical system, to obtain a low-order lifted linear model for control design.**

I. Introduction

There is a growing need in the construction and mining industries for excavation automation. Various technologies are being developed for operating excavators autonomously with increased productivity and fuel efficiency [1]. The recent and projected growth of the global construction industry [2] and the dangers of the excavation work environment [3] are major drivers behind the development of intelligent excavators for performing earth-moving tasks.

Excavation is a highly nonlinear process where soil and rocks interact with the bucket of an excavator in a complex manner (see Fig. 1). While terramechanics models have been studied for many decades, their validity is limited due to the difficulty of identifying the numerous parameters of mechanistic models. Data-driven methods have recently been introduced to autonomous excavation for capturing complex nonlinearities [3]–[7], yet the nonlinear models are still too complex to use, in particular, for real-time control.

Lifting linearization is a methodology for representing a nonlinear dynamical system with a linear dynamic model in a high dimensional space. Underpinned by Koopman operator theory, nonlinear systems represented with supernumerary state variables behave more linearly in the lifted space. The method has recently been applied to various robotics and automation challenges, including active learning [8], soft robotics [9], human-robot interaction [10], power systems [11], and mission planning [12]. More broadly, deep learning has proven a valuable tool for a variety of lifting linearization techniques [13]–[16].

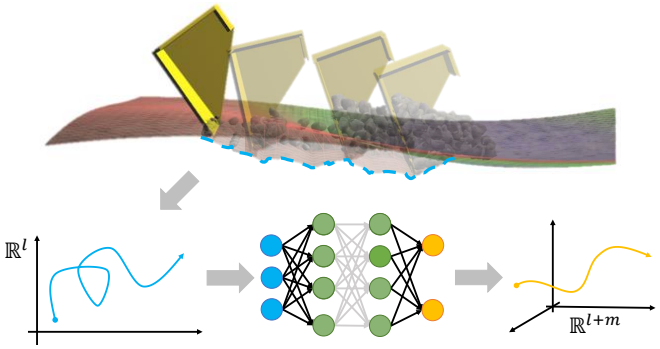

Fig. 1. Learned lifting linearization of autonomous excavation.

The original Koopman Operator has two major limitations:

1) The theory is applicable only to dynamical systems with no exogenous input, i.e. autonomous systems, and
2) Exact linearization requires an infinite-dimensional space, except for a restricted class of systems.

Any extension to non-autonomous, finite-dimensional systems is no longer exact, but an approximation. Various methods for truncating the system with a finite-dimensional space have been reported. Among others, the eigendecomposition of the lifted system allows us to represent the system at desirable accuracy and granularity while providing useful insights into the system [17]. Furthermore, the extended Dynamic Mode Decomposition (eDMD) is completely data-driven, providing a practical tool for complex nonlinear system representation [18]. These methods, however, need a set of observables, i.e. output variables, which are nonlinear functions of independent state variables. It is still an open question how to find an effective set of observable functions.

One of the key challenges in the lifting linearization of nonlinear systems with exogenous input is causality. If observable functions are functions of both state variables and input variables, we cannot use such observables for lifting the system. Lifting entails computing time derivatives of the observables and, thereby, the dynamic equations inevitably

*This material is based upon work supported by National Science Foundation Grant NSF-CMMI 2021625.

[1]Nicholas S. Selby is a PhD Candidate in the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA, USA nselby@mit.edu

[2]H. Harry Asada is with the Faculty of the Department of Mechanical Engineering at the Massachusetts Institute of Technology asada@mit.edu

include the time derivative of input. In discrete formulation, this means the use of future input. Including these input terms, the time-evolution of the observables turns out not to be causal. If one measures a set of candidates of observables from a nonlinear system that is subject to control inputs and uses the measured variables for lifting the system, they may end up with a non-causal dynamical system.

In Dual-Faceted Linearization (DFL)—another approach to lifting linearization—this causality issue has been analyzed based on physical system modeling theory [19]. In DFL, the propagation of inputs across the nonlinear dynamical system can be tracked, and their effect upon all observables, called auxiliary variables, can be localized. Assuming that inputs are linearly involved in observables, a method has been established for eliminating the input-dependent component from each observable and lifting the dynamics by using the remaining input-free observables. In the Koopman-based lifting linearization, too, it is assumed that the observables are input-affine in order to eliminate the input-dependent components from observable functions, so that a causal dynamic model can be obtained [20].

Lifting linearization is a powerful methodology for tackling a broad spectrum of nonlinear problems, in particular, excavation process modeling and control. However, two critical challenges have not yet been fully solved:

1) Finding an effective set of observables to approximate a nonlinear system in a low-dimensional lifted space
2) Finding causal observables uncorrelated with inputs

The objective of the current work is to solve these two challenges. We present a low-dimensional, causal, lifting linear model obtained from experimental data. Neural networks are used to find effective observables through learning.

In the following, we summarize a basic formulation of lifting linearization in § II. We present the learning method for obtaining an effective set of observables in § III. First, we deal with nonlinear controlled systems where all measured observables are not affected by inputs. Then, the method is extended for physical observables that may be functions of inputs. Simple numerical examples are discussed for validating the proposed method in § IV, and we apply it to excavation process modeling in § IV-B.

II. Background

This section summarizes background knowledge for readability. More details can be found in [19], [21].

A. Koopman Operator Theory

First proposed in 1931 by Koopman [21], Koopman operator theory originally modeled only autonomous systems, and the operator mapped the nonlinear dynamics only onto an infinite-dimensional linear space. Later techniques expanded the use of the operator for nonautonomous systems [22] and developed methods for approximating the infinite-dimensional mapping with a computationally feasible, finite-dimensional space [23]–[25].

Let the discrete-time dynamics of a nonlinear, autonomous system with state $x_t \in \mathbb{R}^l$ at time t be given by $x_{t+1} = f(x_t)$. Furthermore, define a vector of nonlinear observables of the state, $\eta_t = g(x_t) \in \mathbb{R}^m$.

The Koopman operator, $\mathcal{K}$, is linear and infinite-dimensional and applies to observable functions: $\mathcal{K}_f g = g \circ f$, where $\circ$ represents the composition operator.

B. Dual-Faceted Linearization (DFL)

Despite the use of Koopman operators to provide a lifting linearization for autonomous systems, the theory provides no method by which to select an effective set of observables. Because it is infeasible to compute the infinite-dimensional space with finite computational resources, the choice of which observables to use is very important. DFL [19] uses a particular class of observables that are determined based on physical modeling theory and bond graphs [26]. Those observables, called auxiliary variables, are physically meaningful, and may be measured physically. Furthermore, causality analysis of the method allows us to examine how exogenous inputs propagate the system and influence specific auxiliary variables. Using those variables with no input influence, one can obtain a lifted system that is causal. Alternatively, input-dependent variables can be "laundered" into causal variables.

Consider the discrete-time dynamics of a nonlinear, nonautonomous system with input $u_t \in \mathbb{R}^n$ at time t given by:

$$x_{t+1} = f(x_t, u_t) \tag{1}$$

Assuming that the system is a lumped-parameter system with integral causality, we can choose outputs of all the nonlinear elements involved in the system as observables $\eta_t = g(x_t) \in \mathbb{R}^m$ to augment the system state and construct a linear representation of the system dynamics:

$$x_{t+1} = A_x x_t + A_\eta \eta_t + B_x u_t \tag{2}$$

where $A_x \in \mathbb{R}^{l \times l}$, $A_\eta \in \mathbb{R}^{l \times m}$, and $B_x \in \mathbb{R}^{l \times n}$ are fixed matrix coefficients determined by the physical structure of the system. This part of the state evolution is exact.

We approximate the η-dynamics using a second equation:

$$\eta_{t+1} = H_x x_t + H_\eta \eta_t + H_u u_t + r_{\eta_{t+1}} \tag{3}$$

where $H_x \in \mathbb{R}^{m \times l}$, $H_\eta \in \mathbb{R}^{m \times m}$, and $H_u \in \mathbb{R}^{m \times n}$ are fixed matrix coefficients and $r_{\eta_{t+1}} \in \mathbb{R}^m$ is a residual. Unlike A_x, A_η, and B_x, which are determined from the physical structure of the system, H_x, H_η, and H_u must be regressed from data. For brevity, define coefficient matrices $A \triangleq (A_x, A_\eta, B_x) \in \mathbb{R}^{l \times p}$ and $H \triangleq (H_x, H_\eta, H_u) \in \mathbb{R}^{m \times p}$ and datum vector $\xi_t \triangleq (x_t^\intercal, \eta_t^\intercal, u_t^\intercal)^\intercal \in \mathbb{R}^p$ where $p = l + m + n$. Apply a negative discrete-time shift operator T_{-1} to (3) to optimize H to minimize the mean squared error of predicting η_{t+1}:

$$\begin{aligned} H^{\mathrm{o}} &= \arg\min_H \mathrm{E}\left[\left|H\xi_{t-1} - \eta_t\right|^2\right] \\ &= \mathrm{E}\left[\eta_t \xi_{t-1}^\intercal\right] \left(\mathrm{E}\left[\xi_{t-1} \xi_{t-1}^\intercal\right]\right)^{-1} \end{aligned} \tag{4}$$

where $\mathrm{E}[\cdot]$ is the expectation operator. Assuming that the system is persistently excited and that u_t is not collinear with x_t, there is a unique solution, H^{o}.

The original nonlinear dynamics f can now be modeled using the dual-faceted linear dynamics:

$$x_{t+1} = A\xi_t; \quad \eta_{t+1} \approx H\xi_t; \quad \eta_0 = g(x_0); \tag{5}$$

Practical benefits of using DFL as a tool to model systems include:

- Augmented state feedback can be used to better inform controllers [27].
- Linear observer design is enabled for augmented state feedback.
- Model-predictive control is convex [17].
- Because DFL is based in physical modelling theory, augmented state systems may be measurable, and dynamics may have physical intuition [28].

For these reasons, DFL has proven to be a valuable tool in modeling nonlinear systems. However, DFL requires knowledge of the structure of the physical system. Furthermore, many systems contain no obvious, measurable observables with which to augment the state, and there is no guarantee that, when they do exist, physically meaningful observables make the best choices for augmenting the system state.

C. Machine Learning for Linear Latent Spaces

Like Koopman operator theory and DFL, learned latent-space dynamic modeling techniques also involve constructing a nonlinear representation of the original state, then using a model to evolve the new "observables" through time. In Koopman operator theory and DFL, the dynamic model is linear. Recently, much work has been done to explore applying deep learning techniques to Koopman operators.

Abraham and Murphey [8] presented an active learning strategy for robotic systems that extended observables, g, from Koopman operator theory to include the control input, u_t. Their algorithm trains a neural network to approximate an optimal g. In each epoch, they recompute a finite-dimensional matrix approximation of the Koopman operator $\mathcal{K}$, but because g is a function of both x_t and u_t, regressing such a matrix requires knowing u_{t+1} in addition to x_{t+1}. To solve this causality problem, they propose replacing u_{t+1} with u_t in the Koopman operator update.

Han *et al.* [29] also proposed using a neural network that approximates an optimal lifting function g. At the end of each epoch, after feeding x_{t-1} forward through g, they solve a least squares optimization problem to regress a modified H° where $H_x = 0$, then use the learned model to backpropagate the error $\mathrm{E}\left[||g(x_t) - H^{\circ}\xi_{t-1}||_F\right]$ plus a penalty on the norms of the components of H°. Because they do not track the evolution of the state, x, directly, they must simultaneously learn an additional matrix to approximate g^{-1}.

Lusch *et al.* [30] and Mastia and Bemporad [31] replaced the g^{-1} matrix approximation of Han *et al.* with a neural network decoder. Their models learn to minimize cost along three axes: neural network g^{-1} must invert neural network g, i.e. $x = g\left(g^{-1}(x)\right)$; the nonlinear model must be able to predict, i.e. $x_{t+1} = g^{-1}\left(Hg(x_t)\right)$; and the lifted state must propagate linearly, i.e. $g(x_{t+1}) = Hg(x_t)$.

Work by Yeung *et al.* [32] applied deep learning to dynamic mode decomposition (DMD) by training a neural network to approximate the "snapshot" function mapping state to physical observable. Instead of optimizing a complicated cost function like in [29]–[31], their work simply trains a model to map states to other measurements of the system, then regresses a linear dynamic model to propagate the measurements forward in time.

Other work in learned latent spaces for lifting linearizations leverages neural networks to approximate functions similar to Koopman's observables in reinforcement learning [16], sampling-based motion planning [14], Kalman filtering [15], and partially observable Markov decision process [13] paradigms. However, these works all learn linearizations A and H that are time- or state-dependent, further reducing the long-term robustness of the linear model and giving up the benefits of provably convex optimal control.

D. Anticausal Observables

Most lifting linearization techniques, including Koopman operator theory and DFL, require that the lifting observables be control input-independent, i.e. $\eta = g(x)$. If auxiliary variables depend on the control input, u, i.e. $\eta = g(x, u)$, then propagating η forward through time requires knowledge of future values of control input:

$$\eta_{t+1} \approx \eta_t + \frac{\partial \eta}{\partial x}\left(x_{t+1} - x_t\right) + \frac{\partial \eta}{\partial u}\left(u_{t+1} - u_t\right) \tag{6}$$

In order to avoid problems with causality, most methods explicitly avoid augmenting the system state with control input-dependent variables. There are two common techniques.

The first solution to the causality problem is to include a state-feedback control law in the model [8], [29]. By constraining the control input to be a known function of state, the system becomes effectively autonomous, and the original formulation of Koopman operator theory applies. This works for regulators, including for controllers regulating a system to follow a predetermined state trajectory, but no exogenous input is allowed.

The second solution to the causality problem is to assume that the auxiliary variables are linear in u [19], [20], [33]:

$$\eta(x, u) = \eta^*(x) + Du \tag{7}$$

where η^* is exclusively state-dependent and D is a fixed matrix coefficient of u.

Although predicting future values of $\eta(x, u)$ remains impossible without a control law, this formulation allows for the modeling of the evolution of $\eta^*(x)$. The auxiliary state equation can be rewritten:

$$\eta^*_{t+1} = H^*_x x_t + H^*_\eta \eta_t + H^*_u u_t + r_{\eta^*,t} \tag{8}$$

where $r_{\eta^*,t}$ is a residual.

Substituting (7) into (8) yields:

$$\eta^*_{t+1} = H^*_x x_t + H^*_\eta \eta^*_t + \left(H^*_u + H^*_\eta D\right) u_t + r_{\eta^*,t} \tag{9}$$

which is a causal, augmented state dynamic equation. The question of causality is therefore solved by preprocessing the auxiliary state data to filter out their dependence on u.

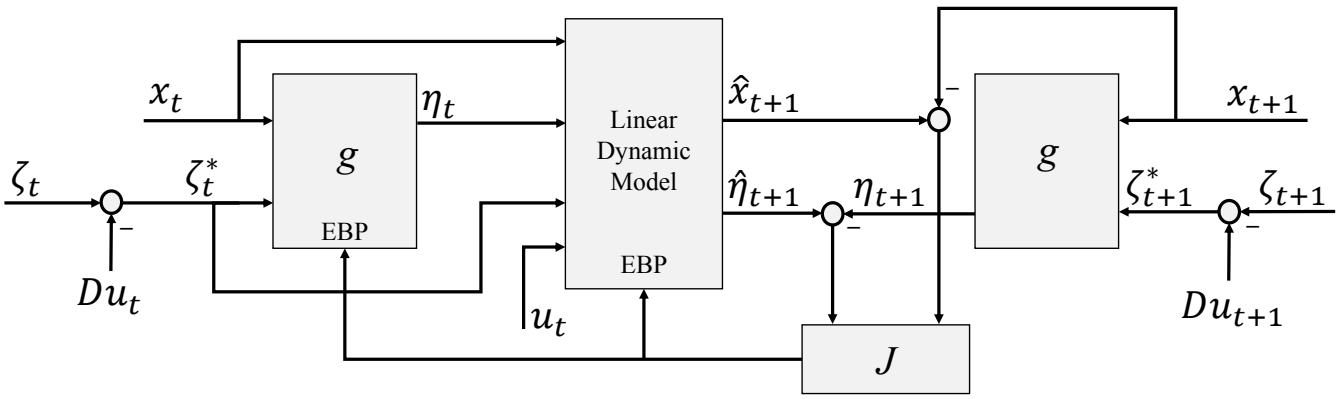

Fig. 2. Block diagram of the learned lifting linearization algorithm. The loss, J, is used to tune the weights of neural network, g, and linear dynamic model matrices A and H via error backpropagation (EBP). Note that both instances of g are equivalent to each other for all time t.

III. Modeling Algorithm

Fig. 2 shows the overview of the modeling algorithm. The learning system consists of three major components. The first is the Linear Dynamic Model predicting the transition of the system. The second is a multi-layer neural network for generating auxiliary variables, and the third is the process of evaluating the prediction error.

We assume all state variables are accessible and sufficient data for training are attainable. All state variables are fed into the left neural network, g, to produce a set of synthetic observables, η_t, to be learned. The state x_t, observables η_t, and input u_t are fed into the linear dynamic model parameterized by matrices A and H. The linear model produces predicted state $\hat{x}_{t+1}$ and predicted observables $\hat{\eta}_{t+1}$. The predicted state and observables are compared to their ground truth values, x_{t+1} and $\eta_{t+1} = g(x_{t+1};\theta)$, respectively. The right neural network in the figure is a twin copy of the left neural network, sharing parameters θ.

The squared error of the predicted state and observables, J, is used for updating the linear dynamic model with respect to parameters A and H and the neural network weights. The update of these parameters is computed via error backpropagation (EBP).

The causality analysis involved in the DFL modeling allows us to examine whether observables, called auxiliary variables, are functions of state alone or include inputs. If some auxiliary variables are causal, having no dependence on control input, they can be added to the state for lifting the dynamics. Let ζ_t^* represent causal auxiliary variables. As shown in Fig. 2, the causal observables can be fed into the neural network, so that the synthetic auxiliary variables η_t can be produced from richer data. Note that the ground truth η_{t+1}, too, can be produced in response to not only state x_{t+1} but also ζ_{t+1}^*. The tunable parameter space of the neural network is expanded with the use of the causal auxiliary variables.

In case physically measurable auxiliary variables are functions of both state and input, such input-dependent auxiliary variables cannot be used in their original form for lifting the dynamics. It is necessary to filter out the input components from the observables. Fig. 2 also shows a simple filter to eliminate the effect of input from those variables, ζ_t, as discussed in § III-B.

A. Discrete-Time Learned Lifting Linearization

Consider the discrete-time dynamic system from (1). Let g be a neural network, illustrated in Fig. 3, defined by randomly initialized parameters θ to generate synthetic observables:

$$\eta_t^\theta = g(x_t;\theta) \tag{10}$$

Define a datum vector $\xi_t^\theta \triangleq \left(x_t^\mathsf{T}, \eta_t^{\theta\mathsf{T}}, u_t^\mathsf{T}\right)^\mathsf{T}$. Let $A \in \mathbb{R}^{l\times p}$ and $H \in \mathbb{R}^{m\times p}$ be matrix coefficients modeling the state and augmented state transition dynamics, respectively. We override (5) to include residuals in the original and augmented state dynamic equations:

$$\begin{cases} x_{t+1} = A\xi_t^\theta + r_{x_{t+1}}^\theta \\ \eta_{t+1}^\theta = H\xi_t^\theta + r_{\eta_{t+1}}^\theta \\ \eta_0^\theta = g(x_0;\theta) \end{cases} \tag{11}$$

Given observation data of x_t, x_{t-1}, and u_{t-1}, we synthesize observations of the augmented state, $\eta_t^\theta = g(x_t;\theta)$ and $\eta_{t-1}^\theta = g(x_{t-1};\theta)$, and assemble datum vectors ξ_{t-1}^θ.

By applying the discrete-time shift operator T_{-1} to (11) and rearranging, we can compute a residual for each observation: $r_{x_t}^\theta = x_t - A\xi_{t-1}^\theta$ and $r_{\eta_t}^\theta = g(x_t;\theta) - H\xi_{t-1}^\theta$.

We define a quadratic loss function, $J_t(\theta, A, H)$, used to train the model:

$$J_t(\theta, A, H) \triangleq r_t^{\theta\mathsf{T}} Q r_t^\theta \tag{12}$$

where Q is a symmetric matrix coefficient and r_t^θ is a total residual given by $r_t^\theta \triangleq \left(r_{x_t}^{\theta\mathsf{T}}, r_{\eta_t}^{\theta\mathsf{T}}\right)^\mathsf{T}$.

Model parameter matrices A and H, as well as the parameters of the neural network, θ, are computed by solving the following optimization problem via error backpropagation:

$$\theta^o, A^o, H^o = \underset{\theta,A,H}{\arg\min}\, \mathrm{E}\left[J_t(\theta, A, H)\right] \tag{13}$$

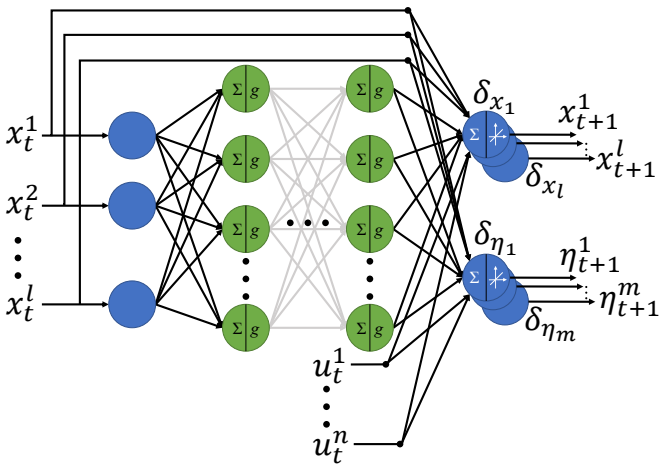

Fig. 3. Diagram of the neural network and linear dynamic model to compute x_{t+1} and η_{t+1} given x_t and u_t. With abuse of notation, we hereafter include ζ^* in x.

B. Extension to Anticausal Observables

As discussed in [19], augmenting the state with physical observables is often useful. Because this learned lifting linearization is data-driven, augmenting the state, x, with control-independent, physical observables is trivial. However, as reviewed in § II, if the augmented state is dependent on the control input, the augmented system dynamics become anticausal. In state space modeling, output equations include inputs algebraically if there is a direct transmission term from inputs to outputs [34], [35]. Namely, observations of the system are functions of x_t and u_t. Consider a vector of physical observables, $\zeta(x,u) \in \mathbb{R}^z$, suspected of including a dependence on u. As in [19], [20], and [33], assume that this dependence is linear:

$$\zeta(x,u) = \zeta^*(x) + Du \tag{14}$$

where $\zeta^*(x) \in \mathbb{R}^z$ is exclusively a function of state and $D \in \mathbb{R}^{z\times n}$ is a fixed matrix coefficient of u. Assuming mean-zero data, because $\zeta^*(x)$ is uncorrelated with the control input, $\mathrm{E}\left[\zeta^*(x)u^\intercal\right] = 0$. Therefore, multiplying (14) by $u^\intercal$ and taking the expectation yields $\mathrm{E}\left[\zeta(x,u)u^\intercal\right] = D\mathrm{E}\left[uu^\intercal\right]$.

Given observations of $\zeta(x,u)$ and u, computing D becomes a least-squares linear regression:

$$\hat{D} = \mathrm{E}\left[\zeta(x,u)u^\intercal\right]\mathrm{E}\left[uu^\intercal\right]^{-1} \tag{15}$$

assuming that the input is persistently exciting.

Before training the learned lifting linearization model in (11), we preprocess the data to "clean" the physical observables from any linear dependence on u via

$$\zeta^*(x_t) = \zeta(x_t,u_t) - \hat{D}u_t \tag{16}$$

for each observation, t. Then, we augment the DFL model to include $\zeta^*(x)$. We override (10) with $\eta_t^\theta = g\left((\zeta^{*\intercal}(x_t), x_t^\intercal)^\intercal;\theta\right)$ and follow the same training procedure described above to tune g, A, and H using (13).

The complete learned lifted linearization algorithm is summarized in Algorithm 1.

Algorithm 1: Learned Lifting Linearization

Result: Lifting linearization of nonlinear dynamics
Randomly initialize neural network g and linear dynamic model A, H;
$\hat{D} \leftarrow \mathrm{E}\left[\zeta u^\intercal\right]\mathrm{E}\left[uu^\intercal\right]^{-1}$;
while *training* **do**
  get batch of x_t, ζ_t, u_t, x_{t+1}, ζ_{t+1}, u_{t+1} from training dataset ;
  $\zeta_t^* \leftarrow \zeta_t - \hat{D}u_t$;
  $\zeta_{t+1}^* \leftarrow \zeta_{t+1} - \hat{D}u_{t+1}$;
  $\eta_t \leftarrow g((x_t, \zeta_t^*))$;
  $\eta_{t+1} \leftarrow g((x_{t+1}, \zeta_{t+1}^*))$;
  $\xi_t \leftarrow (x_t^\intercal, \zeta_t^{*\intercal}, \eta_t^\intercal, u_t^\intercal)^\intercal$;
  $r \leftarrow \left(((x_{t+1}^\intercal, \zeta_{t+1}^{*\intercal})^\intercal - A\xi_t)^\intercal, (\eta_{t+1} - H\xi_t)^\intercal\right)^\intercal$;
  $J \leftarrow r^\intercal Q r$;
  backpropagate J to update g, A, and H using Adam ;
end
$A_u \leftarrow A_u + A_\zeta D$;
$H_u \leftarrow H_u + H_\zeta D$;

IV. Numerical Examples

A. Toy Problem

The modeling algorithm in § III is implemented in PyTorch [36] on a laptop running Ubuntu 18.04.5 LTS. The codebase is hosted as a git repository at [37].

We test the learned lifting linearization (L3) algorithm on the nonlinear, massless spring-damper illustrated in Fig. 4 with $\Phi_\mathrm{R}(\mathrm{e_R}) = 2/\left(1+e^{-4\mathrm{e_R}}\right) - 1$, and $\Phi_\mathrm{C}(q) = \mathrm{sgn}(q)q^2$. We generate 100 5s trajectories at 20Hz with initial conditions and control inputs drawn from uniform random distributions. The state, x, consists only of the linear position, q, and the control input, u, is the scalar effort ($n = 1$). We use the system bond graph to identify observables $\zeta = (\mathrm{f}, \mathrm{e_C})^\intercal$.

The neural network, g, approximating the optimal synthetic observables, η, is a fully connected network of 3 linear input neurons ($l = 3$); two hidden layers, each with 256 ReLU neu-

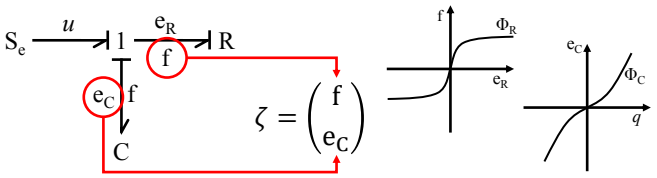

Fig. 4. Bond graph of a nonlinear first-order system with state variable q. In the bond graph, a nonlinear capacitor, C, and a nonlinear resistor, R, are connected to an effort source, $\mathrm{S_e}$, that is an exogenous input $u(t)$. Causality analysis of the bond graph determines that effort variable $\mathrm{e_C}$ is the output of the nonlinear capacitor, while the output of the nonlinear resistor is flow variable f. In the electrical circuit analogy, the effort variable $\mathrm{e_C}$ is the voltage across the capacitor, and the flow variable f is the current flowing through the resistor. They are connected with the exogenous input voltage $u(t)$ at the "1" junction, which is equivalent to Kirchhoff's Voltage Law. The causality analysis also reveals that a direct transmission path exists from input $u(t)$ to flow variable f. Therefore, auxiliary variable f is not a causal variable for lifting the system.

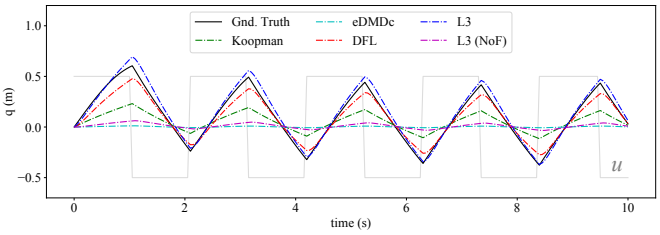

Fig. 5. Results of open-loop simulation predicting the state, q of the toy problem from Fig. 4 excited by a square wave input (in gray). The solid black line indicates the true trajectory. The dashed lines indicate the open-loop simulated trajectories of various models: Koopman-with-control (Koopman, dim = 33), extended dynamic mode decomposition with control (eDMDc, dim = 6), dual-faceted linearization (DFL, dim = 4), learned lifting linearization (L3, dim = 6), and L3 without first filtering out the control input using (16) (L3 NoF, dim = 6).

rons; and 2 output neurons, creating 2 synthetic observables ($m = 2$). Before training, the data are randomly divided 80-20 into a training set and a validation set. The neural network, g, and the linear model consisting of A and H are trained in batches of 32 input-output pairs using an Adam optimizer [38] with $\alpha = 10^{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The quadratic cost parameter $Q = I$. Before each training epoch, the learned lifting linearization model is evaluated using the validation dataset without backpropagating the loss. Training continues until the validation loss begins to increase.

We benchmark the learned lifting linearization algorithm against Koopman-with-control, eDMDc, DFL, and L3 without the anticausal filter. Using the same data from the training and validation sets described above, 32 observables using polynomial basis functions are created for the Koopman observables and two similar observables are created for eDMDc. We train the eDMDc and DFL models using the same measurements, ζ, as L3.

After training, we simulate all models given a zero initial condition and a square wave input trajectory. The modeled state trajectory is compared against the ground truth in Fig. 5. The learned lifting linearization model outperforms the Koopman model despite the significantly lower dimensionality. The integrated squared errors of the simulated models are recorded in Table I.

Note that the fidelity of the Koopman operator model is sensitive to hyperparameters. As discussed in [9], without L1 regularization, high-dimensional Koopman models quickly overfit to the training data. Both DFL and L3 outperform eDMDc due to the anticausal filter compensating for the dependence of $\mathrm{e_R}$ on u. Without the anticausal filter, L3 performs only marginally better than eDMDc. L3 also has a slight advantage over DFL: in addition to penalizing nonlinearities in the state transition equation, the cost function

TABLE I
INTEGRATED SQUARED ERROR OF THE MODELS USED TO SIMULATE THE TOY PROBLEM OVER TEN SECONDS.

Koopman	eDMDc	DFL	L3	L3 (NoF)
5.9	14	0.73	0.48	12

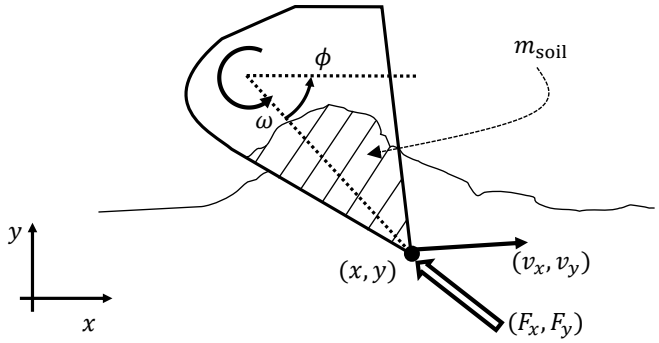

Fig. 6. Diagram of the states and physical observables included in the data.

of the learned lifting linearization in (12) also penalizes nonlinearities in the augmented state transition equation.

B. Excavation Process Modeling

We also test the learned lifting linearization algorithm on an autonomous excavator simulation using *agxTerrain*, a specialized module of the AGX Dynamics [39] physics simulator used to test algorithms for autonomous excavation [40]–[43] illustrated in Fig. 1. We generate a random soil profile by summing several 2-D Gaussians of random height and variance, yielding soil shapes like those in Fig. 1. Soil properties are set in accordance to the AGX "gravel" profile.

We collect 100 7.5s randomized trajectories from large sections of the workspace at 100Hz, setting one of them aside for testing. A diagram of the collected data is illustrated in Fig. 6. The trajectories include six states, x: position along the x-axis, x; position along the y-axis, y; bucket angle, ϕ; velocity along the x-axis, v_{x}; velocity along the y-axis, v_y; and rotational velocity of the bucket, ω. The trajectories also include three control inputs, u: force along the x-axis, u_{x}; force along the y-axis, u_y; and torque actuating the bucket, u_ϕ. The trajectories also include three physical observables, ζ: soil reaction force on the bucket along the x-axis, F_{x}; soil reaction force on the bucket along the y-axis, F_y; and mass of the soil in the bucket, m_{soil}.

These trajectories are generated using a naïve, noisy PID controller on the translation forces and bucket angle:

$$\begin{aligned} u_{\mathrm{x}} &= \mathrm{PID}(\dot{\mathrm{x}} - \mathcal{U}(\dot{\mathrm{x}}_{\min}, \dot{\mathrm{x}}_{\max})) + \mathcal{U}(-w_{\mathrm{x}}, w_{\mathrm{x}}) \\ u_y &= \mathrm{PID}(\dot{y} - \mathcal{U}(\dot{y}_{\min}, \dot{y}_{\max})) + \mathcal{U}(-w_y, w_y) \\ u_\phi &= \mathrm{PID}(\phi - \mathcal{U}(\phi_{\min}, \phi_{\max})) + \mathcal{U}(-w_\phi, w_\phi) \end{aligned} \tag{17}$$

where $\mathcal{U}$ is the uniform random distribution and w_{x}, w_y, and w_ϕ are bounds on additional noise added to ensure persistent excitation. The set points are drawn from a uniform random distribution in accordance with [44].

The learned lifting linearization and Koopman models for the terramechanics experiment are almost identical to those of the nonlinear spring-damper experiment, with some exceptions. The neural network, g, has 9 input neurons, 1 hidden layer with 256 ReLU neurons, and 4 output neurons. On average, training the L3 model took 2.5 hours. The domain of the Koopman dynamic model has a dimensionality of 67, compared to 16 for the learned lifting linearization model. We

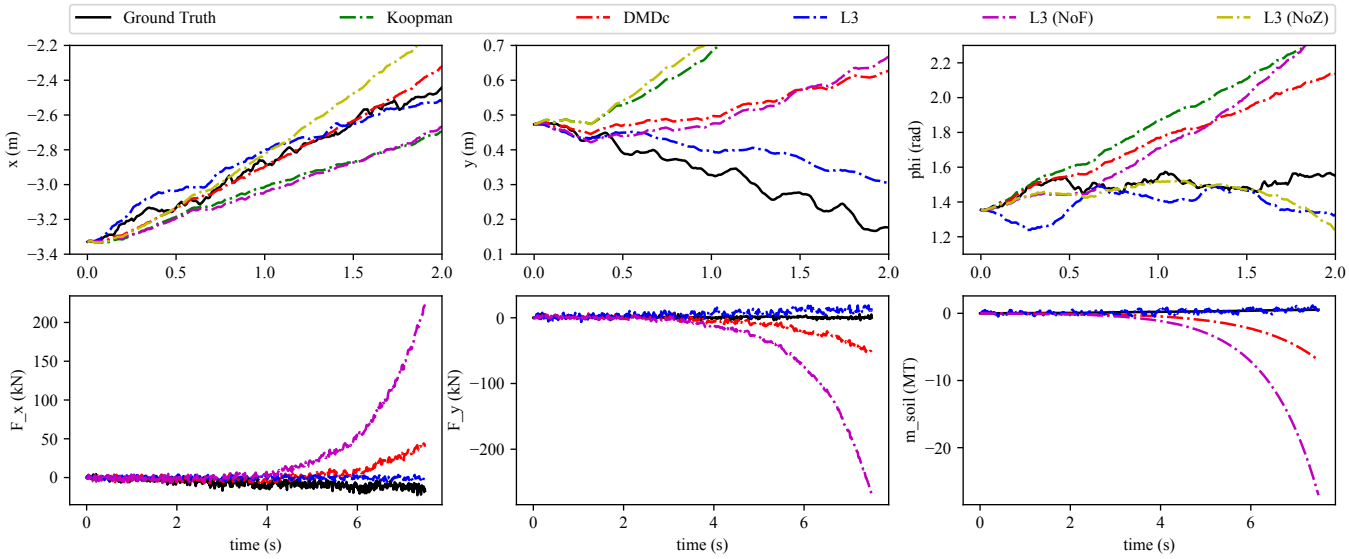

Fig. 7. Results of open-loop simulation predicting the positional state and observable trajectories of the bucket through the soil given the initial condition and control input. The solid black line indicates the true, observed trajectory. The dashed green line indicates the trajectory predicted by the Koopman model. The dashed lines indicate the open-loop simulated trajectories of various models: dynamic mode decomposition with control (DMDc), learned lifting linearization (L3), L3 without first filtering out the control input using (16) (L3 NoF), and L3 without any information from ζ (L3 NoZ).

also benchmark against a DMDc model with a dimensionality of 12 trained using the observables, ζ, in addition to the state. There was not a significant performance difference between DMDc and eDMDc. In addition to benchmarking the learned lifting linearization model against the Koopman and DMDc models, we also compare the results with and without filtering out the control input from the physical observables using (16). In this model, instead of following the procedure described in § III-B, we incorporated $\zeta(x, u)$ directly into the state without filtering. Effectively, this forces the model to violate causality by predicting future values of observables dependent upon control input, u. Finally, we also test L3 without any information from observations, ζ, to examine the effect of removing supplementary measurements.

After training the learned lifting linearization and Koopman models, we simulate both models using the control input trajectory and initial condition from the testing trajectory. The modeled trajectories of the positional states, x, y, and ω, and the three observables, F_x, F_y, and m_{soil}, are compared against the ground truth trajectories in Fig. 7. The integrated squared error across the three states for the learned lifting linearization model is only 5% and 18% of the same errors for the Koopman and DMDc models, respectively.

We also retrain the learned lifting linearization model *without* first filtering out the control input using (16) and simulate using the same technique, L3 (NoF). The integrated squared error across the three states is more than eight times the same error for the standard L3 model. Input filtering is vital to model robustness. A similar experiment performed without any information from supplemental measurements, L3 (NoZ), results in an error more than ten times greater than the standard L3 model. Various experiments retraining the L3 model without access to individual states or observables (left out of Fig. 7) result in similar reductions in accuracy.

V. Discussion and Conclusion

In this section, we highlight main takeaways from our experiments.

In this paper, we presented a learned lifting linearization algorithm to model nonlinear dynamic systems. This model extended Koopman operator theory and dual-faceted linearization by training a neural network to produce nonlinear observables to augment the state. We also presented an algorithm to "clean" anticausal physical observables of any linear dependence on control input so that they can be used by the neural network to generate richer synthetic observables. We tested this algorithm on a nonlinear, massless spring-damper model and an autonomous excavation simulation, and we compared the results against Koopman, DMD, and DFL models. Learned lifting linearization outperformed all benchmarks at minimizing state prediction error.

As with many data-driven techniques, the quality of the training data is paramount to model accuracy. The reason for the reduced performance in modeling progression along the y-axis is likely the relatively small size of the domain of the training data along that dimension. If the system is initialized in a configuration that is different from what has been recorded during data collection, or if the quantity of data available for learning is reduced by more than half, models such as L3 and DMD perform substantially worse.

In real-world excavation tasks, soil properties can vary. While this paper addresses more homogeneous soil profiles like gravel, intelligent use of observables to include information about the environment could enable L3 to learn optimal auxiliary variables and linear models for dynamic soil properties. Additionally, L3 could be primed with simulator data to reduce the amount of hardware-in-the-loop learning required.

Conventional excavators often lack the ability to do proper

end effector force control without expensive modifications to the hydraulics. State-of-the-art excavators have solved this problem [7], and this work focused on the nonlinearities involved in soil dynamics. Future work on this topic should include modeling of the nonlinearity of hydraulic systems.

REFERENCES

[1] S. Dadhich, U. Bodin, and U. Andersson, "Key challenges in automation of earth-moving machines," *Automation in Construction*, vol. 68, pp. 212–222, 2016.

[2] O. Economics, "A global forecast for the construction industry to 2030," 2015.

[3] F. E. Sotiropoulos and H. H. Asada, "A model-free extremum-seeking approach to autonomous excavator control based on output power maximization," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1005–1012, 2019.

[4] R. Fukui, T. Niho, M. Nakao, and M. Uetake, "Imitation-based control of automated ore excavator: improvement of autonomous excavation database quality using clustering and association analysis processes," *Advanced Robotics*, vol. 31, no. 11, pp. 595–606, 2017.

[5] O. Luengo, S. Singh, and H. Cannon, "Modeling and identification of soil-tool interaction in automated excavation," in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)*, vol. 3, pp. 1900–1906, IEEE, 1998.

[6] R. J. Sandzimier and H. H. Asada, "A data-driven approach to prediction and optimal bucket-filling control for autonomous excavators," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2682–2689, 2020.

[7] D. Jud, P. Leemann, S. Kerscher, and M. Hutter, "Autonomous free-form trenching using a walking excavator," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3208–3215, 2019.

[8] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using koopman operators," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019.

[9] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, "Modeling and control of soft robots using the koopman operator and model predictive control," *arXiv preprint arXiv:1902.02827*, 2019.

[10] A. Broad, T. Murphey, and B. Argall, "Learning models for shared control of human-machine systems with unknown dynamics," *arXiv preprint arXiv:1808.08268*, 2018.

[11] Y. Susuki, I. Mezic, F. Raak, and T. Hikihara, "Applied koopman operator theory for power systems technology," *Nonlinear Theory and Its Applications, IEICE*, vol. 7, no. 4, pp. 430–459, 2016.

[12] A. Leonard, J. Rogers, and A. Gerlach, "Koopman operator approach to airdrop mission planning under uncertainty," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 11, pp. 2382–2398, 2019.

[13] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine, "Solar: Deep structured representations for model-based reinforcement learning," in *International Conference on Machine Learning*, pp. 7444–7453, PMLR, 2019.

[14] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.

[15] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop kf: Learning discriminative deterministic state estimators," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[16] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[17] A. Mauroy, Y. Susuki, and I. Mezić, *The Koopman Operator in Systems and Control*. Springer, 2020.

[18] S. Vijayshankar, S. Nabi, A. Chakrabarty, P. Grover, and M. Benosman, "Dynamic mode decomposition and robust estimation: Case study of a 2d turbulent boussinesq flow," in *2020 American Control Conference (ACC)*, pp. 2351–2356, IEEE, 2020.

[19] H. Harry Asada and F. E. Sotiropoulos, "Dual faceted linearization of nonlinear dynamical systems based on physical modeling theory," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, no. 2, 2019.

[20] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[21] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the national academy of sciences of the united states of america*, vol. 17, no. 5, p. 315, 1931.

[22] S. Maćešić and N. Črnjarić-Žic, "Koopman operator theory for nonautonomous and stochastic systems," in *The Koopman Operator in Systems and Control*, pp. 131–160, Springer, 2020.

[23] I. Mezić, "Analysis of fluid flows via spectral properties of the koopman operator," *Annual Review of Fluid Mechanics*, vol. 45, pp. 357–378, 2013.

[24] M. Budišić, R. Mohr, and I. Mezić, "Applied koopmanism," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 4, p. 047510, 2012.

[25] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," *arXiv preprint arXiv:1312.0041*, 2013.

[26] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg, *System dynamics: modeling, simulation, and control of mechatronic systems*. John Wiley & Sons, 2012.

[27] Y. Igarashi, M. Yamakita, J. Ng, and H. H. Asada, "Mpc performances for nonlinear systems using several linearization models," in *2020 American Control Conference (ACC)*, pp. 2426–2431, IEEE, 2020.

[28] F. E. Sotiropoulos and H. H. Asada, "Causality in dual faceted linearization of nonlinear dynamical systems," in *2018 Annual American Control Conference (ACC)*, pp. 1230–1237, IEEE, 2018.

[29] Y. Han, W. Hao, and U. Vaidya, "Deep learning of koopman representation for control," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 1890–1895, IEEE, 2020.

[30] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, pp. 1–10, 2018.

[31] D. Masti and A. Bemporad, "Learning nonlinear state–space models using autoencoders," *Automatica*, vol. 129, p. 109666, 2021.

[32] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for koopman operators of nonlinear dynamical systems," in *2019 American Control Conference (ACC)*, pp. 4832–4839, IEEE, 2019.

[33] G. Mamakoukas, M. Castano, X. Tan, and T. Murphey, "Local koopman operators for data-driven control of robotic systems," in *Robotics: science and systems*, 2019.

[34] T. Kailath, *Linear systems*, vol. 156. Prentice-Hall Englewood Cliffs, NJ, 1980.

[35] P. M. DeRusso, A. A. Desrochers, R. J. Roy, and C. M. Close, *State variables for engineers*. John Wiley & Sons, Inc., 1997.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[37] N. S. Selby, "Dfl." https://github.com/rupumped/DFL, 2021.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[39] M. Servin, T. Berglund, and S. Nystedt, "A multiscale model of terrain dynamics for real-time earthmoving simulation," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 8, p. 11, May 2021.

[40] Y. Yang, J. Pan, P. Long, X. Song, and L. Zhang, "Time variable minimum torque trajectory optimization for autonomous excavator," *arXiv preprint arXiv:2006.00811*, 2020.

[41] Y. Yang, L. Zhang, X. Cheng, J. Pan, and R. Yang, "Compact reachability map for excavator motion planning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2308–2313, IEEE, 2019.

[42] S. Ulin, "Digging deep: A data-driven approach to model reduction in a granular bulldozing scenario," 2018.

[43] S. Backman, D. Lindmark, K. Bodin, M. Servin, J. Mörk, and H. Löfgren, "Continuous control of an underground loader using deep reinforcement learning," 2021.

[44] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Generalizing koopman theory to allow for inputs and control," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 1, pp. 909–930, 2018.