Learning Deep Input-Output Stable Dynamics

Ryosuke Kojima*
Graduate School of Medicine
Kyoto University
Kyoto, 606-8501
kojima.ryosuke.8e@kyoto-u.ac.jp

Yuji Okamoto*
Graduate School of Medicine
Kyoto University
Kyoto, 606-8501
okamoto.yuji.2c@kyoto-u.ac.jp

Abstract

Learning stable dynamics from observed time-series data is an essential problem in robotics, physical modeling, and systems biology. Many of these dynamics are represented as an inputs-output system to communicate with the external environment. In this study, we focus on input-output stable systems, exhibiting robustness against unexpected stimuli and noise. We propose a method to learn nonlinear systems guaranteeing the input-output stability. Our proposed method utilizes the differentiable projection onto the space satisfying the Hamilton-Jacobi inequality to realize the input-output stability. The problem of finding this projection can be formulated as a quadratic constraint quadratic programming problem, and we derive the particular solution analytically. Also, we apply our method to a toy bistable model and the task of training a benchmark generated from a glucose-insulin simulator. The results show that the nonlinear system with neural networks by our method achieves the input-output stability, unlike naive neural networks. Our code is available at `https://github.com/clinfo/DeepIOStability`.

1 Introduction

Learning dynamics from time-series data has many applications such as industrial robot systems [1], physical systems[2], and biological systems [3, 4]. Many of these real-world systems equipped with inputs and outputs to connect for each other, which are called *input-output systems* [5]. For example, biological systems sustain life by obtaining energy from the external environment through their inputs. Such real-world systems have various properties such as stability, controllability, and observability, which provide clues to analyze the complex systems.

Our purpose is to learn a complex system with "desired properties" from a dataset consisting of pairs of input and output signals. To represent the target system, this paper considers the following nonlinear dynamics:

$$\begin{aligned} \dot{x} &= f(x) + G(x)u, \quad x(0) = x_0 \\ y &= h(x). \end{aligned} \tag{1}$$

where the inner state x, the input u, and the output y belong to a signal space that maps time interval $[0, \infty)$ to the Euclidean space. We denote the dimension of x, u, and y as n, m, and l, respectively.

Recently, with the development of deep learning, many methods to learn systems from time-series data using neural networks have been proposed [6, 7, 8]. By representing the maps (f, G, h) in Eq. (1) as neural networks, complex systems can be modeled and trained from a given dataset. However, guaranteeing that a trained system has the desired properties is challenging.

A naively trained system fits the input signals contained in the training dataset, but does not always fit for new input signals. For example, Figure 1 shows our preliminary experiments where we

*Equal contribution.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

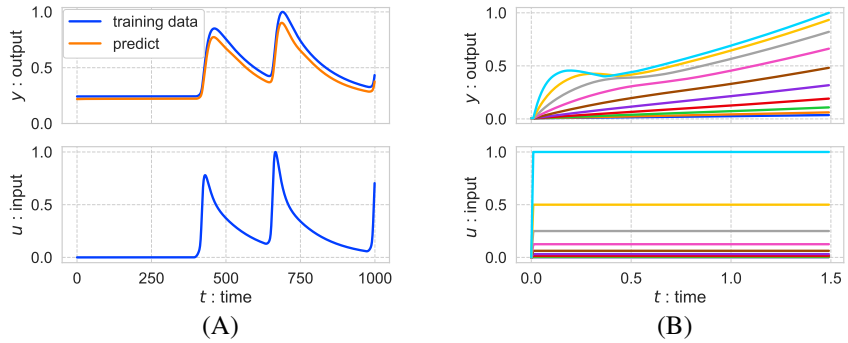


Figure 1: (A) The model prediction of neural networks (B) the reaction of trained model.

naively learned neural networks (f, G, h) in Eq. (1) from input and output signals. The trained neural networks provided small predictive errors for an input signal in the training dataset (Figure 1 (A)). Contrastingly, the unbounded output signals were computed by the trained system with new step input signals (Figure 1 (B)). The reason can be expected that the magnitude (integral value) of this input signal was larger than that of the input signals in the training dataset.

The internal stability is known as one of the attractive properties that should be often satisfied in real-world dynamical systems. The conventional methods to train internal stable systems consisting of neural networks adopt the Lyapunov-based approaches [9, 10, 11]. These methods focus on the internal system, $\dot{x} = f(x)$ in the input-output system (1). Thus, how to learn the entire system (1) with the desired property related to the influence of the input signal is still challenging.

We propose a novel method to learn a dynamical system consisting of neural networks considering the input-output stability. The notion of the input-output stability is often used together with the Hamilton-Jacobi inequality for controller design of the target system in the field of control theory. The Hamilton-Jacobi inequality is one of the sufficient conditions for the input-output stability. The feature of this condition is that the variable for an input signal u does not appear in the expression, i.e., we do not need to evaluate the condition for unknown inputs [12]. To the best of our knowledge, this is the first work that establishes a learning method for the dynamical systems consisting of neural networks using the Hamilton-Jacobi inequality.

The contributions of this paper are as follows:

- This paper derives differentiable projections to the space satisfying the Hamilton-Jacobi inequality.
- This paper presents the learning method for the input-output system proven to always satisfy the Hamilton-Jacobi inequality.
- This paper also provides a loss function derived from the Hamilton-Jacobi inequality. By combining this loss function with the projection described above, efficient learning can be expected.
- This paper presents experiments using two types of benchmarks to evaluate our method.

2 Background

This section describes the $\mathcal{L}_2$ stability, a standard definition of the input-output stability, and the Hamilton-Jacobi inequality.

First, we define the $\mathcal{L}_2$ stability of the nonlinear dynamical system (1). If the norm ratio of the output signal to the input signal is bounded, the system is $\mathcal{L}_2$ stable and this norm ratio is called the $\mathcal{L}_2$ gain. The $\mathcal{L}_2$ norm on the input and output signal space is used for the definition of the $\mathcal{L}_2$ stability. To deal with the $\mathcal{L}_2$ stability on the nonlinear system (1), we assume that the origin $x \equiv 0$ of the internal system $\dot{x} = f(x)$ is an asymptotically stable equilibrium point and $h(0) = 0$. Since the translation of any asymptotically stable equilibrium points is possible, this assumption can be satisfied without loss of generality.

Definition 1. *The $\mathcal{L}_2$ norm is defined as $\|x\|_{\mathcal{L}_2} := \sqrt{\int_0^\infty \|x(t)\|^2 \mathrm{d}t}$. If there exists $\gamma \geq 0$ and a function β on a domain $D \subset \mathbb{R}^n$ such that*

$$\|y\|_{\mathcal{L}_2} \leq \gamma \|u\|_{\mathcal{L}_2} + \beta(x_0), \tag{2}$$

then the system (1) is $\mathcal{L}_2$ stable, where the function $\beta(\cdot)$ is non-negative and $\beta(0) = 0$. Furthermore, the minimum γ satisfying (2) is called the $\mathcal{L}_2$ gain of the nonlinear system (1).

Next, we describe a sufficient condition of the $\mathcal{L}_2$ stability using the Lyapunov function $V : D \to \mathbb{R}$, where the function V is positive definite, i.e., $V(x) \geq 0$ and $V(0) = 0$. The Hamilton-Jacobi inequality is an input-independent sufficient condition.

Proposition 1 ([5, Theorem 5.5])**.** *Let f be locally Lipschitz and let G and h be continuous. If there exist a constant $\gamma > 0$ and a continuously differentiable positive definite function $V : D \subset \mathbb{R}^n \to \mathbb{R}$ such that*

$$\nabla V^{\mathrm{T}}(x) f(x) + \frac{1}{2\gamma^2} \|G^{\mathrm{T}}(x) \nabla V(x)\|^2 + \frac{1}{2} \|h(x)\|^2 \leq 0, \quad \forall x \in D \setminus \{0\}, \tag{3}$$

then the system (1) is $\mathcal{L}_2$ stable and the $\mathcal{L}_2$ gain is less than or equal to γ. This condition is called the Hamilton-Jacobi inequality.

The above proposition can be generalized to allow the more complicated situations such as limit-cycle and bistable cases, where the domain D contains multiple asymptotically stable equilibrium points. The equilibrium point assumed in this proposition can be replaced with positive invariant sets by extending the $\mathcal{L}_2$ norm [13]. Furthermore, by mixing multiple Lyapunov functions, this proposition can be generalized around multiple isolated equilibrium points.

3 Method

The goal of this paper is to learn the $\mathcal{L}_2$ stable system represented by using neural networks (f, G, h). The Hamiltonian-Jacobi inequality, which implies the $\mathcal{L}_2$ stability, is expressed by (f, G, h). We present a method to project (f, G, h) onto the space where the Hamilton-Jacobi inequality holds.

3.1 Modification of nonlinear systems

Supposing $f_{\mathbf{n}} : \mathbb{R}^n \to \mathbb{R}^n$, $G_{\mathbf{n}} : \mathbb{R}^n \to \mathbb{R}^{n \times m}$, and $h_{\mathbf{n}} : \mathbb{R}^n \to \mathbb{R}^l$, a triplet map $(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}})$ denote as *nominal* dynamics. Introducing a distance on the triplet maps, the nearest triplet satisfying the Hamilton-Jacobi inequality from the nominal dynamics $(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}})$ is called modified dynamics $(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}})$. The purpose of this section is to describe the modified dynamics $(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}})$ associated with the nominal dynamics $(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}})$ by analytically deriving a projection onto the space satisfying the Hamilton-Jacobi inequality.

The problem of finding the modified dynamics $(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}})$ is written as a quadratic constraint quadratic programming (QCQP) problem for the nominal dynamics $(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}})$. Since there is generally no analytical solution for QCQP problems, we aim to find the particular solution by adjusting the distance on the triplets.

To prepare for the following theorem, we define the ramp and the clamp functions as

$$\mathrm{R}(x) \triangleq \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}, \quad \mathrm{C}(x; a, b) \triangleq \begin{cases} a, & x \leq a \\ x, & a < x \leq b \\ b, & x > b \end{cases},$$

and define the Hamilton-Jacobi function as

$$\mathrm{HJ}(f, G, h) \triangleq \nabla V^{\mathrm{T}} f + \frac{1}{2\gamma^2} \|G^{\mathrm{T}} \nabla V\|^2 + \frac{1}{2} \|h\|^2, \tag{4}$$

where V is a given positive definite function. A way to design V is to determine the desired positive invariant set and design the increasing function around this set. For example, if the target system has a unique stable point, $x = 0$ can be regarded as the positive invariant set and the increasing function V can be designed as $V(x) = \frac{1}{2} x^2$.

Theorem 1. *Consider that the following optimal problem:*

$$\underset{f_{\mathbf{m}},G_{\mathbf{m}},h_{\mathbf{m}}}{\textbf{minimize}} \quad \frac{(1-k)}{\|\nabla V\|}\|f_{\mathbf{m}}-f_{\mathbf{n}}\|+\frac{k}{2\gamma^2}\|G_{\mathbf{m}}-G_{\mathbf{n}}\|^2+\frac{k}{\|\nabla V\|^2}\|h_{\mathbf{m}}-h_{\mathbf{n}}\|^2 \tag{5a}$$

$$\textbf{subject to} \quad \mathrm{HJ}(f_{\mathbf{m}},G_{\mathbf{m}},h_{\mathbf{m}})\leq 0, \tag{5b}$$

where $k\in[0,1]$. The solution of (5) is given by

$$f_{\mathbf{m}}=f_{\mathbf{n}}-\frac{1}{\|\nabla V\|^2}\mathrm{R}\left(V_f+k^2V_{Gh}\right)\nabla V,$$

$$G_{\mathbf{m}}=G_{\mathbf{n}}-\left(1-\sqrt{\mathrm{C}\left(-\tfrac{V_f}{V_{Gh}};k^2,1\right)}\right)P_VG_{\mathbf{n}},$$

$$h_{\mathbf{m}}=\sqrt{\mathrm{C}\left(-\tfrac{V_f}{V_{Gh}};k^2,1\right)}h_{\mathbf{n}},$$

where

$$V_f\triangleq\nabla V^{\mathrm{T}}f_{\mathbf{n}},\quad V_{Gh}\triangleq\frac{1}{2\gamma^2}\|G_{\mathbf{n}}^{\mathrm{T}}\nabla V\|^2+\frac{1}{2}\|h_{\mathbf{n}}\|^2,\quad P_V\triangleq\frac{\nabla V\nabla V^{\mathrm{T}}}{\|\nabla V\|^2}.$$

Proof: See Appendix A. □

The objective function of (5a) is a new distance between the nominal dynamics $(f_{\mathbf{n}},G_{\mathbf{n}},h_{\mathbf{n}})$ and the modified dynamics $(f_{\mathbf{m}},G_{\mathbf{m}},h_{\mathbf{m}})$. This new distance allows the derivation of analytical solutions by combining three distances of f, G, and h.

Focusing on the objective function (5a), the constant k represents the ratio of the distance scale of f to g and h. When $k=0$, the problem (5) is consistent with the optimization problem for the internal stability [9]. As the constant k converges 1, the modification method of Theorem 1 approaches $G_{\mathbf{m}}$ and $h_{\mathbf{m}}$ to $G_{\mathbf{n}}$ and $h_{\mathbf{n}}$, respectively. In this case, the objective function (5a) becomes the distance between $f_{\mathbf{m}}$ to $f_{\mathbf{n}}$. Therefore, the following corollary is satisfied.

Corollary 1. *The solution of*

$$\underset{f_{\mathbf{m}}}{\textbf{minimize}} \quad \|f_{\mathbf{m}}-f_{\mathbf{n}}\| \tag{6a}$$

$$\textbf{subject to} \quad \mathrm{HJ}(f_{\mathbf{m}},G_{\mathbf{n}},h_{\mathbf{n}})\leq 0, \tag{6b}$$

is given by

$$f_{\mathbf{m}}=f_{\mathbf{n}}-\frac{1}{\|\nabla V\|^2}\mathrm{R}\left(\mathrm{HJ}(f_{\mathbf{m}},G_{\mathbf{m}},h_{\mathbf{m}})\right)\nabla V.$$

Proof: This solution is easily derived from Theorem 1. □

Corollary 1 derives a solution to a linear programming problem rather than QCQP problems. Replacing the Hamilton-Jacobi function $\mathrm{HJ}(f_{\mathbf{m}},G_{\mathbf{n}},h_{\mathbf{n}})$ with the time derivative of a positive definite function $\nabla V f$, this corollary matches the result of the conventional study [9].

When the map $h_{\mathbf{m}}$ is fixed as $h_{\mathbf{n}}$, a similar solution as Theorem 1 is derived. Although Corollary 1 is proved by changing k, the modified dynamics with the fixed $h_{\mathbf{n}}$ are not derived. We reprove this modified dynamics in a similar way to Theorem 1.

Corollary 2. *Consider the following problem:*

$$\underset{f_{\mathbf{m}},G_{\mathbf{m}}}{\textbf{minimize}} \quad \frac{(1-k)}{\|\nabla V\|}\|f_{\mathbf{m}}-f_{\mathbf{n}}\|+\frac{k}{2\gamma^2}\|G_{\mathbf{m}}-G_{\mathbf{n}}\|^2 \tag{7a}$$

$$\textbf{subject to} \quad \mathrm{HJ}(f_{\mathbf{m}},G_{\mathbf{m}},h_{\mathbf{n}})\leq 0, \tag{7b}$$

where $k\in[0,1]$. The solution of (7) is given by

$$f_{\mathbf{m}}=f_{\mathbf{n}}-\frac{1}{\|\beta\|^2}\mathrm{R}\left(V_{fh}+k^2V_G\right)\nabla V,\quad G_{\mathbf{m}}=G_{\mathbf{n}}-\left(1-\sqrt{\mathrm{C}\Big(-\tfrac{V_{fh}}{V_G};k^2,1\Big)}\right)P_VG_{\mathbf{n}}, \tag{8}$$

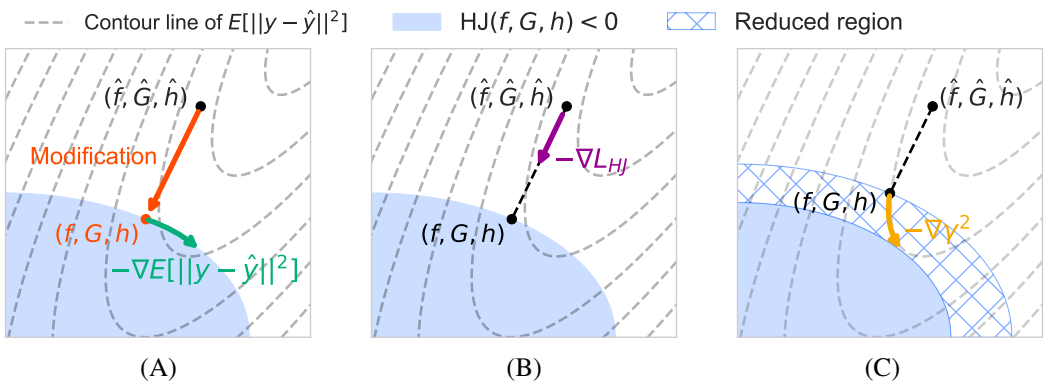


Figure 2: Sketches of our method : (A) minimizing the prediction error (the first term of the loss function (9)) in the blue region, (B) moving the nominal dynamics to the blue region (the second term), and (C) reducing the blue region while keeping the same level of the prediction error (the last term).

where

$$V_{fh} \triangleq \nabla V^{\mathrm{T}} f_{\mathbf{n}} + \frac{1}{2}\|h_{\mathbf{n}}\|^2, \quad V_G \triangleq \frac{1}{2\gamma^2}\|G_{\mathbf{n}}^{\mathrm{T}} \nabla V\|^2.$$

Proof: See Appendix A. □

3.2 Loss function

We represent $(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}})$ as neural networks and denote $(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}})$ as the $\mathcal{L}_2$ stable dynamics modified by Theorem 1, Corollary 1, or 2. Note that the modification depends on the candidate of $\mathcal{L}_2$ gain γ. Figure 2 (A) shows the sketch of this modification, where the blue region satisfies the Hamilton-Jacobi inequality.

Since the nonlinear system of the modified dynamics $(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}})$ is represented as ordinary differential equations (ODEs) consisting of the differentiable functions $f_{\mathbf{n}}$,$G_{\mathbf{n}}$, and $h_{\mathbf{n}}$, the techniques of training neural ODEs can be applied [14]. Once a loss function is designed, the parameters of neural networks in the modified system can be learned from given data.

$$\mathrm{Loss} = \mathrm{E}_{(x_0,u,\hat{y})\in\mathcal{D}}[||y-\hat{y}||^2_{\mathcal{L}_2}] + \lambda L_{\mathrm{HJ}} + \alpha\gamma^2, \tag{9}$$

where λ and α are non-negative coefficients.

The first term shows the prediction error of the output signal y (Figure 2 (A)). A dataset $\mathcal{D}$ consists of tuples (x_0, u, y) where the initial value x_0, the input signal u, and the output signal y. The predicted output $\hat{y}$ is calculated from x_0, u, and the modified dynamics $(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}})$.

The second term aims to improve the nominal dynamics $(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}})$ closer to the modified dynamics $(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}})$ and is defined as

$$L_{\mathrm{HJ}} = \mathrm{E}_x[\mathrm{R}(\mathrm{HJ}(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}})(x) + \varepsilon)],$$

where ε is a positive constant (Figure 2 (B)). Since this term is a form of the hinge loss, ε represents the magnitude of the penalties for the Hamilton-Jacobi inequality. To evaluate this inequality for any $x \in D$, we introduce a distribution of x over the domain D. In our experiments, this distribution is decided as a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ where the mean μ is placed at the asymptotically stable point and the variance σ^2 is an experimental parameter.

The modifying parameter γ can be manually designed for the application or automatically trained from data by introducing the last term. This training explores smaller γ while keeping the same level of the prediction error (Figure 2 (C)).

4 Related work

Estimating parameters of a given system is traditionally studied as system identification. In the field of system identification, much research on the identification of linear systems have been done, where the maps (f, G, h) in Eq. (1) assumes to be linear [15]. Linear state-space models include identification methods by impulse response like the Eigensystem Realization Algorithm (ERA)[16], by the state-space representation like Multivariable Output Error State sPace (MOESP)[17] and ORThogonal decomposition method (ORT)[18]. System identification methods for non-state-space models, unlike our target system, contain Dynamic Mode Decomposition with control (DMDc)[19] and its nonlinear version, Sparse Identification of Nonlinear DYnamics with control (SINDYc)[20]. Also, the nonlinear models such as the nonlinear ARX model [21] and the Hammerstein-Wiener model [22] have been developed and often trained by error minimization. For example, the system identification method using a piece-wise ARX model allows more complicated functions by relaxing the assumption of linearity [21].

These traditional methods often concern gray-box systems, where the maps (f, G, h) in Eq. (1) are partially known [23]. This paper deals with a case of black-box systems, where f, G, and h in the system (1) are represented by neural networks. Our method can be used regardless of whether all f, G, and h functions are parameterized using neural networks. So, our method can be easily applied to application-dependent gray-box systems when the functions f, G, and h are differentiable with respect to the parameters.

Because the input-output system (1) can be regarded as a differential equation, our study is closely related to the method of combining neural networks and ODEs [7, 14]. These techniques have been improved in recent years, including discretization errors and computational complexity. Although we used an Euler method for simplicity, we can expect that learning efficiency would be further improved by using these techniques.

The internal stability is a fundamental property in ODEs, and learning a system with this property using neural networks plays an important role in the identification of real-world systems [24]. In particular, the first method to guarantee the internal stability of the trained system has been proposed in [9]. Furthermore, another method [25] extends this method to apply positive invariant sets, e.g. limit cycles and line attractors. Encouraged by these methods based on the Lyapunov function, our method further generalizes these methods using the Hamilton-Jacobi inequality to guarantee the input-output stability.

In this paper, the Lyapunov function V is considered to be given, but this function can be learned from data. Since a pair of dynamics (1) and V has redundant degrees of freedom, additional assumptions are required to determine V uniquely. In [9], it is realized by limiting the dynamics to the internal system and restricting V to a convex function [26].

Lyapunov functions are also used to design controllers, where the whole system should satisfy the stability condition. A method for learning such a controller using neural nets has been proposed [27]. This method deals with optimization problems over the space that satisfies the stability condition, which is similar to our method. Whereas we solve the QCQP problem to derive the projection onto the space, this method uses a Satisfiability Modulo Theories (SMT) solver to satisfy this condition. The method has also been extended to apply unknown systems [28].

Although this paper deals with deterministic systems, neural networks for stochastic dynamics with the variational inference is well studied [6, 29, 30, 31]. Guaranteeing the internal stability of trained stochastic dynamics is important for noise filtering and robust controller design [10].

5 Experiments

We conduct two experiments to evaluate our proposed method. The first experiment uses a benchmark dataset generated from a nonlinear model with multiple asymptotically equilibrium points. In the next experiment, we applied our method to a biological system using a simulator of the glucose-insulin system.

5.1 Experimental setting

Before describing the results of the experiments, this section explains the evaluation metrics and our experimental setting.

For evaluation metrics, we define the root mean square error (RMSE) and the average $\mathcal{L}_2$ gain (GainIO) for the given input and output signals as follows:

$$\text{RMSE} \triangleq \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|y_i - \hat{y}_i\|_{\mathcal{L}_2}^2}, \quad \text{GainIO} \triangleq \frac{1}{N}\sum_{i=1}^{N}\frac{\|\hat{y}_i\|_{\mathcal{L}_2}}{\|u_i\|_{\mathcal{L}_2}},$$

where N is the number of signals in the dataset. $u_i(\cdot)$ and $y_i(\cdot)$ are the input and output signal at the i-th index, respectively. The prediction signal $\hat{y}_i(\cdot)$ is computed from $u_i(\cdot)$, the trained dynamics, and the initial state. Note that the integral contained in the $\mathcal{L}_2$ norm is approximated by a finite summation. The RMSE is a metric of the prediction errors related to the output signal, and the GainIO is a metric of the property of the $\mathcal{L}_2$ stability. Whether the target system satisfies or does not satisfy the $\mathcal{L}_2$ stability, the GainIO with a given finite-size dataset can be calculated. The GainIO error is defined by the absolute error between the GainIO of the test dataset and that of the prediction.

In our experiments, 90% of the dataset is used for training and the remaining 10% is used for testing. We retry five times for all experiments and show the mean and standard deviations of the metrics.

For simplicity in our experiments, the sampling step Δt for the output y is set as constant and the Euler method is used to solve ODEs. x_0 is put at an asymptotically stable point for each benchmark and is known. In this experiment, to prevent the state from diverging during learning of dynamics, the clipping operation is used so that the absolute values of the states are less than ten.

For comparative methods, we use vanilla neural networks, ARX, ORT [18], MOESP [17], and piece-wise ARX[21]. In the method of vanilla neural networks, the maps (f, G, h) in the nonlinear system (1) is represented by using three neural networks, i.e., this method is consistent with a method used in Figure 1. To determine the hyperparameters of comparative methods except for neural networks, the grid search is used. Note that these comparative methods only consider the prediction errors.

For training each method with neural networks, an NVIDIA Tesla T4 GPU was used. Our experiments are totally run on 20 GPUs over about three days.

5.2 Bistable model benchmark

The first experiment is carried out using a bistable model, which is known as a bounded system with multiple asymptotically equilibrium points. This bistable model is defined as

$$\dot{x} = x(1 - x^2) + u, \quad x(0) = -1, \quad y = x.$$

The internal system of this model has two asymptotically stable equilibrium points $x \equiv 1, -1$.

We generate 1000 input and output signals for this experiment. To construct this dataset, we prepare input signals using positive and negative pulse wave signals whose pulse width is changed at random. The input and output signals on the period $[0, 10]$ are sampled with an interval $\Delta t = 0.1$. In this benchmark, we set the number of dimensions of the internal system as one and use a fixed function $V(x) = \min((x-1)^2, (x+1)^2)$, a mixture of the two positive definite functions.

In the result of our experiments, we name the proposed methods modified by Theorem 1, Corollary 1, and 2 as DIOS-fgh, DIOS-f, and DIOS-fg, respectively. In these methods, the parameters of our loss function are set as $\lambda = 0$ and $\alpha = 0.01$. Also, DIOS-fgh+ uses $\lambda = 0.01$ and $\alpha = 0.01$ under the same conditions as DIOS-fgh. For this example (1000 input signal), it took about 1 hour using 1 GPU to learn one model training.

The results of the RMSE and the GainIO error in this experiment are shown in Figure 3. Figures 3 (A) and (B) demonstrate that a piece-wise ARX model (PWARX) gives very low RMSE but its GainIO error is high. Our proposed methods achieve a small GainIO error while keeping the RMSE. Note that linear models such as MOESP and ORT only approach one point although the bistable model has two asymptotically stable equilibrium points. Figures 3 (C) and (D) display the effect of dataset sizes. When the dataset size was varied to 100, 1000, and 3000, the result of the larger dataset provided

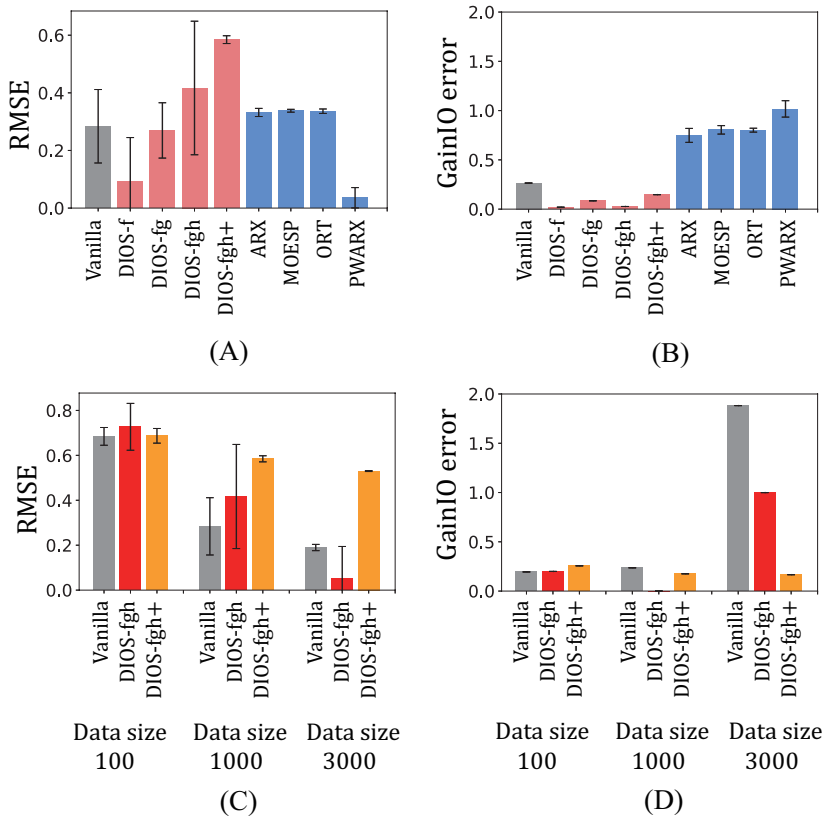


Figure 3: Results of the bistable model benchmark. The upper part shows (A) the RMSE and (B) the GainIO error of the vanilla neural networks (gray), our proposed methods (red), and the conventional methods (blue). The lower part shows (C) the RMSE and (D) the GainIO error for the different sizes of the datasets.

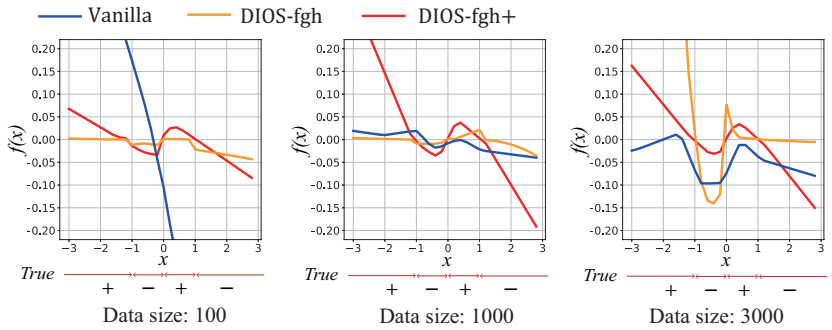


Figure 4: The sketch x-$f(x)$ displaying the internal dynamics trained from the bistable datasets with the different sizes. The sign of the bistable model is shown at the bottom of each figure.

the smaller RMSE for all methods. The vanilla neural networks do not consider the $\mathcal{L}_2$ gain, so the GainIO error was large in the case of the low RMSE.

Figure 4 shows the relationship between x and $f(x)$ in the trained system in (1) to compare the vanilla neural networks, DIOS-fgh and DIOS-fgh+. DIOS-fgh and DIOS-fgh+ successfully find two stable points, i.e., the trained function $f(x)$ had roots of two stable points $x = \pm 1$ and an unstable point $x = 0$. Especially, these stable points were robustly estimated in the trained system using our presented loss function (DIOS-fgh+) even when the dataset size was small. The vanilla neural networks failed to obtain the two stable points in all cases.

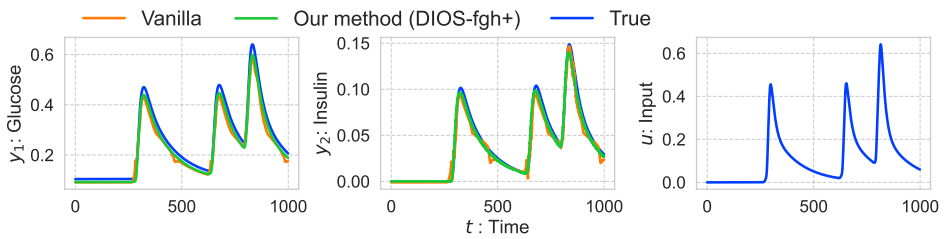
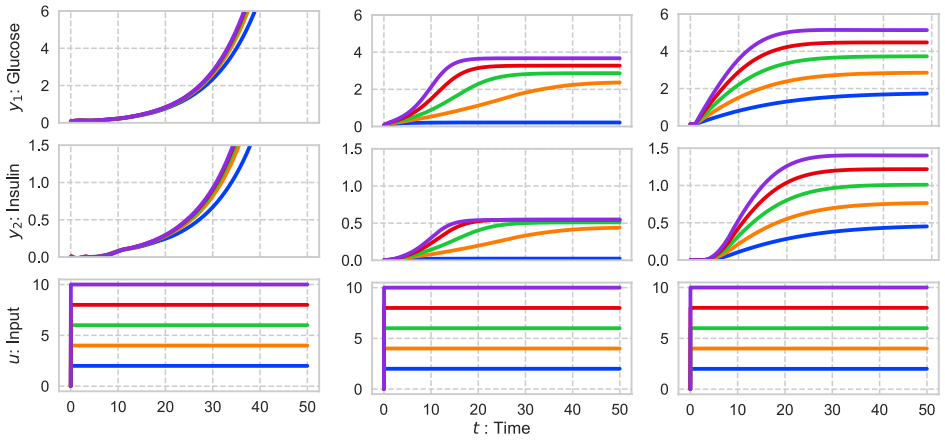


Figure 5: The input and output signals of the glucose-insulin simulator and the predicted output.



(A) Vanilla (B) Our method (DIOS-fgh+) (B) True

Figure 6: The step reaction of the trained systems. The color of each line indicates the magnitude of the step input.

5.3 Glucose-insulin benchmark

This section addresses an example of the identification of biological systems. We consider the task of learning the glucose-insulin system using a simulator [32] to construct responses for various inputs and evaluate the robustness of the proposed method for unexpected inputs. This simulator outputs the concentrations of plasma glucose y_1 and insulin y_2 for the appearance of plasma glucose per minute u. To determine the realistic input u, we adopt another model, an oral glucose absorption model [33].

Using this simulator, 1000 input and output signals are synthesized for this experiment. The input and output signals are sampled with a sampling interval $\Delta t = 1$ and 1000 steps for each sequence. In this benchmark, we set the number of dimensions of the internal system as six, fix a positive definite function $V(x) = x^2$, and use our loss function with $\lambda = 0.001$ and $\alpha = 0.001$. Training one model for this examples (1000 input signal) tooks about 7.5hours using 1GPU. The hyperparameters including the number of layers in the neural networks, the learning rate, optimizer, and the weighted decay are determined using the tree-structured Parzen estimator (TPE) implemented in Optuna [34].

The RMSE of vanilla and our proposed method are 0.0103 and 0.0050, respectively. So, from the perspective of RMSE, these methods achieved almost the same performance. Figure 5 shows input and output signals in the test dataset and the predicted output by vanilla neural networks and our method (DIOS-fgh+). The $\mathcal{L}_2$ stability of the system using the vanilla neural networks is not guaranteed. Since the proposed method guarantees the $\mathcal{L}_2$ stability, the output signals of DIOS-fgh+ are bounded even if the input signals are unexpectedly large.

To demonstrate this, we conducted an additional experiment using the trained system. Figure 6 shows the transition of output behavior caused by the magnitude of the input signal changing from 2 to 10. Note that the maximum magnitude in the training dataset is one. In this experiment, Δt was changed to 0.01 and the clipping operation was removed to deal with the large values of the state.

This result shows the output of the vanilla neural networks quickly diverged with an unexpectedly large input. Contrastingly, the output behavior of our proposed method is always bounded. Therefore, we actually confirmed that our proposed method satisfies the $\mathcal{L}_2$ stability.

6 Conclusion

This paper proposed a learning method for nonlinear dynamical system guaranteeing the $\mathcal{L}_2$ stability. By theoretically deriving the projection of a triplet (f, G, h) to the space satisfying the Hamilton-Jacobi inequality, our proposed method realized the $\mathcal{L}_2$ stability of trained systems. Also, we introduced a loss function to empirically achieve a smaller $\mathcal{L}_2$ gain while reducing prediction errors. We conducted two experiments to learn dynamical systems consisting of neural networks. The first experiment used a nonlinear model with multiple asymptotically equilibrium points. The result of this experiment showed that our proposed method can robustly estimate a system with multiple stable points. In the next experiment, we applied our method to a biological system using a simulator of the glucose-insulin system. It was confirmed that the proposed method can successfully learn a proper system that works well under unexpectedly large inputs due to the $\mathcal{L}_2$ stability. There is a limitation that our method cannot apply the system without the $\mathcal{L}_2$ stability. Future work will expand the $\mathcal{L}_2$ stability-based method to a more generalized learning method by dissipativity and apply our approach of this study to stochastic systems.

Acknowledgements

This research was supported by JST Moonshot R&D Grant Number JPMJMS2021 and JPMJMS2024. This work was also supported by JSPS KAKENHI Grant No.21H04905. This paper was also based on a part of results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] Jan Swevers, Walter Verdonck, and Joris De Schutter. Dynamic model identification for industrial robots. *IEEE control systems magazine*, 27(5):58–71, 2007.

[2] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 2016.

[3] Timothy S. Gardner, Diego Di Bernardo, David Lorenz, and James J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105, 2003.

[4] Geoffrey Roeder, Paul K Grant, Andrew Phillips, Neil Dalchau, and Edwards Meeds. Efficient amortised bayesian inference for hierarchical and nonlinear dynamical systems. *Proceeding of International Conference on Machine Learning (ICML)*, 2019.

[5] Hassan K. Khalil. *Nonlinear systems*. Prentice-Hall, 3rd edition, 2002.

[6] Rahul G. Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, page 2101–2109, 2017.

[7] Ricky T.Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[8] Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. In *Proceeding of International Conference on Learning Representations*, 2019.

[9] Gaurav Manek and J. Zico Kolter. Learning stable deep dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[10] Nathan Lawrence, Philip Loewen, Michael Forbes, Johan Backstrom, and Bhushan Gopaluni. Almost surely stable deep dynamics. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[11] Andreas Schlaginhaufen, Philippe Wenk, Andreas Krause, and Florian Dörfler. Learning stable deep dynamics models for partially observed or delayed dynamical systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

[12] Arjan van der Schaft. *$\mathcal{L}_2$-Gain and Passivity Techniques in Nonlinear Control*. Springer Publishing Company, Incorporated, 3rd edition, 2016.

[13] W.M. Haddad and V.S. Chellaboina. *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*. Princeton University Press, 2008.

[14] Xinshi Chen. Review: Ordinary differential equations for deep learning. *CoRR*, abs/1911.00502, 2019.

[15] Tohru Katayama. *Subspace Method for System Identification*. Springer, 2005.

[16] Jer-Nan Juang and Richard S Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of guidance, control, and dynamics*, 8(5):620–627, 1985.

[17] Michel Verhaegen and Patrick Dewilde. Subspace model identification part 1. the output-error state-space model identification class of algorithms. *International journal of control*, 56(5):1187–1210, 1992.

[18] Tohru Katayama, Hideyuki Tanaka, and Takeya Enomoto. A simple subspace identification method of closed-loop systems using orthogonal decomposition. *IFAC Proceedings Volumes*, 38(1):512–517, 2005.

[19] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

[20] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Sparse identification of nonlinear dynamics with control (SINDYc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.

[21] Andrea Garulli, Simone Paoletti, and Antonio Vicino. A survey on switched and piecewise affine system identification. In *IFAC Proceedings Volumes*, volume 45, pages 344–355. IFAC, 2012.

[22] Adrian Wills, Thomas B. Schon, Lennart Ljung, and Brett Ninness. Identification of hammerstein-wiener models. *Automatica*, 49(1):70–81, 2013.

[23] Lennart Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.

[24] Martin Braun and Martin Golubitsky. *Differential equations and their applications*. Springer, 1983.

[25] Naoya Takeishi and Yoshinobu Kawahara. Learning dynamics models with stable invariant sets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9782–9790, 2021.

[26] Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155, 2017.

[27] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[28] Vrushabh Zinage and Efstathios Bakolas. Neural koopman lyapunov control. *arXiv:2201.05098*, 2022.

[29] Daniel Gedon, Niklas Wahlström, Thomas B. Schön, and Lennart Ljung. Deep state space models for nonlinear system identification. In *Proceedings of the 19th IFAC Symposium on System Identification (SYSID)*, 2021.

[30] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

[31] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *ArXiv*, abs/1411.7610, 2014.

[32] A. De Gaetano and O. Arino. A statistical approach to the determination of stability for dynamical systems modelling physiological processes. *Mathematical and Computer Modelling*, 31(4):41–51, 2000.

[33] Chiara Dalla Man, Robert A. Rizza, and Claudio Cobelli. Meal simulation model of the glucose-insulin system. *IEEE Transactions on Biomedical Engineering*, 54(10):1740–1749, 2007.

[34] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section 2.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] The paper conforms with the provided guidelines.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2
   (b) Did you include complete proofs of all theoretical results? [Yes] See the supplemental material (Appendix.pdf)
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the supplemental material. Our codes contains README.md to reproduce the our experimental results.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.1
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [Yes] See the supplemental materials. Our codes contains LICENSE.txt.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See the supplemental materials containing our source codes
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Our dataset was reproduced by ourselves based on the literature
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof of Theorem 1 and Corollary 2

To prove Theorem 1 and Corollary 2, we use the following particular solution of QCQP problems.

Lemma 1. *Suppose that $A \in \mathbb{R}^{n\times n}$ is a positive definite matrix, a vector $b \in \mathbb{R}^n$, a scalar c, and positive constants k_{x}, k_{y}. The solution of this problem*

$$\begin{aligned} &\textbf{minimize} && k_{\mathrm{x}} x^{\mathrm{T}} A x + k_{\mathrm{y}}|y| \\ &\textbf{subject to} && y \geq x^{\mathrm{T}} A x - 2b^{\mathrm{T}} x + c. \end{aligned}$$

is given by

$$\begin{aligned} x^* &= \left(1 - \sqrt{\mathrm{C}(1-\tilde{c}; \tilde{k}_{\mathrm{x}}^2, 1)}\right) A^{-1} b, \\ y^* &= \mathrm{R}\left(c - (1-\tilde{k}_x^2) b^{\mathrm{T}} A^{-1} b\right), \end{aligned}$$

where

$$\tilde{c} = \frac{c}{b^{\mathrm{T}} A^{-1} b}, \quad \tilde{k}_{\mathrm{x}} = \frac{k_{\mathrm{x}}}{k_{\mathrm{x}} + k_{\mathrm{y}}}.$$

Proof: See Appendix B. □

To keep the notation short in the following proofs, we define

$$\beta \triangleq \nabla V.$$

Proof of Theorem 2: For the Hamilton-Jacobi function $\mathrm{HJ}(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}})$ of the modified dynamics $(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}})$, the map $f_{\mathbf{m}}$ and each row of the map $G_{\mathbf{m}}$ don't depend on the orthogonal vector of β. Hence, $f_{\mathbf{m}}$ and $G_{\mathbf{m}}$ are written as

$$f_{\mathbf{m}} = f_{\mathbf{n}} - \beta p, \quad G_{\mathbf{m}} = G_{\mathbf{n}} - \beta q^{\mathrm{T}},$$

where p is scalar and $q \in \mathbb{R}^m$. $h_{\mathbf{m}}$ depends only on the norm in $\mathrm{HJ}(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}})$, i. e.,

$$h_{\mathbf{m}} = (1-r)hn,$$

where r is scalar.

The optimal function (5a) is rewritten as the p, q, and r, i.e.,

$$\begin{aligned} &\frac{k_1}{\|\beta\|}\|f_{\mathbf{m}} - f_{\mathbf{n}}\| + \frac{k_2}{2\gamma^2}\|G_{\mathbf{m}} - G_{\mathbf{n}}\|^2 + \frac{k_2}{2\|\beta\|^2}\|h_{\mathbf{m}} - h_{\mathbf{n}}\| \\ &= k_1|p| + k_2\frac{\|\beta\|^2}{2\gamma^2}\|q\|^2 + k_2\frac{\|h_{\mathbf{n}}\|}{2\|\beta\|^2}r^2. \end{aligned}$$

Also, the condition of constraint (5b) is rewritten as

$$\begin{aligned} &\mathrm{HJ}_\beta(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}}) \\ &= \beta^{\mathrm{T}} f_{\mathbf{m}} + \frac{1}{2\gamma^2}\|G_{\mathbf{m}}^{\mathrm{T}}\beta\|^2 + \frac{1}{2}\|h_{\mathbf{m}}\|^2 \\ &= \beta^{\mathrm{T}}(f_{\mathbf{n}} - \beta p) + \frac{1}{2\gamma^2}\|(G_{\mathbf{n}} - \beta q^{\mathrm{T}})^{\mathrm{T}}\beta\|^2 + \frac{1}{2}\Big\|(1-r)h_{\mathbf{n}}\|^2 \\ &= -\|\beta\|^2 p + \frac{1}{2\gamma^2}\|\beta\|^4\|q\|^2 + \frac{1}{2}\|\hat{h}\|^2 r^2 - \frac{1}{\gamma^2}\|\beta\|^2\beta^{\mathrm{T}} G_{\mathbf{n}} q - \|h_{\mathbf{n}}\|^2 r + \mathrm{HJ}_\beta(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}}) \\ &\leq 0. \end{aligned}$$

If $x \triangleq [q^{\mathrm{T}}, r]^{\mathrm{T}}$, $y \triangleq p$, and

$$A \triangleq \begin{bmatrix} \frac{\|\beta\|^2}{2\gamma^2} I_m & 0 \\ 0 & \frac{\|h_{\mathbf{n}}\|^2}{2\|\beta\|^2} \end{bmatrix}, \quad b \triangleq \begin{bmatrix} \frac{1}{2\gamma^2} G_{\mathbf{n}}^{\mathrm{T}}\beta \\ \frac{\|h_{\mathbf{n}}\|^2}{2\|\beta\|^2} \end{bmatrix} \quad c \triangleq \frac{1}{\|\beta\|^2}\Big(\mathrm{HJ}_\beta(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}})\Big),$$

then this optimal problem (5) of this theorem becomes a problem used in Lemma 1. The optimal point of p, q and r is given by

$$
\begin{aligned}
p^* &= \frac{1}{\|\beta\|^2}\mathrm{R}\left(V_{\hat{f}} + \hat{k}^2 V_{G_{\mathbf{n}},h_{\mathbf{n}}}\right),\\
q^* &= \frac{1}{\|\beta\|^2}\left(1-\sqrt{\mathrm{C}\Big(-\tfrac{V_{f_{\mathbf{n}}}}{V_{G_{\mathbf{n}},h_{\mathbf{n}}}};\tilde{k}^2,1\Big)}\right)G_{\mathbf{n}}^{\mathrm{T}}\beta,\\
r^* &= 1-\sqrt{\mathrm{C}\Big(-\tfrac{V_{f_{\mathbf{n}}}}{V_{G_{\mathbf{n}},h_{\mathbf{n}}}};\tilde{k}^2,1\Big)}.
\end{aligned}
$$

Therefore, Theorem 1 is derived.

Proof of Theorem 2:

Optimal maps $f_{\mathbf{m}}$ and $G_{\mathbf{m}}$ are also written as

$$
f_{\mathbf{m}} = f_{\mathbf{n}} - \beta p, \quad G_{\mathbf{m}} = G_{\mathbf{n}} - \beta q^{\mathrm{T}}.
$$

Hence, the objective function (7a) is rewritten as p and q function, i.e.,

$$
\frac{k_1}{\|\beta\|}\|f_{\mathbf{m}} - f_{\mathbf{n}}\| + \frac{k_2}{2\gamma^2}\|G_{\mathbf{m}} - G_{\mathbf{n}}\|^2 = k_1|p| + k_2\frac{\|\beta\|^2}{2\gamma^2}\|q\|^2.
$$

Also the condition of constraint (7b) is written

$$
\begin{aligned}
&\mathrm{HJ}_\beta(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{n}})\\
&= \beta^{\mathrm{T}} f_{\mathbf{m}} + \frac{1}{2\gamma^2}\|G_{\mathbf{m}}^{\mathrm{T}}\beta\|^2 + \frac{1}{2}\|h_{\mathbf{m}}\|^2\\
&= \beta^{\mathrm{T}}(f_{\mathbf{n}} - \beta p) + \frac{1}{2\gamma^2}\|(G_{\mathbf{n}} - \beta q^{\mathrm{T}})^{\mathrm{T}}\beta\|^2 + \frac{1}{2}\|h_{\mathbf{n}}\|^2\\
&= -\|\beta\|^2 p + \frac{1}{2\gamma^2}\|\beta\|^4\|q\|^2 - \frac{1}{\gamma^2}\|\beta\|^2\beta^{\mathrm{T}}G_{\mathbf{n}}q + \mathrm{HJ}_\beta(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}})\\
&\le 0.
\end{aligned}
$$

If $x \triangleq q$, $y \triangleq p$, and

$$
A \triangleq \frac{\|\beta\|^2}{2\gamma^2}I_m, \quad b \triangleq \frac{1}{2\gamma^2}G_{\mathbf{m}}^{\mathrm{T}}\beta, \quad c \triangleq \frac{1}{\|\beta\|^2}\Big(\mathrm{HJ}_\beta(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}})\Big),
$$

then this optimal problem (7) becomes the problem used in Lemma 1. The optimal points of p and q are given by

$$
\begin{aligned}
p^* &= \frac{1}{\|\beta\|^2}\mathrm{R}\left(V_{f_{\mathbf{n}},h_{\mathbf{n}}} + \hat{k}_2^2(V_{G_{\mathbf{n}}})\right),\\
q^* &= \frac{1}{\|\beta\|^2}\left(1-\sqrt{\mathrm{C}\Big(-\tfrac{V_{f_{\mathbf{n}},h_{\mathbf{n}}}}{V_{G_{\mathbf{n}}}};\tilde{k}^2,1\Big)}\right)G_{\mathbf{n}}^{\mathrm{T}}\beta,
\end{aligned}
$$

Therefore, the solution of problem (7) becomes Eqs. (8).

B Proof of the particular solution of QCQP

This section presents the proof of the following QCQP problem.

$$
\begin{aligned}
&\textbf{minimize} && k_{\mathrm{x}}x^{\mathrm{T}}Ax + k_{\mathrm{y}}|y| && \text{(10a)}\\
&\textbf{subject to} && y \ge x^{\mathrm{T}}Ax - 2b^{\mathrm{T}}x + c, && \text{(10b)}
\end{aligned}
$$

where A is a positive definite matrix. We classify this problem according to the parameter A, b and c.

First, the solution of this optimal problem is switched depending on the positive or negative value of c.

Lemma 2. *If $c \leq 0$ then the solution of Eq. (10) is $(x, y) = (0, 0)$. If $c > 0$, the solution x^* of Eq. (10) equals the solution of the following problem:*

$$\textbf{minimize} \quad k_x x^{\mathrm{T}} A x + k_y |x^{\mathrm{T}} A x - 2b^{\mathrm{T}} x + c|. \tag{11}$$

Furthermore, the solution y^* is given by

$$y^* = k_x x^{*\mathrm{T}} A x^* - 2b^{\mathrm{T}} x^* + c. \tag{12}$$

Proof: The objective function (10a) is strictly convex and the minimum point is $(x, y) = (0, 0)$. If $c \leq 0$, the origin $(x, y) = (0, 0)$ satisfies the condition of constraint (10b). Therefore, the solution of Eq. (10) is $(x, y) = (0, 0)$.

If $c > 0$, $(x, y) = (0, 0)$ does not satisfies the constraint condition (10b), and the optimal solution belong the boundary of the region satisfying (10b). Therefore, the solution point (x^*, y^*) satisfies Eq. (12). □

Also, the solution of the new optimal problem (11) is switched by the ratio between c and $b^{\mathrm{T}} A^{-1} b$.

Lemma 3. *If $\tilde{c} > 1 - \tilde{k}_x^2$ then the solution x^* of the optimal problem (11) is given by*

$$x^* = (1 - \tilde{k}_x) A^{-1} b, \tag{13}$$

where

$$\tilde{c} = \frac{c}{b^{\mathrm{T}} A^{-1} b}, \quad \tilde{k}_x = \frac{k_x}{k_x + k_y}.$$

If $\tilde{c} \leq 1 - \tilde{k}_x^2$, the solution x^* equals the solution of the following QCQP problem, such that

$$\textbf{minimize} \quad x^{\mathrm{T}} A x \tag{14a}$$

$$\textbf{subject to} \quad x^{\mathrm{T}} A x - 2b^{\mathrm{T}} x + c = 0. \tag{14b}$$

Proof: The optimal problem (11) is split on the sign of $x^{*\mathrm{T}} A x^* - 2b^{\mathrm{T}} x^* + c$.

Case $x^{*\mathrm{T}} A x^* - 2b^{\mathrm{T}} x^* + c < 0$:

The optimal problem of this case is written as

$$\textbf{minimize} \quad (k_x - k_y) x^{\mathrm{T}} A x + 2k_y b^{\mathrm{T}} x - k_y c.$$

If $k_x - k_y \leq 0$, there is no optimal point. Otherwise $k_x - k_y > 0$, the optimal point is written as

$$x^* = -\frac{k_y}{k_x - k_y} A^{-1} b.$$

The optimal point does not satisfies the condition $x^{*\mathrm{T}} A x^* - 2b^{\mathrm{T}} x^* + c < 0$, because

$$\begin{aligned} & x^{*\mathrm{T}} A x^* - 2b^{\mathrm{T}} x^* + c \\ &= \frac{k_y^2}{(k_x - k_y)^2} b^{\mathrm{T}} A^{-1} b + \frac{2k_y}{k_x - k_y} b^{\mathrm{T}} A^{-1} b + c > 0, \end{aligned}$$

where A^{-1} is a positive definite matrix and $c > 0$.

Case $x^{*\mathrm{T}} A x^* - 2b^{\mathrm{T}} x^* + c > 0$:

The problem (11) is written as

$$\textbf{minimize} \quad (k_x + k_y) x^{\mathrm{T}} A x - 2k_y b^{\mathrm{T}} x + k_y c.$$

Hence, this optimal point is written as

$$\begin{aligned} x^* &= \frac{k_y}{k_x + k_y} A^{-1} b \\ &= (1 - \tilde{k}_x) A^{-1} b, \end{aligned}$$

where $\sqrt{A}$ is a square root of the positive definite matrix A. The condition $x^{*\mathrm{T}}Ax^* - 2b^{\mathrm{T}}x^* + c > 0$ is rewritten by the previous optimal solution, such that

$$\begin{aligned}
&x^{*\mathrm{T}}Ax^* - 2b^{\mathrm{T}}x^* + c \\
&= (1-\tilde{k}_x)^2 b^{\mathrm{T}}A^{-1}b - 2(1-\tilde{k}_x)b^{\mathrm{T}}A^{-1}b + c \\
&= -\left(1-\tilde{k}_x^2\right) b^{\mathrm{T}}A^{-1}b + c > 0 \\
&\Leftrightarrow \tilde{c} > 1-\tilde{k}_x^2.
\end{aligned}$$

Case $x^{\mathrm{T}}Ax^* - 2b^{\mathrm{T}}x^* + c = 0$: The problem (11) is written as the QCQP problem (14). □

The solution of the simple QCQP problem (14) is easily derived using the method of Lagrange multiplier.

Lemma 4. *The solution of the simple QCQP problem (14) is given by*

$$x^* = \left(1-\sqrt{1-\tilde{c}}\right) A^{-1}b.$$

Proof: Supposing a Lagrange multiplier $\lambda > 0$, the Lagrange function is written as

$$L(x,\lambda) = x^{\mathrm{T}}Ax + \lambda(x^{\mathrm{T}}Ax - 2b^{\mathrm{T}}x + c).$$

The KKT condition is given by

$$\frac{\partial L(x^*,\lambda)}{\partial x} = 2(1+\lambda)Ax^* - 2\lambda b = 0. \tag{15}$$

$$\frac{\partial L(x^*,\lambda)}{\partial \lambda} = x^{*\mathrm{T}}Ax^* - 2b^{\mathrm{T}}x^* + c = 0, \tag{16}$$

Eq. (15) is written as

$$x^* = \frac{\lambda}{1+\lambda}A^{-1}b,$$

and the Eq. (16) is given by

$$\begin{aligned}
&x^{*\mathrm{T}}Ax^* - 2b^{\mathrm{T}}x^* + c = 0 \\
\Leftrightarrow &\frac{\lambda^2}{(1+\lambda)^2}b^{\mathrm{T}}A^{-1}b - \frac{2\lambda}{1+\lambda}b^{\mathrm{T}}A^{-1}b + c = 0 \\
\Leftrightarrow &-\frac{\lambda^2+2\lambda}{(1+\lambda)^2}b^{\mathrm{T}}A^{-1}b + c = 0 \\
\Leftrightarrow &-(\lambda^2+2\lambda)b^{\mathrm{T}}A^{-1}b + (1+\lambda)^2 c = 0 \\
\Leftrightarrow &(c - b^{\mathrm{T}}A^{-1}b)\lambda^2 + 2(c - b^{\mathrm{T}}A^{-1}b)\lambda + c = 0 \\
\Leftrightarrow &\lambda^2 + 2\lambda + \frac{\tilde{c}}{(\tilde{c}-1)} = 0 \\
\Leftrightarrow &\lambda = -1 \pm \sqrt{1 - \frac{\tilde{c}}{\tilde{c}-1}} \\
\Leftrightarrow &\lambda = -1 \pm \sqrt{\frac{1}{1-\tilde{c}}}.
\end{aligned}$$

As $\lambda > 0$ and $\tilde{c} > 0$, the Lagrange multiplier is written as

$$\lambda = -1 + \sqrt{\frac{1}{1-\tilde{c}}}.$$

Therefore, the optimal point x^* is given by

$$\begin{aligned}
x^* &= \frac{\lambda}{1+\lambda}A^{-1}b \\
&= \frac{-1+\sqrt{\frac{1}{1-\tilde{c}}}}{\sqrt{\frac{1}{1-\tilde{c}}}}A^{-1}b \\
&= (1-\sqrt{1-\tilde{c}})A^{-1}b.
\end{aligned}$$

□

Finally, we summarize three Lemmas 2-4 and solve the QCQP problem (10).

Lemma 1. *The solution of the problem (10) is given by*

$$x^* = \left(1 - \sqrt{\mathrm{C}\left(1-\tilde{c}; \tilde{k}_x^2, 1\right)}\right) A^{-1}b, \tag{17a}$$

$$y^* = \mathrm{R}\left(c - (1-\tilde{k}_x^2) b^{\mathrm{T}} A^{-1} b\right). \tag{17b}$$

Proof: Lemmas 2-4 split the solution x^* as three cases:

$$\begin{aligned} x^* &:= \begin{cases} (1-\tilde{k}_x)A^{-1}b & 1-\tilde{k}_x^2 < \tilde{c} \\ \left(1-\sqrt{1-\tilde{c}}\right)A^{-1}b & 0 < \tilde{c} \leq 1-\tilde{k}_x^2 \\ 0 & c \leq 0 \end{cases} \\ &:= \begin{cases} \left(1-\sqrt{\tilde{k}_x^2}\right)A^{-1}b & 1-\tilde{c} < \tilde{k}_x^2 \\ \left(1-\sqrt{1-\tilde{c}}\right)A^{-1}b & \tilde{k}_x^2 \leq 1-\tilde{c} < 1 \\ \left(1-\sqrt{1}\right)A^{-1}b & 1 \leq 1-\tilde{c} \end{cases} \\ &= \left(1 - \sqrt{\mathrm{C}\left(1-\tilde{c}; \tilde{k}_x^2, 1\right)}\right) A^{-1}b. \end{aligned}$$

Furthermore, the solution of y^* is written as,

$$\begin{aligned} y^* &:= \begin{cases} x^{*\mathrm{T}}Ax^* - 2b^{\mathrm{T}}x^* + c & 1-\tilde{k}_x^2 < \tilde{c} \\ x^{*\mathrm{T}}Ax^* - 2b^{\mathrm{T}}x^* + c & 0 < \tilde{c} \leq 1-\tilde{k}_x^2 \\ 0 & c \leq 0 \end{cases} \\ &= \begin{cases} c - (1-\tilde{k}_x^2)b^{\mathrm{T}}A^{-1}b & 1-\tilde{k}_x^2 < \tilde{c} \\ 0 & 0 < \tilde{c} \leq 1-\tilde{k}_x^2 \\ 0 & \tilde{c} \leq 0 \end{cases} \\ &= \mathrm{R}(c - (1-\tilde{k}_x^2)b^{\mathrm{T}}A^{-1}b). \end{aligned}$$

Therefore, the solution become Eq. (17). □

C Overall schematic of the learning process

Algorithm 1 shows the overall schematic of the learning process. The first line defines the modified dynamics $(f_{\mathbf{m}}, G_{\mathbf{m}}, h_{\mathbf{m}})$ from the nominal dynamics $(f_{\mathbf{n}}, G_{\mathbf{n}}, h_{\mathbf{n}})$, defined by the neural network, where ϕ is a set of parameters of the nominal dynamics. The 2-7 line represents a training loop, where the gradient-based optimization methods can be used by using the forward and backward calculation. Note that an ODE solver is used for forward calculation, and Algorithm 2 shows the forward calculation when the Euler method is used. For simplicity, mini-batch computation omitted in this schematic.

D Neural network architecture and hyper parameters

This section details how to determine the neural network architecture. The architecture and hyper parameters of the neural networks were basically determined by Bayesian optimizing using the validation dataset.

Table 1 shows the search space of Bayesian optimization. The first three parameters: learning rate, weight decay, and batch size are parameters for training the neural networks. Also, an optimizer is selected from AdamW, Adam, and RMSProp. The structure of neural network is determined from

Algorithm 1 Training process

Input: x_0: initial state, u: input signal,y: output signal, $(f_\mathbf{n}, G_\mathbf{n}, h_\mathbf{n})$: nominal dynamics, V: a designed function

1: define modified functions $(f_\mathbf{m}, G_\mathbf{m}, h_\mathbf{m})$ from $(f_\mathbf{n}, G_\mathbf{n}, h_\mathbf{n})$ and V
2: **for** 1 to #iterations **do**
3: $\hat{y} \leftarrow$ ODE with $(f_\mathbf{m}, G_\mathbf{m}, h_\mathbf{m})$ from x_0, u (**Algorithm 2**)
4: forward computation of Loss function (9) from y
5: ∇_ϕLoss $\leftarrow$ backward computation with Loss
6: $\phi \leftarrow$ Optimizer(ϕ, ∇_ϕLoss)
7: **end for**

Algorithm 2 Forward computation for dynamics Eq. (1)

Input: x_0: initial state, u: input signal, $(f_\mathbf{m}, G_\mathbf{m}, h_\mathbf{m})$: dynamics
Output: $\hat{y}$: output signal

1: **for** t $\leftarrow 0$ to T **do**
2: $x_{t+1} \leftarrow x_t + \Delta t(f_\mathbf{m}(x_t) + G_\mathbf{m}(x_t)u_t)$
3: $\hat{y}_t \leftarrow h_\mathbf{m}(x_t)$
4: **end for**
5: **return** $\hat{y}$

the number of intermediate layers and dimensions for each layer. One layer in our setting consists of a fully connected layer with a ReLU activation. The last three rows represent parameters related to our proposed methods. ϵ is a parameter of the loss function L_{HJ}. Initial scale parameter is multiplied with the output of $f_\mathbf{n}$ to prevent the value of $f_\mathbf{n}(x)$ from becoming large in the initial stages of learning. When $f_\mathbf{n}(x)$, which determine the behavior of the internal system, outputs a large value, it diverges due to time evolution, and the learning of the entire system may not progress. Therefore, it is empirically preferable to start with a small value for $f_\mathbf{n}(x)$ at the initial stage of learning. When the flag of 'stop gradient for projection' is false, backward computation related to the second term of modification of $f_\mathbf{m}$ and $G_\mathbf{m}$ is disabled. Note that modification related to $f_\mathbf{m}$ and $G_\mathbf{m}$ consists of two terms (see Theorem 1). Setting this parameter to false resulted in better performance in our all experiments.

We ran 300 trials using the Bayesian optimization for the bistable model benchmark and the glucose insulin benchmark with the above settings, and the hyper parameters obtained are shown in Table 2, where the number of dimensions for each hidden layer is shown in tuple from the order closest to the input layer.

Hyperparameters of comparative methods were determined by grid search using the validation dataset. ARX and PWARX have an order parameter n of the autoregressive model, and this parameter is searched in the range of $1 - 5$. The number of iterations was set to 10000 so that the optimization of PWARX converges sufficiently. MOESP and ORT have an internal dimension n ($1 \leq n \leq 20$) and the number of subsequences used for estimation k ($2n < k \leq 20$).

Table 1: The search space of Bayesian optimization

parameter name	range	type
learning rate	10^{-5} – 10^{-3}	log scale
weight decay	10^{-10} – 10^{-6}	log scale
batch size	10 – 100	integer
optimizer	{ AdamW, Adam, RMSProp}	categorical
#layer for $f_{\mathbf{n}}$	0 – 3	integer
#layer for $G_{\mathbf{n}}$	0 – 3	integer
#layer for $h_{\mathbf{n}}$	0 – 3	integer
#dim. for a hidden layer of $f_{\mathbf{n}}$	$8 - 32$	integer
#dim. for a hidden layer of $G_{\mathbf{n}}$	$8 - 32$	integer
#dim. for a hidden layer of $h_{\mathbf{n}}$	$8 - 32$	integer
ϵ	0 – 1.0	log scale
Initial scale parameter for $f_{\mathbf{n}}$	10^{-5} – 0.1	log scale
Stop gradient for projection	true, false	boolean

Table 2: Selected parameters for each benchmark

parameter name	bistable	glucose insulin
learning rate	3.01×10^{-4}	3.28×10^{-4}
weight decay	4.76×10^{-9}	2.28×10^{-9}
batch size	100	100
optimizer	RMSProp	RMSProp
#layer for $f_{\mathbf{n}}$	3	1
#layer for $G_{\mathbf{n}}$	1	2
#layer for $h_{\mathbf{n}}$	3	2
#dim. for a hidden layer of $f_{\mathbf{n}}$	(17,10,22)	(8)
#dim. for a hidden layer of $G_{\mathbf{n}}$	(34)	(27,29)
#dim. for a hidden layer of $h_{\mathbf{n}}$	(10,62,58)	(35,18)
ϵ	0.63	0.75
Initial scale parameter for $f_{\mathbf{n}}$	9.64×10^{-2}	8.94×10^{-2}
Stop gradient for projection	false	false

E Glucose-Insulin system

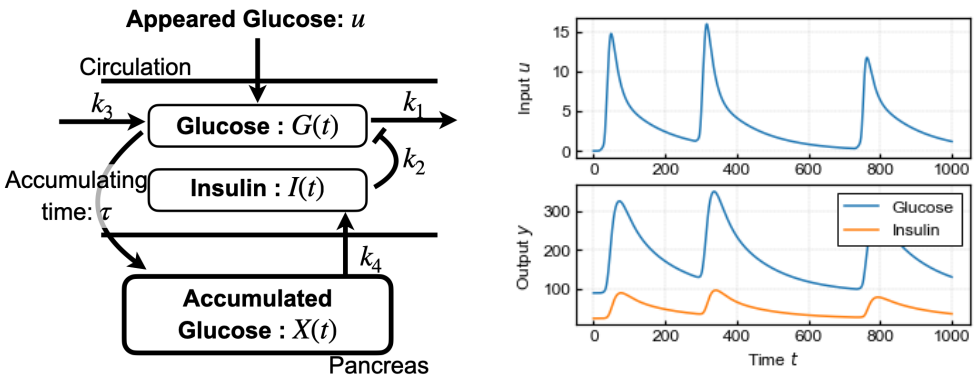
Glucose concentration in the blood is modeled as a time-delay system regulated by insulin concentration (See Fig. 7 (A)) [32]. Suppose that G I, and X are the glucose, insulin, and accumulated glucose plasma concentration ([mg/100ml],[μUI/ml],and [min mg/100ml], respectively) and u is the amount of ingested glucose per minute [min^{-1} mg/100ml]. The dynamics of each concentration is given by

$$
\begin{aligned}
\dot{G}(t) &= -k_1 G(t) - k_2 G(t) I(t) + g_0 + u(t), \\
\dot{I}(t) &= -k_3 I(t) + \frac{k_4}{\tau} X(t), \\
\dot{X}(t) &= G(t) - G(t-\tau), \\
y(t) &= [G(t), I(t)]^{\mathrm{T}},
\end{aligned}
$$

where k_1 is a spontaneous glucose disappearance rate, k_2 is an insulin-dependent glucose disappearance rate, g_0 is a constant increase in plasma glucose concentration, k_3 is an insulin disappearance rate, k_4 is an insulin release rate per the average glucose concentration within the last τ minute.

Table 3: Model parameters.

Parameter	Value	Unit
k_1	3.35×10^{-2}	$\frac{1}{\text{min}}$
k_2	5.22×10^{-5}	$\frac{1}{\text{min}(\mu\text{UI/ml})}$
k_3	1.055	$\frac{1}{\text{min}}$
k_4	0.293	$\frac{(\mu\text{UI/ml})}{\text{min}(\text{mg/100ml})}$
g_0	3.13	$\frac{(\text{mg/100ml})}{\text{min}}$
τ	6	min



(A) (B)

Figure 7: (A) Overview and (B) input and output behavior of the glucose insulin system.

This system has a unique asymptotically stable equilibrium point $(G, I, X) \equiv (G^*, I^*, X^*)$ on the nonlinear plain such that

$$G^* = \frac{-k_1k_3 + \sqrt{(k_1k_3)^2 + 4k_2k_3k_4g_0}}{2k_2k_4},$$
$$I^* = \frac{-k_1k_3 + \sqrt{(k_1k_3)^2 + 4k_2k_3k_4g_0}}{2k_2k_3},$$
$$X^* = \frac{-k_1k_3 + \sqrt{(k_1k_3)^2 + 4k_2k_3k_4g_0}}{2k_2k_4}\tau.$$

Here we set the initial state of this model as $G(0) = G^*, I(0) = I^*, X(0) = X^*$ and set the model parameters as shown in the following Table 3. Furthermore, we adopt the output of the previous oral glucose absorption system [33] as u (See Fig. 7 (B)), and the glucose absorption amount u is normalized based on human blood volume per body weight (0.80[100 ml/kg]).