Neural Event-Triggered Control with Optimal Scheduling

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Learning-enabled controllers with stability certificate functions have demonstrated impressive empirical performance in addressing control problems in recent years. Nevertheless, directly deploying the neural controllers onto actual digital platforms requires impractically high communication resources due to a continuously updating demand from the closed-loop feedback controller. We propose a framework aimed at learning the event-triggered controller (ETC) with optimal scheduling, i.e., minimal triggering times, to address this challenge in resource-constrained scenarios. Our framework Neural ETC consists of two practical algorithms: the path integral algorithm, which is based on the neural event ODE method, and the Monte Carlo algorithm based on our new theoretical results regarding lower bound of inter-event time. In comparison to the conventional neural controllers, our empirically finding indicates that the Neural ETC significantly decreases the required communication resources and improve the control performance in limited communication resources scenarios.

1 Introduction

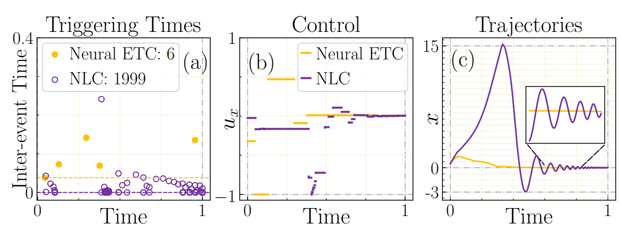

Figure 1: Comparison of Neural ETC (yellow) and neural Lyapunov control (NLC, purple) in stabilizing the Lorenz system under the event-triggered control setting. (a): The inter-event time of consecutive triggering events. The dashed lines represent the minimal inter-event time of each control. (b): The control values acting on variable x in the control process. (c): The controlled trajectories of the variable x.

Stabilizing the complex nonlinear systems represents a formidable focal task within the realms of mathematics and engineering. Previous research in the field of cybernetics has applied the Lyapunov stability theory to formulate stabilizing policies for linear or polynomial dynamical systems, including the linear quadratic regulator (LQR) [1] and the sum-of-squares (SOS) polynomials, using the semi-definite planning (SDP) [2]. Stabilizing more intricate dynamical systems with high dimension and nonlinearity, as encountered in real applications, has prompted the integration of machine learning techniques into the cybernetics community[3]. Recent advancements in learning neural networks based controllers with certificate functions, such as Lyapunov function [4, 5], LaSalle function [6], barrier functions [7] and contraction metrics [8], have demonstrated outstanding performance in controlling diverse dynamics [9]. Nevertheless, it is noteworthy that all these controllers require updating the control signal continuously over time, leading to a considerable communication cost between controller and platform.

The periodic control is mostly advocated for implementing feedback control laws on digital platforms [10]. However, such implementations often incur significant over-provisioning of the communication network, especially in the recently developed large-scale resource-constrained wireless embedded control systems [11]. To mitigate this issue, the event-triggering mechanism is introduced to generate sporadic transmissions across the feedback channels of the system. Compared to the periodic control which updates the control signal at a series of predefined explicit times, event-triggered control updates the control signal at the instants when the current measurement violates a predefined triggering condition, thereby triggering a state-dependent event [12]. Given that these instants are implicitly determined by the state trajectories, the scheduling of computation and communication resources for event-triggered control becomes a very challenging problem, involving the minimization of the number of events and the increase of inter-event time. While significant strides have been made in designing the stabilizing event-triggered control for specific dynamics in recent years, the task of designing event-triggered control for general nonlinear and large-scale dynamics with optimal scheduling remains an open problem [13, 14, 15, 16].

Our goal is to design event-triggered control for general complex dynamics, ensuring both stability guarantee and optimal scheduling, i.e., to implement event-triggered control with the minimal triggering times and the maximal inter-event time. Fig. 1 depicts the comparison of control performance of the Neural ETC and the NLC in the event-triggered realization to stabilize a Lorenz dynamic. In Fig. 1(a)-1(b), it is evident that the triggering times of Neural ETC are significantly fewer than those of NLC, and the minimal inter-event time of consecutive triggering times of Neural ETC considerably exceeds that of NLC. These disparities lead to the different behaviors of the controlled trajectories, as depicted in Fig. 1(c).Under Neural ETC, the trajectory rapidly converges to the target state, while the NLC exhibits violent oscillation around the target.

Contribution. The principal contributions of this paper can be summarized as follows:

- We propose Neural ETC, a framework for learning event-triggered controllers ensuring both stability guarantee and optimal scheduling. We provide a general form of event function to provide exponential stability guarantee for the event-triggered controlled system. We firstly propose a path integral approach to realize the implementation of the machine learning framework based on the root solver and neural event ODE solver that calculate the trainable event triggering times.
- Secondly, we theoretically address the estimation of the minimal inter-event time of the event triggered controlled system, which leading to the Monte Carlo approach of our framework that circumvents the expensive computation cost of back-propagation through ODE solvers. The two approaches trade off in terms of stabilization performance and training efficiency, which is convenient for users to flexibly choose the specific approach according to the task in hand.
- Finally, we evaluate Neural ETCs on a variety of representative physical and engineering systems. Compared to existing stabilizing controllers, we find that Neural ETCs exhibit significant superiority in decreasing the triggering times and maximizing the minimal inter-event time. The code for reproducing all the numerical experiments is released at `anonymous/Neural-Event-triggered-Control`.

2 Background

Notations. Denote by $\|\cdot\|$ the L^2-norm for any given vector in $\mathbb{R}^d$. Denote by $\|\cdot\|_{C(\mathcal{D})}$ the maximum norm on continuous function space $C(\mathcal{D})$. For $A=(a_{ij})$, a matrix of dimension $d\times r$, denote by $\|A\|_{\mathrm{F}}^2=\sum_{i=1}^d\sum_{j=1}^r a_{ij}^2$ the Frobenius norm. Denote $\max(a,0)$ by $(a)^+$. Denote $\boldsymbol{x}\cdot\boldsymbol{y}$ as the inner product of two vectors.

2.1 Neural Lyapunov Control

To begin with, we consider the feedback controlled dynamical system of the following general form:

$$\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}(\boldsymbol{x}))\triangleq\boldsymbol{f}_{\boldsymbol{u}}(\boldsymbol{x}),\ \boldsymbol{x}\in\mathbb{R}^d,\ \boldsymbol{u}\in\mathbb{R}^m, \tag{1}$$

where $\boldsymbol{f_u}(\boldsymbol{x}) : \mathcal{D} \to \mathbb{R}^d$ is the Lipschitz-continuous vector field acting on some prescribed open set $\mathcal{D} \subset \mathbb{R}^d$. The solution initiated at time t_0 from $\boldsymbol{x}_0$ under controller $\boldsymbol{u}$ is denoted by $\boldsymbol{x_u}(t; t_0, \boldsymbol{x}_0)$. For brevity, we let the unstable equilibrium $\boldsymbol{x}^* \in \mathcal{D}$ be origin, i.e., $\boldsymbol{f}(\boldsymbol{0}, \boldsymbol{0}) = \boldsymbol{0}$. One major problem in cybernetics field is to design stabilizing controller $\boldsymbol{u}(\boldsymbol{x})$ [17] such that $\lim_{t\to\infty} \boldsymbol{x_u}(t; t_0, \boldsymbol{x}_0) = \boldsymbol{0}$, for any initial value $\boldsymbol{x}_0 \in \mathcal{D}$.

Theorem 2.1 *[18] Suppose there exists a continuously differentiable function $V : \mathcal{D} \to R$ that satisfies the following conditions:* (i) $V(0) = 0$, (ii) $V(\boldsymbol{x}) \geq c\|\boldsymbol{x}\|^p$ *for some constants* $c, p > 0$, (iii) *and* $\mathcal{L}_{\boldsymbol{f_u}} V < -\delta V$, *for some* $\delta > 0$. [1] *Then, the system is exponentially stable at the origin, that is,* $\limsup_{t\to\infty} \frac{1}{t} \log \|\boldsymbol{x_u}(t; t_0, \boldsymbol{x}_0)\| \leq -\frac{\delta}{p}$. *Here V is called a Lyapunov function.*

Previous works parameterize the controller and the Lyapunov function as $\boldsymbol{u_\phi}$, $V_{\boldsymbol{\theta}}$, and integrate the sufficient conditions for Lyapunov stability into the loss function as $L(\boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{N}\sum_{i=1}^{N}\Big((c\|\boldsymbol{x}_i\|^p - V_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^+ + (\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u_\phi}}} V_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \delta V_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^+\Big) + V_{\boldsymbol{\theta}}(\boldsymbol{0})^2$ [4, 5, 19]. The learned Lyapunov V plays a role of stability certificate function.

Remark 2.2 *Unlike model-free reinforcement learning (RL) approaches that search for an online control policy guided by a reward function along the trajectories of the dynamical systems. [20], the neural Lyapunov control searches for an offline policy and a certificate function V that proves the soundness of the learned policy [9]. Nevertheless, updating the feedback policy continuously in the implementation process incurs prohibitive high communication cost.*

2.2 Event-triggered Control

Although the feedback controller works well in numerical simulations, updating and implementing the controller continuously is impractical in most real-world digital platforms under communication constraints [21]. To conquer this weakness, event-triggered stabilizing control is introduced as follows [12],

Definition 2.1 *(Event-triggered Control) Consider the controlled system* (1)*, the event-triggered controller is defined as* $\mathbf{u}(t) = \boldsymbol{u}(\boldsymbol{x}(t_k))$, $t_k \leq t < t_{k+1}$, *where the triggering time is decided by* $t_{k+1} = \inf\{t > t_k : h(\boldsymbol{x}(t)) = 0\}$ *for some predefined event function h. The largest lower bound τ^* of $\{t_{k+1} - t_k\}$ is called as minimal inter-event time. For example, if there exists a Lyapunov function V for the feedback controlled system* (1)*, then the event function is set to guarantee the Lyapunov condition on each event triggering time interval, i.e.,* $\nabla V \cdot \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(\boldsymbol{x}_{t_k})) < 0,\ t \in [t_k, t_{k+1})$.

Problem Statement. We assume that the zero solution of the uncontrolled system in Eq. (1) is unstable, i.e. $\lim_{t\to\infty} \boldsymbol{x}_{\boldsymbol{u}=\boldsymbol{0}}(t; t_0, \boldsymbol{x}_0) \neq \boldsymbol{0}$. We aim at stabilizing the zero solution using event-triggered control based on neural networks (NNs) with optimal scheduling, i.e., the least triggering times, which is urgently required by the digital platforms wherein the communication resources of updating the control value are limited. Notice that in an average sense, the triggering times are inversely proportional to the inter-event time, our goal is equivalently to leverage the NNs to design an appropriate controller $\boldsymbol{u}$ with $\boldsymbol{u}(\boldsymbol{0}) = \boldsymbol{0}$ such that the controlled system under event-triggered implementation is steered to the zero solution with the maximal inter-event time. We summarize the problem formulation as the following optimization problem,

$$\begin{aligned} &\max_{\boldsymbol{u}} \left(\min_{\{t_k \leq T\}} (t_{k+1} - t_k)\right) + \lambda_1 \|\boldsymbol{u}(\boldsymbol{x})\|_{C(\mathcal{D})} \\ \text{s.t.}\quad &\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(\boldsymbol{x}(t_k))),\ t \in [t_k, t_{k+1}), \\ &\boldsymbol{x}(0) = \boldsymbol{x}_0 \in \mathcal{D},\ \lim_{t\to T} \boldsymbol{x}(t) = \boldsymbol{0}, \end{aligned} \tag{2}$$

here the triggering time $\{t_k : t_k \leq T\}$ depends on the controller $\boldsymbol{u}$ and the triggering mechanism, and $T \leq \infty$ is the prefixed time limit according to the specific tasks. We aim at devising controller $\boldsymbol{u}$ and triggering mechanism to solve this problem based on the known model $\boldsymbol{f}$ and time limit T.

The major difficulty of this problem comes from that the implicitly defined triggering times are not equidistant, and are only known when the events are triggered [22]. The majority of existing works focus on the stabilization performance of event-triggered control and often omit the communication

[1] $\mathcal{L}_{\boldsymbol{f_u}} V$ represent the Lie derivative of V along the direction $\boldsymbol{f_u}$, i.e., $\mathcal{L}_{\boldsymbol{f_u}} V = \nabla V \cdot \boldsymbol{f_u}$.

cost of updating the control value at triggering moments. In what follows, we propose neural event-triggered control (Neural ETC) framework to address both the stabilization and the communication cost issues of event-triggered control.

3 Method

Closed-loop controlled dynamics. The dynamics under event-triggered control is generally an open-loop system with controller varying from different triggering time intervals. In order to simplify the theoretical analysis and to utilize the existing numerical tools for ODE solvers, we transform the event-triggered controlled system to the closed-loop version via augmenting the dynamics with an error state $\boldsymbol{e}(t) = \boldsymbol{x}(t_k) - \boldsymbol{x}(t)$, $t \in [t_k, t_{k+1})$ and an update operation $\boldsymbol{e}(t_{k+1}) = \boldsymbol{0}$. Then we obtain the closed-loop controlled dynamics as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}+\boldsymbol{e})), \dot{\boldsymbol{e}} = -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}+\boldsymbol{e})),\ t \in [t_k, t_{k+1}).$$

In the next sections, we construct the event function with exponential stability guarantee and deduce the theoretical estimation of minimal inter-event time based on the augmented dynamics of $(\boldsymbol{x}, \boldsymbol{e})$.

Event function for exponential stability. We consider the exponential Lyapunov stability for controlled system (1) such that the corresponding Lyapunov function defined in Theorem 2.1 satisfies the stability condition $\mathcal{L}_{\boldsymbol{f_u}} V \leq -\delta V$ and $V(\boldsymbol{x}) \geq \alpha(\|\boldsymbol{x}\|)$, where α is a class-K function[2]. For brevity, we fix $\delta = 1$ in this paper such that the decay exponent of the Lyapunov function is 1. Since the event-triggered controller is a discrete time realization of the original feedback controller $\boldsymbol{u}$, the corresponding exponential decay rate of the Lyapunov function is less than 1. Therefore, we design the event function $h = h(\boldsymbol{x}, \boldsymbol{e})$ as

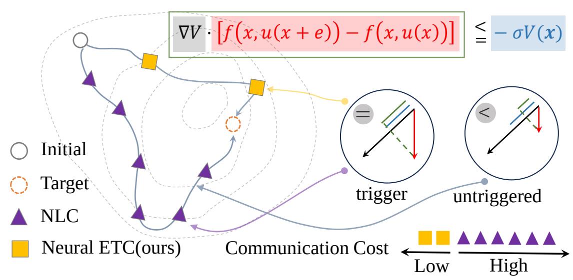

Figure 2: Illustration of the Neural ETC with optimal scheduling.

$$h = \nabla V(\boldsymbol{x}) \cdot (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}+\boldsymbol{e})) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))) - \sigma V(\boldsymbol{x}) \tag{3}$$

with $0 < \sigma < 1$, such that the event-triggered controlled system satisfies

$$\nabla V(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}+\boldsymbol{e})) \leq \nabla V(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x})) + \sigma V(\boldsymbol{x}) \leq -(1-\sigma) V(\boldsymbol{x}). \tag{4}$$

Hence, the exponential stability of the event-triggered controlled system is assured with exponential decay rate $1 - \sigma$. As illustrated in Fig. 2, the NLC is used as an example method to be compared with the Neural ETC. The control value is updated when an event is triggered, i.e., the event function h equals to zero. Our method achieves the exponential stability and has the least triggered events, leading to the lowest communication cost of updating the control value.

3.1 Path Integral Approach

Parameterization. In order to design the feedback controller such that its event-triggered implementation stabilize the unstable equilibrium efficiently and has the largest minimal inter-event time, we consider the following parameterized optimization problem.

$$\begin{aligned}
&\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \left(\min_{\{t_k \leq T\}} \frac{1}{t_{k+1} - t_k} \right) + \lambda_1 \|\boldsymbol{u}_{\boldsymbol{\phi}}(\boldsymbol{x})\|_{C(\mathcal{D})} \\
\text{s.t.}\quad &\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_{\boldsymbol{\phi}}(\boldsymbol{x}+\boldsymbol{e})),\ \dot{\boldsymbol{e}} = -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_{\boldsymbol{\phi}}(\boldsymbol{x}+\boldsymbol{e})),\ t \in [t_k, t_{k+1}), \\
&\boldsymbol{x}(0) = \boldsymbol{x}_0,\ \boldsymbol{e}(t_k) = \boldsymbol{0},\ V_{\boldsymbol{\theta}}(\boldsymbol{0}) = 0,\ \boldsymbol{u}_{\boldsymbol{\phi}}(\boldsymbol{0}) = \boldsymbol{0}, \\
&t_{k+1} = \inf_{t > t_k} \{t : h_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}(t), \boldsymbol{e}(t)) = 0\} \\
&\alpha(\|\boldsymbol{x}\|) - V_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 0,\ \mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_{\boldsymbol{\phi}}}} V_{\boldsymbol{\theta}}(\boldsymbol{x}) + V_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 0.
\end{aligned}$$

[2]A continuous function $\alpha : (0, \infty) \to (0, \infty)$ is said to belong to class-K if it is strictly increasing and $\alpha(0) = 0$.

Here, λ_1 is a predefined weight factor, T is the temporal length of the controlled trajectory, α is a class-K function, and $h_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{e},\boldsymbol{x}) = \mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_{\boldsymbol{\phi}}}} V_{\boldsymbol{\theta}} \cdot (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_{\boldsymbol{\phi}}(\boldsymbol{x}+\boldsymbol{e})) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_{\boldsymbol{\phi}}(\boldsymbol{x}))) - \sigma V_{\boldsymbol{\theta}}(\boldsymbol{x}(t))$ is the parameterized event function. To ensure the neural functions $V_{\boldsymbol{\theta}}$, $\boldsymbol{u}_{\boldsymbol{\phi}}$ satisfy some constraints naturally, we adopt the parametrization in [5] as follows,

$$\begin{aligned} V_{\boldsymbol{\theta}} &= \mathrm{ICNN}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \mathrm{ICNN}_{\boldsymbol{\theta}}(\boldsymbol{0}) + \varepsilon \|\boldsymbol{x}\|^2, \\ \boldsymbol{u}_{\boldsymbol{\phi}} &= \mathrm{diag}(\boldsymbol{x})\mathrm{NN}_{\boldsymbol{\phi}}(\boldsymbol{x}) \text{ or } \mathrm{NN}_{\boldsymbol{\phi}}(\boldsymbol{x}) - \mathrm{NN}_{\boldsymbol{\phi}}(\boldsymbol{0}), \end{aligned} \tag{5}$$

where $\mathrm{diag}(\boldsymbol{x})$ transforms a vector to a diagonal matrix with $(\mathrm{diag}(\boldsymbol{x}))_{ij} = \delta_{ij} x_i$, $\mathrm{ICNN}_{\boldsymbol{\theta}}$ and $\mathrm{NN}_{\boldsymbol{\phi}}$ represent the input convex neural network and the feedforward neural networks, respectively, the detailed formulation is provided in Appendix A.3.1. We minimize the continuous function norm $\|\boldsymbol{u}_{\boldsymbol{\phi}}\|_{C(\mathcal{D})}$ by regularizing the Lipschitz constant of the neural network, we apply the spectral norm regularization method in [23] to minimize the spectral norm of the weight matrices $\{\boldsymbol{W}_{\boldsymbol{\phi},i}\}_{i=1}^{l}$ in $\boldsymbol{u}_{\boldsymbol{\phi}}$ with the regularization term $L_{\text{lip}} = \sum_{i=1}^{l} \sigma(\boldsymbol{W}_{\boldsymbol{\phi},i})^2$. To solve the substantially non-convex optimization problem, we relax the original hard constraint $\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_{\boldsymbol{\phi}}}} V_{\boldsymbol{\theta}}(\boldsymbol{x}) + V_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 0$ to a soft constraint in the loss function as $L_{\text{stab}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_{\boldsymbol{\phi}}}} V_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + V_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)^+$.

Calculate gradients of t_k. To proceed, we handle the objective function related to the triggering times. Instead of directly training the parameters $\boldsymbol{\phi}, \boldsymbol{\theta}$ based on the direct samples of $V_{\boldsymbol{\theta}}$, $\boldsymbol{u}_{\boldsymbol{\phi}}$ and $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_{\boldsymbol{\phi}}(\boldsymbol{x}))$ as done in neural certificate-based controllers, we have to numerically solve the controlled ODEs to identify the triggering times. To proceed, we need to calculate the gradients of t_k for optimizing $\frac{1}{t_{k+1}-t_k}$ term in loss function during gradient-based optimization. We employ the neural event ODE method as: $t_{k+1}, \boldsymbol{x}(t_{k+1}) = \texttt{ODESolveEvent}(\boldsymbol{x}(t_k), \boldsymbol{f}, \boldsymbol{u}_{\boldsymbol{\phi}}, t_k)$, where `ODESolveEvent` is proposed by [24], which introduces root solver and adjoint method [25] to the numerical solver and deduce the gradient $\frac{\partial t_k}{\partial \boldsymbol{\phi}}$ from the implicit function theorem [26].

Reduce computation complexity. We denote by $t_k(\boldsymbol{x})$ the k_{th} triggering time from initial value $t_0 = 0, \boldsymbol{x}(0) = \boldsymbol{x}$. The computation cost of `ODESolveEvent` is $\mathcal{O}(M\bar{K}Ld^2)$, where M is the batch size of the initial value $\{\boldsymbol{x}_i(0)\}_{i=1}^{M}$, $\bar{K} = \frac{1}{M}\sum_{i=1}^{M} K(\boldsymbol{x}_i(0))$, $K(\boldsymbol{x}_i(0)) = \#\{t_k(\boldsymbol{x}_i(0)) : t_k \leq T\}$ is the number of triggering times before T, and L is the iteration times in the root solver. In this case, the computation cost pivots on the sampled batch and its variance is hard to decrease. In addition, the numerical error in ODE solver accumulates over the triggering time sequence $\{t_k\}$. To mitigate these issues, according to the time invariance property of ODEs, i.e., $t_{k+1}(\boldsymbol{x}(0)) - t_k(\boldsymbol{x}(0)) = t_1(\boldsymbol{x}(t_k))$, we recast the problem of solving M batch trajectories $\{\boldsymbol{x}_i(t_k),\ t_k \leq T | \boldsymbol{x}_i(0) \sim q_0(\boldsymbol{x})\}_{i=1}^{M}$ of controlled ODE as solving MK trajectories $\{\boldsymbol{x}_i(t_1),\ t_1 \leq T | \boldsymbol{x}_i(0) \sim \tilde{q}_0(\boldsymbol{x})\}_{k=1}^{MK}$ up to t_1. Here, K represent the expectation of $\bar{K}$. In practice, we directly treat MK together as a single hyperparameter M. Then the triggering times contribute into the loss function as $L_{\text{event}} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{t_1(\boldsymbol{x}_i(0))}$. Finally, we train the overall parameterized model with the total loss function as follows,

$$L(\boldsymbol{\phi}, \boldsymbol{\theta}) = L_{\text{stab}} + \lambda_1 L_{\text{lip}} + \lambda_2 L_{\text{event}} \tag{6}$$

The whole training procedure is summarized in Algorithm 1.

Remark 3.1 *A more reasonable augmented distribution should takes the form as* $\tilde{q}_0(\boldsymbol{x}) = \frac{1}{K-1}\sum_{k=0}^{K-1} q_k(\boldsymbol{x})$*, where* $\boldsymbol{x}_{t_k} \sim q_k(\boldsymbol{x})$ *is deduced from the initial distribution* q_0 *and the ODE integration from* 0 *to* t_k*. Since* t_k *varies for different initial value and cannot be determined in advance, we fix* $\tilde{q}_0 = q_0$ *for simplicity.*

3.2 Monte Carlo Approach

Although the proposed algorithm works efficiently in low dimensional ODEs, the high computation cost and accumulate error caused by the ODE solver affect its performance in higher dimensional tasks. To circumvent this drawback, we propose a Monte Carlo approach for training the Neural ETC. Inspired by the event-triggered scheduling theory in [14], we provide the following estimation of minimal inter-event time.

Theorem 3.2 *Consider the event-triggered controlled dynamics in Eq.* (1)*, if the following assumptions are satisfied:* (i) $\|\boldsymbol{f}(\boldsymbol{x}', \boldsymbol{u}') - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})\| \leq l_{\boldsymbol{f}}\left(\|\boldsymbol{x}' - \boldsymbol{x}\| + \|\boldsymbol{u}' - \boldsymbol{u}\|\right)$*;* (ii) $\|\boldsymbol{u}(\boldsymbol{x}') - \boldsymbol{u}(\boldsymbol{x})\| \leq$

$l_{\boldsymbol{u}}\|\boldsymbol{x}' - \boldsymbol{x}\|$; (iii) $\mathcal{L}_{\boldsymbol{f_u}} V(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}+\boldsymbol{e})) \leq -\alpha(\|\boldsymbol{x}\|) + \gamma(\|\boldsymbol{e}\|)$ *for some class-K functions* α, γ *with* $\alpha^{-1}(\gamma(\|\boldsymbol{e}\|)) \leq P\|\boldsymbol{e}\|$*. Then, the minimal inter-event time implicitly defined by event function* $h = \alpha(\|\boldsymbol{x}\|) - \gamma(\|\boldsymbol{e}\|)$ *is lower bounded by* $\tau_h = \frac{1}{l_{\boldsymbol{f}}} \log \frac{P+1}{P + \frac{l_{\boldsymbol{f}} l_{\boldsymbol{u}}}{l_{\boldsymbol{f}}(1+l_{\boldsymbol{u}})}}$.

The detailed proof is provided in Appendix A.1.2. According to the theorem, the lower bound of minimal inter-event time increases as Lipschitz constants of $\alpha^{-1} \circ \gamma$ and $\boldsymbol{u}$ decrease. Therefore, we can maximize the minimal inter-event time by regularizing these Lipschitz constants. Nonetheless, directly integrating the conditions and results of Theorem 3.2 into the training process is unrealistic and cumbersome, because the error state $\boldsymbol{e}$ in condition (iii) should depend on $\boldsymbol{x}$ and we cannot determine the sampling distribution of $\boldsymbol{e}$ before training. To solve this problem, we split the inequality in condition (iii) into a sufficient inequality group as

$$\nabla V \cdot (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}+\boldsymbol{e})) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))) \leq \gamma(\|\boldsymbol{e}\|) \tag{7}$$

$$\mathcal{L}_{\boldsymbol{f_u}} V(\boldsymbol{x}) \leq -\alpha(\|\boldsymbol{x}\|) \tag{8}$$

$$\rightarrow \mathcal{L}_{\boldsymbol{f_u}} V(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}+\boldsymbol{e})) \leq -\alpha(\|\boldsymbol{x}\|) + \gamma(\|\boldsymbol{e}\|)$$

The dependence on $\boldsymbol{x}$ of right term γ in Eq. (7) can be omitted when the state space $\mathcal{D}$ is bounded, which occurs in most real-world scenarios. The Eq. (7) implies that the Lipschitz constant of γ is related to the Lipschitz constant of $\boldsymbol{u}$. Furthermore, we notice that if we replace the event function in Theorem 3.2 by the following event function with stability guarantee,

$$\tilde{h} = \nabla V \cdot (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}+\boldsymbol{e})) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))) - \alpha(\|\boldsymbol{x}\|), \tag{9}$$

then the inter-event time of these two event functions hold the relation $t_{k+1,h} - t_{k,h} \leq t_{k+1,\tilde{h}} - t_{k,\tilde{h}}$ due to Eq. (7). Therefore, the inter-event time of event function $\tilde{h}$ is also lower bounded by τ_h in Theorem 3.2. We summarize the results in the following theorem.

Theorem 3.3 *For the event-triggered controlled dynamics in Eq.* (1) *with event function* $\tilde{h}$ *defined in Eq.* (9)*, if the state space* $\mathcal{D}$ *is bounded, the Eqs.* (7),(8) *and the conditions* (*i*), (*ii*) *in Theorem 3.2 hold, then the minimal inter-event time is lower bounded by* $\tau_{\tilde{h}} = \frac{1}{l_{\boldsymbol{f}}} \log \frac{c l_{\alpha^{-1}} l_{\boldsymbol{u}} + 1}{c l_{\alpha^{-1}} l_{\boldsymbol{u}} + \frac{l_{\boldsymbol{f}} l_{\boldsymbol{u}}}{l_{\boldsymbol{f}}(1+l_{\boldsymbol{u}})}}$*, here* $l_{\alpha^{-1}}$ *is the Lipschitz constant of* α^{-1}*,* c *is a constant depending on* $V, \boldsymbol{f}, \mathcal{D}$*.*

The proof is provided in Appendix A.1.3. With this theorem, we come to a Monte Carlo approach for training the Neural ETC framework by directly learning the parameterized functions $V_{\boldsymbol{\theta}}$, $\alpha_{\boldsymbol{\theta}_\alpha}$, and control function $\boldsymbol{u}_{\boldsymbol{\phi}}$ simultaneously, as well as regularizing the Lipschitz constants of $\boldsymbol{u}_{\boldsymbol{\phi}}$ and $\alpha^{-1}_{\boldsymbol{\theta}_\alpha}$. For constructing neural class-K functions, we adopt the monotonic NNs to construct the candidate class-$\mathcal{K}$ function as

$$\alpha_{\boldsymbol{\theta}_\alpha}(x) = \int_0^x q_{\boldsymbol{\theta}_\alpha}(s)\mathrm{d}s, \tag{10}$$

where $q_{\boldsymbol{\theta}_\alpha}(\cdot) \geq 0$ is the output of the NNs [27]. We regularize the inverse of integrand to minimize the Lipschitz constant of $\alpha_{\boldsymbol{\theta}_\alpha}$. We apply the spectral norm regularization L_{lip} defined above to minimize the Lipschitz constant of controller $\boldsymbol{u}_{\boldsymbol{\phi}}$. Finally, we train the overall model with the loss function as follows,

$$\begin{aligned} &\tilde{L}_{\text{stab}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_{\boldsymbol{\phi}}}} V_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_i)\right)^+, \ L_{\alpha^{-1}} = \frac{1}{M_\alpha}\sum_{i=1}^{M_\alpha}\frac{1}{q_{\boldsymbol{\theta}_\alpha}(x_i)}, \\ &L(\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\theta}_\alpha) = \tilde{L}_{\text{stab}} + \lambda_1 L_{\text{lip}} + \lambda_2 L_{\alpha^{-1}}. \end{aligned} \tag{11}$$

The specific training procedure of this algorithm, dubbed Neural ETC-MC, is shown in Algorithm 2.

Remark 3.4 *To obtain a stronger exponential decay rate of* V*, we multiply the term* $\alpha(\|\boldsymbol{x}\|)$ *in Eq.* (9) *by* $\sigma \in (0,1)$ *in realization. Similarly to Eq.* (4)*, the controlled vector under event* $\tilde{h}$ *satisfied*

$$\nabla V \cdot (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}+\boldsymbol{e})) \leq -(1-\sigma)\alpha(\|\boldsymbol{x}\|). \tag{12}$$

Then the exponential decay rate of V *is* $1-\sigma$*. The lower bound of inter-event time can be obtained by replacing* $l_{\alpha^{-1}}$ *with* $\sigma^{-1} l_{\alpha^{-1}}$ *in Theorem 3.3.*

4 Theoretical Guarantee for Stability and Optimality

In this section, we provide several theoretical results for rigorously guaranteeing the stability and optimality of our neural controllers. Firstly, we note that the NNs trained on finite samples cannot guarantee the Lyapunov stability condition in the loss function is satisfied in the whole state space with infinite data points. To circumvent this weakness, we introduce the projection operation in the following theorem.

Theorem 4.1 *(Stability guarantee) For a candidate controller $\boldsymbol{u}$ and the stable controller space $\mathcal{U}(V) = \{\boldsymbol{u} : \mathcal{L}_{\boldsymbol{f_u}} V + V \leq 0\}$, we define the projection operator as,*

$$\pi(\boldsymbol{u}, \mathcal{U}(V)) \triangleq \boldsymbol{u} - \frac{\max(0, \mathcal{L}_{\boldsymbol{f_u}} V - V)}{\|\nabla V\|^2} \cdot \nabla V.$$

If the controller has affine actuator, then we have $\pi(\boldsymbol{u}, \mathcal{U}(V)) \in \mathcal{U}(V)$. Furthermore, under the triggering mechanism

$$\nabla V(\boldsymbol{x}) \cdot [\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}+\boldsymbol{e})) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))] - \sigma V(\boldsymbol{x}) = 0,$$
$$\sigma \in (0,1), \boldsymbol{e} = \boldsymbol{x}(t_k) - \boldsymbol{x}(t), t \in [t_k, t_{k+1})$$

the controlled system under $\pi(\boldsymbol{u}, \mathcal{U})$ is assured exponential stable with decay rate $1-\sigma$, and the inter-event time has positive lower bound.

We provide the proof in Appendix A.1.4. By applying the projection operation to the learned controller $\boldsymbol{u_\phi}$ and potential function $V_{\boldsymbol{\theta}}$, we obtain the theoretical stability guarantee for our approach. Based on the Theorem 4.1 and Theorem 3.2, we could provide necessary condition for the optimal event-triggered control with the largest minimal inter-event time by utilizing the lower bound of the inter-event time and the projection operation.

Theorem 4.2 *(Optimality guarantee) Denote the Lipschitz constant of the controller u on state space as l_u, then the optimal control with the largest minimal inter-event time satisfies,*

$$\boldsymbol{u} \in \arg\min_{\mathcal{U}(V)} l_{\boldsymbol{u}}. \tag{13}$$

Furthermore, for any candidate controllers u, the optimal condition can be simplified as

$$\pi(\boldsymbol{u}, \mathcal{U}(V)) \in \arg\min l_{\pi(\boldsymbol{u}, \mathcal{U}(V))}. \tag{14}$$

This theorem is a direct result from the Theorem 4.1 and Theorem 3.2, and the projection operation simplifies the constrained necessary condition in Eq. 13 to the unconstrained condition Eq. 14. We can easily provide optimality guarantee for the neural network controller $\boldsymbol{u_\phi}$ and the Lyapunov function $V_{\boldsymbol{\theta}}$ by regularizing the Lipschitz constant of $\pi(\boldsymbol{u_\phi}, \mathcal{U}(V_{\boldsymbol{\theta}}))$.

5 Experiments and Analysis

In this section, we demonstrate the superiority of the Neural ETCs over existing methods using series of experiments from low dimensional tasks to high dimensional tasks, then we unravel the key factors of Neural ETCs. More details of the experiments can be found in Appendix A.3.

5.1 Benchmark Experiments

Benchmark dynamical systems. (1) Gene Regulatory Network (GRN) plays a central role in describing the gene expression levels of mRNA and proteins in cell [28], here we consider a two-node GRN, $\dot{x}_1 = a_1 \frac{x_1^n}{s^n + x_1^n} + b_1 \frac{s^n}{s^n + x_2^n} - kx_1,\ \dot{x}_2 = a_2 \frac{x_2^n}{s^n + x_2^n} + b_2 \frac{s^n}{s^n + x_1^n} - kx_2$, where the tunable parameters a_1, a_2, b_1 and b_2 represent the strengths of auto or mutual regulations. We aim at stabilizing the system from one attractor to another attractor via only tuning a_1 in time interval $[0, 20]$.

(2) Lorenz system is a fundamental model in atmospheric science [29]: $\dot{x} = \sigma(y - x), \dot{y} = \rho x - y - xz, \dot{z} = xy - \beta z$. For this chaotic system, we stabilize its unstable zero solution by an fully actuated controller $\boldsymbol{u} = (\boldsymbol{u}_x, \boldsymbol{u}_y, \boldsymbol{u}_z)$ in time interval $[0, 2]$.

(3) Michaelis–Menten model for subcellular dynamics (Cell) captures the collective behavior of the coupled cells [30]: $\dot{x}_i = -Bx_i + \sum_{i=1}^{n} A_{ij} \frac{x_j^2}{1+x_j^2}$. This model has two important equilibrium phase,

inactive phase indicating the malignant state and active state indicating the benign state. We consider $n = 100$ and regulate the high dimensional model from the inactive phase to the active phase by only tuning the topology structure $\{A_{ii}\}_{i=1}^{n}$ in time interval $[0, 30]$.

All these systems have application scenarios and urgently call for event-triggered control with minimal communication burden, we summarize the motivation for selecting the them in Appendix A.3.5.

Benchmark methods. We benchmark against the extensively used neural Lyapunov control (NLC) method [4], an improvement version of neural Lyapunov control via constructing quadratic Lyapunov function proposed in [31], dubbed as Quad-NLC here, a integral reinforcement learning (IRL) based ETC [32] and a critic-actor neural network based ETC method [33]. We also compare with the classic linear quadratic regulator (LQR) method, BALSA [34], an online control policy based on the quadratic programming (QP) solver, and our Neural ETC variants: Neural ETC-PI and Neural ETC-MC. We implement all the control methods with the similar kinds of event functions proposed in Eqs. (3),(9). For a fair comparison, we set the number of hidden units per layer such that all learning models have nearly the same number of total parameters. We provide further details of model selection, hyperparameter selection and experimental configuration in Appendix A.3.

Table 1: Comparison studies of benchmark models and dynamical systems. Best results bolded. Averaged over 5 runs. The dimension of tasks are: GRN (2-D), Lorenz (3-D), Cell (100-D).

Method	Number of triggers ↓			Minimal inter-event time ↑			MSE after 10 triggers ↓		
	GRN	Lorenz	Cell	GRN	Lorenz	Cell	GRN	Lorenz	Cell
BALSA [34]	12	273	19	0.29	6e-4	3e-3	**0.05**	7.20	35.64
LQR [35]	1816	2000	449	6e-3	2e-5	0.02	2.19	53.02	2e-3
Quad-NLC [31]	1914	242	77	6e-6	2e-5	1e-3	2.29	7.82	54.38
NLC [4]	23	1602	15	5e-8	4e-8	6e-6	0.20	97.11	27.76
IRL ETC [32]	131	2000	370	8e-3	0.00	3e-3	4.94	9.76	38.12
Cirtic-Actor NNETC [33]	605	50	330	5e-8	2e-3	1e-3	4.1	7.16	38.13
Neural ETC-PI (ours)	20	20	11	0.25	0.02	0.95	**0.05**	**0.11**	**5e-8**
Neural ETC-MC (ours)	**4**	**11**	**2**	**15.52**	**0.06**	**27.18**	0.07	0.14	1.66

Results. Table 1 summarizes the control performance results in terms of the triggering times in the same temporal length, the minimal inter-event time and the mean square error (MSE) between the target state and the controlled trajectories after 10 triggering events, representing the control performance in limited communication resources. We see that our Neural ETC variants achieve superior performance compared to the other online and offline methods.

For the communication cost, our Neural ETCs need the least number of triggers in the same time interval while have the largest minimal inter-event time compared to other methods, leading to the most optimal scheduling in actual implementation. In addition, the MSE results illustrate our Neural ETCs have the ability in stabilizing the systems at various scales with limited communication resources. We also find the Neural ETC-PI and Neural ETC-MC form the trade off in scheduling and the stabilization performance, we further compare them in the next section.

The results underpin the practicability of the Neural ETCs. Take GRN model for an example, the auto regulation strength a_1 can be adjusted externally through the application of repressive or inductive drugs in a typical experimental setting [36]. In reality, the drugs can only be administered a few times and it takes time for the drug to take effect, requiring the controller should only be updated at several times with large interval. Therefore, while all the benchmark methods successfully regulate the GRN to the target gene expression level in simulation, only the Neural ETC-MC is acceptable.

Combining online and offline policy. In the context of event-triggered control, Table 1 demonstrates that the online control method outperforms other offline policies. However, the online policy's computational cost is high due to solving the quadratic programming (QP) problem at each realization time. In contrast, our Neural ETCs achieve superior performance compared to online methods while maintaining the same computation cost as the offline policy during the control process. The event-triggered control employs an event function that continuously assesses whether an event is triggered, effectively acting as an online solver to determine real-time control values. Consequently, we can view event-triggered control as an online realization of the offline policy, inheriting the advantages of both online and offline approaches

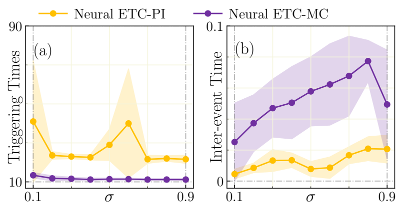

Figure 3: The solid lines are obtained through averaging the 5 sampled trajectories, while the shaded areas stand for the variance regions.

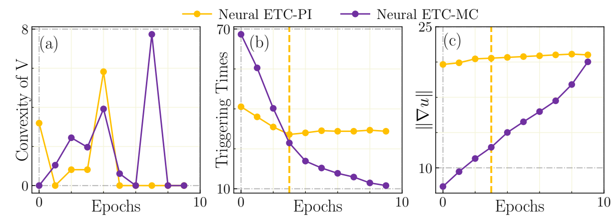

Figure 4: (a) Convexity of V is calculated as the trace of the $\nabla^2 V$ on 1000 points in $[-2.5, 2.5]^3$. (b) Triggering times and norm of $\nabla \boldsymbol{u}$ in the training process.

5.2 Comparison Between Neural ETCs

Table 2: Comparison of Neural ETCs (denoted by NETC) in terms of training time and variance of stabilized trajectories.

Model	Training time ↓		Temporal variance ↓	
	NETC-PI	NETC-MC	NETC-PI	NETC-MC
GRN	1230	**32**	**5e-4**	7e-3
Lorenz	503	**29**	**4e-3**	0.09
FHN	4634	**62**	**3e-15**	2.78

We further evaluate the strengths and weaknesses of the Neural ETC-PI and Neural ETC-MC. As shown in Table 2, the Neural ETC-MC is more efficient in training process, especially in the high dimensional tasks. Nevertheless, the temporal variance of the controlled trajectories of Neural ETC-PI is far below that of Neural ETC-MC, implying Neural ETC-PI is more robust in the control process. These two algorithms thus are complementary in applications. In addition, the training time of Neural ETC-PI in 2-D GRN and 3-D Lorenz has significant difference, the reason is that the minimal inter-event time of the former is larger than the latter (see Table 1), requiring more time to solve t_1.

5.3 Ablation Study

Table 3: Control performance against different event loss weight λ_2.

Method	Neural ETC-PI			Neural ETC-MC		
λ_2	0.005	0.05	0.5	0.01	0.1	1.0
Triggering times ↓	114	29	34	37	**10**	**10**
Min Inter-event time ↑	0.010	0.008	0.025	0.02	**0.07**	0.06
$\langle \text{MSE} \rangle_{[1.8,2]}$↓	**8e-8**	7e-4	0.32	3.92	0.25	0.53

The parameter σ corresponds to the exponential decay rate of Lyapunov function along controlled trajectory in Eqs. (3),(12). We investigate the influence of σ in applying Neural ETC variants to Lorenz dynamic. The results in Fig. 3 suggests the best choice is $\sigma = 0.8$. Then we investigate the influence of weight factor λ_2 of event loss in Eqs. (6),(11) and summarize the results in Table 3. We find the small λ_2 leads to poor triggering scheduling because the event loss does not play a leading role in training, the large λ_2 will break the stabilization performance because the optimization function of event loss is not guaranteed to satisfy the stabilization loss. This phenomenon inspires us to extend the framework to the setting where the parameterized controllers are already stabilization controllers in the future work. For reference, in Table 1 Neural ETC-PI is using $\sigma = 0.5$, $\lambda_2 = 0.05$ and Neural ETC-MC is using $\sigma = 0.5$, $\lambda_2 = 0.1$.

5.4 Essential Factor of Neural ETC

We investigate the essential factor in the Neural ETC framework that determine the optimization of scheduling. We plot the convexity of V function (Tr$(\nabla^2 V)$) and the strength of the variation of controller ($\|\nabla \boldsymbol{u}\|$) in the training process, and compare their evolution with triggering times of the corresponding trained controller. Fig. 4 shows that the $\|\nabla \boldsymbol{u}\|$ plays a leading role in minimizing the triggering times as it has strong negative correlation to the triggering times while the convexity of V function does not. We also observe an early convergence phenomenon of the triggering times and $\|\nabla \boldsymbol{u}\|$ simultaneously in Neural ETC-PI from Fig. 4(b).

6 Related Work

The pioneering works [21, 13] highlighted the advantages of event-based control against the periodic implementation in reducing the communication cost. Since then, [14] investigates the sufficient conditions for avoiding the Zeno behavior in event-triggered implementations of stabilizing feedback control laws and [15] gives the system theory of event-triggered control scheme for perturbed linear systems. Machine learning methods have also been introduced to the ETC settings, [32, 33] employ the critic-actor RL structure to solve the dynamic Hamilton-Jacobi-Bellman equation under the ETC, [37] cultivates a model-free hierarchical RL method to optimize both the control and communication policies for discrete dynamics, and [38] applies deep RL to ETC in the nonlinear systems. All the previous works focus on the stabilization analysis of the controlled systems, the existence of the minimal inter-event time (and hence avoids the Zeno behavior), and directly introducing machine learning methods to ETC. To our knowledge, we are the first to study the optimization scheduling problem of ETC in the continuous dynamics. We provide more discussion of related works in Appendix A.4.

7 Scope and Limitations

ODE solver. The use of the fixed step ODE solvers in finding the triggering times in the training process is less optimal than the adaptive ODE solver. One can still improve the performance of the framework by applying the adaptive solvers with higher accuracy tolerance with a stronger computing platform. However, in practice the performance of the Neural ETC did not decrease substantially when using adaptive solvers. In addition, the employ of ODE solvers in the Neural ETC-PI may not always work, especially for systems described by stiff equations, stiff-based ODE solvers can be introduced to mitigate this issue [39].

Neural ETC for SDEs. Although the current Neural ETC framework works efficiently in ODEs, many real-world scenarios affected by the noise are described by stochastic differential equations (SDEs). The major challenge for establishing the Neural ETC framework for SDEs ensues from the stochasticity of the triggering time. Specifically, the triggering time in SDEs, $t_1 = \inf_{t\geq 0}\{t : h(\boldsymbol{x}(t)) = 0\}$ initiated from any fixed $\boldsymbol{x}(0)$ with $h(\boldsymbol{x}(0)) < 0$, is a stopping time. Therefore, t_1 is a random variable and can take different values in different sample paths. In this case, none of the existing methods can find t_1 for SDEs as a counterpart of `ODESolveEvent` for ODEs.

8 Conclusion

This work focuses on a new connection of machine learning and control field in the context of learning event-triggered stabilization control with optimal scheduling. In contrast to the existing learning control methods, the learned event-triggered control, named Neural ETC, only updates the control value in very few times when an event is triggered. As a consequence, our Neural ETC can be deployed on the actual platform where the communication cost for updating the control value is limited (e.g. tuning the protein regulation strength in cell via drugs). The superiority of the Neural ETC over the existing methods is demonstrated through the benchmark experiments, including different scales of dynamical systems.

References

[1] Hassan K Khalil. Nonlinear systems third edition. *Patience Hall*, 115, 2002.

[2] Pablo A Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. California Institute of Technology, 2000.

[3] Hiroyasu Tsukamoto, Soon-Jo Chung, and Jean-Jaques E Slotine. Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview. *Annual Reviews in Control*, 52:135–169, 2021.

[4] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 3245–3254, 2019.

[5] Jingdong Zhang, Qunxi Zhu, and Wei Lin. Neural stochastic control. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[6] Jingdong Zhang, Qunxi Zhu, Wei Yang, and Wei Lin. Sync: Safety-aware neural control for stabilizing stochastic delay-differential equations. In *The Eleventh International Conference on Learning Representations*, 2022.

[7] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. Learning safe multi-agent control with decentralized neural barrier certificates. In *International Conference on Learning Representations*, 2020.

[8] Dawei Sun, Susmit Jha, and Chuchu Fan. Learning certified control using contraction metric. In *Conference on Robot Learning*, pages 1519–1539. PMLR, 2021.

[9] Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022.

[10] Gene F Franklin, J David Powell, Abbas Emami-Naeini, and J David Powell. *Feedback control of dynamic systems*, volume 4. Prentice hall Upper Saddle River, 2002.

[11] Michael Lemmon. Event-triggered feedback in control, estimation, and optimization. *Networked control systems*, pages 293–358, 2010.

[12] Wilhelmus PMH Heemels, Karl Henrik Johansson, and Paulo Tabuada. An introduction to event-triggered and self-triggered control. In *2012 ieee 51st ieee conference on decision and control (cdc)*, pages 3270–3285. IEEE, 2012.

[13] Karl-Erik Åarzén. A simple event-based pid controller. *IFAC Proceedings Volumes*, 32(2):8687–8692, 1999.

[14] Paulo Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic control*, 52(9):1680–1685, 2007.

[15] WPMH Heemels, JH Sandee, and PPJ Van Den Bosch. Analysis of event-driven controllers for linear systems. *International journal of control*, 81(4):571–590, 2008.

[16] Toivo Henningsson, Erik Johannesson, and Anton Cervin. Sporadic event-based control of first-order linear stochastic systems. *Automatica*, 44(11):2890–2895, 2008.

[17] Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*. MIT press, 2019.

[18] Xuerong Mao. *Stochastic differential equations and applications*. Elsevier, 2007.

[19] Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 2023.

[20] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[21] Karl Johan Åström and Bo Bernhardsson. Comparison of periodic and event based sampling for first-order stochastic systems. *IFAC Proceedings Volumes*, 32(2):5006–5011, 1999.

[22] Marek Miskowicz. *Event-based control and signal processing*. CRC press, 2018.

[23] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

[24] Ricky TQ Chen, Brandon Amos, and Maximilian Nickel. Learning neural event functions for ordinary differential equations. In *International Conference on Learning Representations*, 2020.

[25] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. Routledge, 2018.

[26] Steven George Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

[27] Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. *Advances in neural information processing systems*, 32, 2019.

[28] Eric Davidson and Michael Levin. Gene regulatory networks. *Proceedings of the National Academy of Sciences*, 102(14):4935–4935, 2005.

[29] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[30] Hillel Sanhedrai, Jianxi Gao, Amir Bashan, Moshe Schwartz, Shlomo Havlin, and Baruch Barzel. Reviving a failed network through microscopic interventions. *Nature Physics*, 18(3):338–349, 2022.

[31] Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. Neural certificates for safe control policies. *arXiv preprint arXiv:2006.08465*, 2020.

[32] Shan Xue, Biao Luo, Derong Liu, and Ying Gao. Neural network-based event-triggered integral reinforcement learning for constrained h tracking control with experience replay. *Neurocomputing*, 513:25–35, 2022.

[33] Songsong Cheng, Haoyun Li, Yuchao Guo, Tianhong Pan, and Yuan Fan. Event-triggered optimal nonlinear systems control based on state observer and neural network. *Journal of Systems Science and Complexity*, 36(1):222–238, 2023.

[34] David D Fan, Jennifer Nguyen, Rohan Thakker, Nikhilesh Alatur, Ali-akbar Agha-mohammadi, and Evangelos A Theodorou. Bayesian learning-based adaptive control for safety critical systems. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 4093–4099. IEEE, 2020.

[35] Henning Schlüter, Friedrich Solowjow, and Sebastian Trimpe. Event-triggered learning for linear quadratic control. *IEEE Transactions on Automatic Control*, 66(10):4485–4498, 2020.

[36] Le-Zhi Wang, Ri-Qi Su, Zi-Gang Huang, Xiao Wang, Wen-Xu Wang, Celso Grebogi, and Ying-Cheng Lai. A geometrical approach to control and controllability of nonlinear dynamical networks. *Nature communications*, 7(1):11323, 2016.

[37] Niklas Funk, Dominik Baumann, Vincent Berenz, and Sebastian Trimpe. Learning event-triggered control from data through joint optimization. *IFAC Journal of Systems and Control*, 16:100144, 2021.

[38] Dominik Baumann, Jia-Jie Zhu, Georg Martius, and Sebastian Trimpe. Deep reinforcement learning for event-triggered control. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 943–950. IEEE, 2018.

[39] Suyong Kim, Weiqi Ji, Sili Deng, Yingbo Ma, and Christopher Rackauckas. Stiff neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(9), 2021.

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

[41] Edward Ott, Celso Grebogi, and James A Yorke. Controlling chaos. *Physical review letters*, 64(11):1196, 1990.

[42] Stefanos Boccaletti, Celso Grebogi, Y-C Lai, Hector Mancini, and Diego Maza. The control of chaos: theory and applications. *Physics reports*, 329(3):103–197, 2000.

[43] Mahmoud Abdelrahim, Romain Postoyan, Jamal Daafouz, and Dragan Nešić. Stabilization of nonlinear systems using event-triggered output feedback controllers. *IEEE transactions on automatic control*, 61(9):2682–2687, 2015.

[44] Peter Giesl and Sigurdur Hafstein. Review on computational methods for lyapunov functions. *Discrete and Continuous Dynamical Systems-B*, 20(8):2291–2331, 2015.

[45] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[46] Andrew Taylor, Andrew Singletary, Yisong Yue, and Aaron Ames. Learning for safety-critical control with control barrier functions. In *Learning for Dynamics and Control*, pages 708–717. PMLR, 2020.

[47] Andrew J Taylor, Victor D Dorobantu, Hoang M Le, Yisong Yue, and Aaron D Ames. Episodic learning with control lyapunov functions for uncertain robotic systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6878–6884. IEEE, 2019.

[48] Andrew J Taylor and Aaron D Ames. Adaptive safety with control barrier functions. In *2020 American Control Conference (ACC)*, pages 1399–1405. IEEE, 2020.

[49] Andrea Peruffo, Daniele Ahmed, and Alessandro Abate. Automated and formal synthesis of neural barrier certificates for dynamical models. In *International conference on tools and algorithms for the construction and analysis of systems*, pages 370–388. Springer, 2021.

[50] Sumeet Singh, Spencer M Richards, Vikas Sindhwani, Jean-Jacques E Slotine, and Marco Pavone. Learning stabilizable nonlinear dynamics with contraction-based regularization. *The International Journal of Robotics Research*, 40(10-11):1123–1150, 2021.

A Appendix

A.1 Proofs and Derivations

In this section, we introduce some basic notations and then provide the proofs of the theoretical results.

A.1.1 Notations

Notations. Throughout the paper, we employ the following notation. Let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ be the inner product of vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. For a second continuous function $f(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$, let ∇f denote the gradient of $f(\boldsymbol{x})$, that is, $\nabla^2 f$ denote the Hessian matrix of f. For the two sets A, B, let $A \subset B$ denote that A is covered in B. Denote by $\log$ the base e logarithmic function. Denote by $\|\cdot\|$ the L^2-norm for any given vector in $\mathbb{R}^d$. Denote by $|\cdot|$ the absolute value of a scalar number or the modulus length of a complex number. For $A = (a_{ij})$, a matrix of dimension $d \times r$, denote by $\|A\|_{\mathrm{F}}^2 = \sum_{i=1}^d \sum_{j=1}^r a_{ij}^2$ the Frobenius norm.

A.1.2 Proof of Theorem 3.2

Theorem A.1 *Consider the event-triggered controlled dynamics in Eq.* (1)*, if the following assumptions are satisfied:* (i) $\|\boldsymbol{f}(\boldsymbol{x}', \boldsymbol{u}') - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})\| \leq l_{\boldsymbol{f}} (\|\boldsymbol{x}' - \boldsymbol{x}\| + \|\boldsymbol{u}' - \boldsymbol{u}\|)$*;* (ii) $\|\boldsymbol{u}(\boldsymbol{x}') - \boldsymbol{u}(\boldsymbol{x})\| \leq l_{\boldsymbol{u}} \|\boldsymbol{x}' - \boldsymbol{x}\|$*;* (iii) $\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}}} V(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x} + \boldsymbol{e})) \leq -\alpha(\|\boldsymbol{x}\|) + \gamma(\|\boldsymbol{e}\|)$ *for some class-K functions* α*,* γ *with* $\alpha^{-1}(\gamma(\|\boldsymbol{e}\|)) \leq P\|\boldsymbol{e}\|$*. Then, the minimal inter-event time implicitly defined by event function* $h = \alpha(\|\boldsymbol{x}\|) - \gamma(\|\boldsymbol{e}\|)$ *is lower bounded by* $\tau_h = \frac{1}{l_{\boldsymbol{f}}} \log \frac{P+1}{P + \frac{l_{\boldsymbol{f}} l_{\boldsymbol{u}}}{l_{\boldsymbol{f}}(1+l_{\boldsymbol{u}})}}$.

From the condition (iii) and the definition of the event function, we have the triggering time happens after $P\|\boldsymbol{e}\| = \|\boldsymbol{x}\|$. Therefore, the inter-event time is lower bounded by the minimal inter-event time defined by the event function $\tilde{h} = P(\|\boldsymbol{e}\|) - \|\boldsymbol{x}\|$. Now we come to deduce the estimation of the inter-event time of $\tilde{h}$, i.e., the time from $\|\boldsymbol{e}\| = 0$ to $\|\boldsymbol{e}\| = \frac{1}{P}\|\boldsymbol{x}\|$. Consider the dynamic of $\dfrac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|}$, we have

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} \frac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|} &= \frac{\mathrm{d}}{\mathrm{d}t} \frac{(\boldsymbol{e}^\top \boldsymbol{e})^{1/2}}{(\boldsymbol{x}^\top \boldsymbol{x})^{1/2}} \\
&= \frac{\frac{1}{2}(\boldsymbol{e}^\top \boldsymbol{e})^{-1/2} 2\boldsymbol{e}^\top \dot{\boldsymbol{e}} (\boldsymbol{x}^\top \boldsymbol{x})^{1/2} - \frac{1}{2}(\boldsymbol{x}^\top \boldsymbol{x})^{-1/2} 2\boldsymbol{x}^\top \dot{\boldsymbol{x}} (\boldsymbol{e}^\top \boldsymbol{e})^{1/2}}{\boldsymbol{x}^\top \boldsymbol{x}} \\
&= \frac{\boldsymbol{e}^\top \dot{\boldsymbol{e}}}{\|\boldsymbol{e}\|\|\boldsymbol{x}\|} - \frac{\boldsymbol{x}^\top \dot{\boldsymbol{x}}}{\|\boldsymbol{x}\|\|\boldsymbol{x}\|} \frac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|} \\
&= -\frac{\boldsymbol{e}^\top \dot{\boldsymbol{x}}}{\|\boldsymbol{e}\|\|\boldsymbol{x}\|} - \frac{\boldsymbol{x}^\top \dot{\boldsymbol{x}}}{\|\boldsymbol{x}\|\|\boldsymbol{x}\|} \frac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|} \\
&\leq \frac{\|\boldsymbol{e}\|\|\dot{\boldsymbol{x}}\|}{\|\boldsymbol{e}\|\|\boldsymbol{x}\|} + \frac{\|\boldsymbol{x}\|\|\dot{\boldsymbol{x}}\|}{\|\boldsymbol{x}\|\|\boldsymbol{x}\|} \frac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|} \\
&= \frac{\|\dot{\boldsymbol{x}}\|}{\|\boldsymbol{x}\|} \left(1 + \frac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|}\right) \\
&= \frac{\|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x} + \boldsymbol{e}))\|}{\|\boldsymbol{x}\|} \left(1 + \frac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|}\right) \\
&\leq \frac{l_{\boldsymbol{f}}\|\boldsymbol{x}\| + l_{\boldsymbol{f}} l_{\boldsymbol{u}}(\|\boldsymbol{x}\| + \|\boldsymbol{e}\|)}{\|\boldsymbol{x}\|} \left(1 + \frac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|}\right) \\
&= \left(l_{\boldsymbol{f}}(1 + l_{\boldsymbol{u}}) + l_{\boldsymbol{f}} l_{\boldsymbol{u}} \frac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|}\right) \left(1 + \frac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|}\right).
\end{aligned}
$$

By denoting $z = \dfrac{\|\boldsymbol{e}\|}{\|\boldsymbol{x}\|}$, we have the triggering time of $\tilde{h}$ happens after the variable z increases from 0 to $\frac{1}{P}$. The dynamic of z is

$$\begin{aligned}\dot{z} &= (l_{\boldsymbol{f}}(1+l_{\boldsymbol{u}}) + l_{\boldsymbol{f}} l_{\boldsymbol{u}} z)(1+z)\\ z_0 &= 0,\\ z_T &= \frac{1}{P}.\end{aligned}$$

We have

$$\frac{\mathrm{d}z}{(1+az)(1+y)} = b\mathrm{d}t,$$

where $a = \frac{l_{\boldsymbol{f}} l_{\boldsymbol{u}}}{l_{\boldsymbol{f}}(1+l_{\boldsymbol{u}})}$, $b = l_{\boldsymbol{f}}(1+l_{\boldsymbol{u}})$. Then we have

$$\begin{aligned}\frac{\mathrm{d}z}{(1+az)(1+z)} &= \frac{a}{a-1}\left(\frac{1}{1+az} - \frac{1}{a(1+z)}\right)\mathrm{d}z\\ &= \frac{1}{a-1}\left(\mathrm{d}\log(1+az) - \mathrm{d}\log(1+z)\right)\\ &= b\mathrm{d}t\end{aligned}$$

By integrating the above equation, we have

$$\begin{aligned}&\frac{1}{a-1}\left(\log(1+\frac{a}{P}) - \log(1+\frac{1}{P})\right) = bT\\ &\rightarrow T = \frac{1}{b(a-1)}\log\left(\frac{1+\frac{a}{P}}{1+\frac{1}{P}}\right)\\ &= \frac{1}{b(1-a)}\log\left(\frac{1+\frac{1}{P}}{1+\frac{a}{P}}\right)\\ &= \frac{1}{l_{\boldsymbol{f}}}\log\frac{P+1}{P+\frac{l_{\boldsymbol{f}} l_{\boldsymbol{u}}}{l_{\boldsymbol{f}}(1+l_{\boldsymbol{u}})}},\end{aligned}$$

which completes the proof.

A.1.3 Proof of Theorem 3.3

Theorem A.2 *For the event-triggered controlled dynamics in Eq.* (1) *with event function* $\tilde{h}$ *defined in Eq.* (9)*, if the state space* $\mathcal{D}$ *is bounded, the Eqs.* (7),(8) *and the conditions* (*i*), (*ii*) *in Theorem 3.2 hold, then the minimal inter-event time is lower bounded by* $\tau_{\tilde{h}} = \frac{1}{l_{\boldsymbol{f}}}\log\frac{cl_{\alpha^{-1}}l_{\boldsymbol{u}}+1}{cl_{\alpha^{-1}}l_{\boldsymbol{u}}+\frac{l_{\boldsymbol{f}} l_{\boldsymbol{u}}}{l_{\boldsymbol{f}}(1+l_{\boldsymbol{u}})}}$, *here* $l_{\alpha^{-1}}$ *is the Lipschitz constant of* α^{-1}.

From the Eqs. (7), we know that the triggering time defined by $\tilde{h}$ in Eq. 9 is larger than that defined by h in Theorem 3.2. Notice in Theorem 3.2 P is a tight upper bound Lipschitz constant of $\alpha^{-1}\circ\gamma$. Since the state space $\mathcal{D}$ is bounded, from Eq. 7, if we set γ as the tight estimation of $\nabla V\cdot(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}(\boldsymbol{x}+\boldsymbol{e})) - \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}(\boldsymbol{x})))$, the Lipschitz constant of γ can be bounded by

$$\max_{\boldsymbol{x}\in\mathcal{D}}\|\nabla V(\boldsymbol{x})\| l_{\boldsymbol{f}} l_{\boldsymbol{u}}.$$

Then we get

$$\mathrm{Lip}(\alpha^{-1}\circ\gamma) \le \max_{\boldsymbol{x}\in\mathcal{D}}\|\nabla V(\boldsymbol{x})\| l_{\boldsymbol{f}} l_{\boldsymbol{u}} l_{\alpha^{-1}}.$$

By denoting $c = \max_{\boldsymbol{x}\in\mathcal{D}}\|\nabla V(\boldsymbol{x})\| l_{\boldsymbol{f}}$ and replace P with $cl_{\alpha^{-1}}l_{\boldsymbol{u}}$ in Theorem 3.2, we obtain the final estimation of $\tau_{\tilde{h}}$.

A.1.4 Proof of Theorem 4.1

To begin with, we check the inequality constraint in $\mathcal{U}(V)$ is satisfied by the projection element, that is

$$\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}}} V\big|_{\boldsymbol{u}=\pi(\boldsymbol{u},\mathcal{U}(V))} \le -V.$$

Since the controller has affine actuator, from the definition of the Lie derivative operator, we have

$$\begin{aligned}\mathcal{L}_{\boldsymbol{f_u}} V\big|_{\boldsymbol{u}=\pi(\boldsymbol{u},\mathcal{U}(V))} &= \nabla V \cdot (\boldsymbol{f} + \boldsymbol{u} - \frac{\max(0, \mathcal{L}_{\boldsymbol{u}} V + V)}{\|\nabla V\|^2} \cdot \nabla V) \\ &= \nabla V \cdot (\boldsymbol{f} + \boldsymbol{u}) - \nabla V \cdot \frac{\max(0, \mathcal{L}_{\boldsymbol{u}} V + V)}{\|\nabla V\|^2} \cdot \nabla V \\ &= \mathcal{L}_{\boldsymbol{u}} V - \max(0, \mathcal{L}_{\boldsymbol{u}} V + V) \leq -V.\end{aligned}$$

The positive lower bound of the inter-event time comes from the Theorem 3.2. We complete the proof.

A.2 Algorithms

In this section, we provide the algorithms of Neural ETC-PI (1) and Neural ETC-MC (2). Firstly, we supplement the warm up stage for path integral algorithm to accelerate the convergence of training process.

Warm up. At the beginning of the training process, the stability constraint is not satisfied, which leads to the solution t_1 of the event function $h_{\boldsymbol{\theta},\boldsymbol{\phi}}$ does not exist. To ensure the training process can proceed smoothly, we pre-train the parameterized model with

$$\tilde{L}(\boldsymbol{\phi},\boldsymbol{\theta},\{c_i\}) = L_{\text{stab}} + \lambda_1 L_{\text{lip}}. \tag{15}$$

Algorithm 1: Neural ETC-PI: Path Integral Algorithm

---

hyperparameters:
  N, M ▷ Sample size and batch size
  β, m ▷ Learning rate and max iterations
  $\mu(\mathcal{D}), \lambda_1, \lambda_2$ ▷ Data distrituion, weight factors
initialize $\boldsymbol{w} = (\boldsymbol{\phi},\boldsymbol{\theta})$ ▷ From Eq. (5)
generate dataset $\mathcal{D}_N = \{\boldsymbol{x}_i\}_{i=1}^N \sim \mu(\mathcal{D})$
for $r = 1 : m$ **do**
  $\boldsymbol{w} \leftarrow \boldsymbol{w} - \beta\nabla_{\boldsymbol{w}}\tilde{L}(\boldsymbol{w})$ ▷ Warm up in Eq. (15)
for $r = 1 : m$ **do**
  $\{\boldsymbol{x}_i(0)\}_{i=1}^M \sim \mathcal{D}_N$ ▷ Sample batch data
  $t_{i,1}, \boldsymbol{x}_i(t_{i,1}) = \texttt{ODESolveEvent}(\boldsymbol{x}_i(0), \boldsymbol{f}, \boldsymbol{u}_{\boldsymbol{\phi}}, 0)$
  $\boldsymbol{w} \leftarrow \boldsymbol{w} - \beta\nabla_{\boldsymbol{w}}L(\boldsymbol{w})$ ▷ From Eq. (6)
return $\boldsymbol{u}_{\boldsymbol{\phi}}, V_{\boldsymbol{\theta}}$

Algorithm 2: Neural ETC-MC: Monte Carlo Algorithm

---

hyperparameters:
  $N, M_\alpha, \lambda_1, \lambda_2$ ▷ Sample sizes and weight factors
  β, m ▷ Learning rate and max iterations
  $\mu(\mathcal{D}), \mu(\mathcal{X})$ ▷ Distributions of state and error
initialize $\boldsymbol{w} = (\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\theta}_\alpha)$ ▷ Eqs. (5),(10)
generate dataset $\{\boldsymbol{x}_i\}_{i=1}^N \times \{x_i\}_{i=1}^{M_\alpha} \sim \mu(\mathcal{D}) \times \mu(\mathcal{X})$
for $r = 1 : m$ **do**
  $\boldsymbol{w} \leftarrow \boldsymbol{w} - \beta\nabla_{\boldsymbol{w}}L(\boldsymbol{w})$ ▷ From Eq. (11)
return $\boldsymbol{u}_{\boldsymbol{\phi}}, V_{\boldsymbol{\theta}}, \alpha_{\boldsymbol{\theta}_\alpha}$

A.3 Experimental Configurations

In this section, we provide the detailed descriptions for the experimental configurations of the benchmark dynamical systems and control methods in the main text. We implement the code on a single i7-10870 CPU with 16GB memory, and we train all the parameters with Adam optimizer.

A.3.1 Neural Network Structures

- For constructing the potential function V, we utilize the ICNN as [40]:

$$\begin{aligned} \boldsymbol{z}_1 &= \sigma(W_0\boldsymbol{x} + b_0), \\ \boldsymbol{z}_{i+1} &= \sigma(U_i\boldsymbol{z}_i + W_i\boldsymbol{x} + b_i),\ i = 1, \cdots, k-1, \\ p(\boldsymbol{x}) &\equiv \boldsymbol{z}_k, \\ V(\boldsymbol{x}) &= \sigma(p(\boldsymbol{x}) - p(\boldsymbol{0})) + \varepsilon\|\boldsymbol{x}\|^2, \end{aligned}$$

  where σ is the smoothed **ReLU** function as defined in the main text, $W_i \in \mathbb{R}^{h_i \times d}$, $U_i \in (\mathbb{R}_+ \cup \{0\})^{h_i \times h_{i-1}}$, $\boldsymbol{x} \in \mathbb{R}^d$, and, for simplicity, this ICNN function is denoted by ICNN$(h_0, h_1, \cdots, h_{k-1})$. We set $\varepsilon = $ 1e-3 as default value for all the experiments;

- The class-$\mathcal{K}$ function α is constructed as:

$$\begin{aligned} \boldsymbol{q}_1 &= \text{ReLU}(W_0 s + b_0), \\ \boldsymbol{q}_{i+1} &= \text{ReLU}(W_i\boldsymbol{q}_i + b_i), i = 1, \cdots, k-2, \\ \boldsymbol{q}_k &= \text{ELU}(W_{k-1}\boldsymbol{q}_{k-1} + b_{k-1}), \\ \alpha(x) &= \int_0^x q_k(s)\mathrm{d}s \end{aligned}$$

  where $W_i \in \mathbb{R}^{h_{i+1} \times h_i}$, and this class-$\mathcal{K}$ function is denoted by $\mathcal{K}(h_0, h_1, \cdots, h_k)$;

- The neural control function (nonlinear version) is constructed as:

$$\begin{aligned} \boldsymbol{z}_1 &= \mathcal{F}(\texttt{SpectralNorm}(W_0\boldsymbol{x} + b_0)), \\ \boldsymbol{z}_{i+1} &= \mathcal{F}(\texttt{SpectralNorm}((W_i\boldsymbol{z}_i + b_i)),\ i = 1, \cdots, k-1, \\ \mathbf{NN}(\boldsymbol{x}) &\equiv \texttt{SpectralNorm}(W_k\boldsymbol{z}_k), \\ \boldsymbol{u}(\boldsymbol{x}) &= \text{diag}(\boldsymbol{x} - \boldsymbol{x}^*)\mathbf{NN}(\boldsymbol{x}) \text{ or } \mathbf{NN}(\boldsymbol{x}) - \mathbf{NN}(\boldsymbol{x}^*), \end{aligned}$$

  where $\mathcal{F}(\cdot)$ is the activation function, `SpectralNorm` is the spectral norm function from [23], $W_i \in \mathbb{R}^{h_{i+1} \times h_i}$, and this control function is denoted by Control$(h_0, h_1, \cdots, h_{k+1})$. Since we deploy the `SpectralNorm` package in our algorithm, the weight factor λ_1 for Lipschitz constant of $\boldsymbol{u}$ is automatically set as the default value in this package and we do not tune it in our experiments due to its good performance.

- The standard neural network is constructed as:

$$\begin{aligned} \boldsymbol{z}_1 &= \mathcal{F}(W_0\boldsymbol{x} + b_0), \\ \boldsymbol{z}_{i+1} &= \mathcal{F}(W_i\boldsymbol{z}_i + b_i),\ i = 1, \cdots, k-1, \\ \mathbf{NN}(\boldsymbol{x}) &\equiv W_k\boldsymbol{z}_k, \end{aligned}$$

  where $\mathcal{F}(\cdot)$ is the activation function, and this standard function is denoted by MLP$(h_0, h_1, \cdots, h_{k+1})$

A.3.2 Gene Regulatory Network

Here we model the controlled gene regulatory network (GRN) as

$$\begin{aligned} \dot{x}_1 &= a_1\frac{x_1^n}{s^n + x_1^n} + b_1\frac{s^n}{s^n + x_2^n} - kx_1 + u\frac{x_1^n}{s^n + x_1^n}, \\ \dot{x}_2 &= a_2\frac{x_2^n}{s^n + x_2^n} + b_2\frac{s^n}{s^n + x_1^n} - kx_2, \end{aligned}$$

where the under-actuated control u only acts on the protein regulation strength a_1. We specify $a_1 = a_2 = 1$, $b_1 = b_2 = 0.2$, $n = 2$, $k = 1.1$, $s = 0.5$. The two attractors of the original model is

$$
\begin{aligned}
\boldsymbol{P}_1 &: \ (x_1^*, x_2^*) = (0.62562059, 0.62562059), \\
\boldsymbol{P}_2 &: \ (x_1^0, x_2^0) = (0.0582738, 0.85801853).
\end{aligned}
$$

We aims at stabilize the attractor P_2 with low protein concentration to P_1 with high protein expression level. We slightly modify the neural networks s.t. $V(\boldsymbol{P}_1) = 0$, $u(\boldsymbol{P}_1) = 0$, e.g. $V = V(\boldsymbol{x}) - V(\boldsymbol{P}_1)$, $u = u(\boldsymbol{x}) - u(\boldsymbol{P}_1)$. Since these two attractors are close in the Euclidean space, it hard for algorithms to identify them from states with numerical error. To address this issue, we rescale the original system as $\tilde{x}_1 = 10x_1$, $\tilde{x}_2 = 10x_2$ to enlarge the attractors. For training controller $\boldsymbol{u}$, we uniformly sample 1000 data from the state region $[-10, 10]$. We test the performance under different learning rate $\mathrm{lr} \in \{0.01, 0.03, 0.05\}$ and pick the best one, the considered control methods are set as following,

Neural ETC-PI. We parameterize $V(\boldsymbol{x})$ as $\mathrm{ICNN}(2, 10, 10, 1)$, $\boldsymbol{u}(\boldsymbol{x})$ as $\mathrm{Control}(2, 20, 20, 1)$ with $\mathcal{F} = \mathrm{ReLU}$. We set the iterations for warm up as 500, the iterations and batch size for calculating the triggering times as 50 and 10, the learning rate as $\mathrm{lr} = 0.01$, the weight factor for event loss as $\lambda_2 = \frac{10}{1000}$.

Neural ETC-MC. We parameterize $V(\boldsymbol{x})$ as $\mathrm{ICNN}(2, 20, 1)$, $\boldsymbol{u}(\boldsymbol{x})$ as $\mathrm{Control}(2, 20, 20, 1)$. We set the iterations as $500 + 50$, the learning rate as $\mathrm{lr} = 0.05$, the weight factor for event loss as $\lambda_2 = 0.1$.

NLC. We parameterize $V(\boldsymbol{x})$ as $\mathrm{MLP}(2, 20, 20, 1)$, $\boldsymbol{u}(\boldsymbol{x})$ as $\mathrm{MLP}(2, 20, 20, 1)$. We set the iterations as $500 + 50$, the learning rate as $\mathrm{lr} = 0.05$, the loss function is

$$
L = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_{\boldsymbol{\phi}}}} V_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)^+ + (V_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^+\right] + V_{\boldsymbol{\theta}}(P_1)^2
$$

Quad-NLC. We parameterize $V(\boldsymbol{x})$ as $(\boldsymbol{x} - \boldsymbol{P}_1)^\top \mathrm{MLP}(2, 20, 2)^\top \mathrm{MLP}(2, 20, 2)(\boldsymbol{x} - \boldsymbol{P}_1)$, $\boldsymbol{u}(\boldsymbol{x})$ as $\mathrm{MLP}(2, 20, 20, 1)$. We set the iterations as $500 + 50$, the learning rate as $\mathrm{lr} = 0.05$, the loss function is

$$
L = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_{\boldsymbol{\phi}}}} V_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + V_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)^+ + V_{\boldsymbol{\theta}}(P_1)^2.
$$

BALSA. For this QP based method, we set the object function as

$$
\begin{aligned}
&\min_{u, d_1, d_2} \frac{1}{2}\|u\|^2 + p_1 d_1^2, \\
&\text{s.t.} \mathcal{L}_{\boldsymbol{f}_u} V - V \le d_1,
\end{aligned}
$$

where d_1 is the relaxation number. We choose $V = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{P}_1\|^2, p_1 = 50$. We solve this problem with the QP solver in `cvxopt` in Python package.

LQR. We linearize the controlled dynamic near the target $\boldsymbol{P}_1$ as

$$
\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{P}_1) + \boldsymbol{B}u, \\
\boldsymbol{A} &= \begin{pmatrix} a_1 \dfrac{n(x_1^*)^{n-1}}{(s^n + (x_1^*)^n)^2} - k & -b_1 \dfrac{n(x_2^*)^{n-1}}{(s^n + (x_2^*)^n)^2} \\ -a_2 \dfrac{n(x_1^*)^{n-1}}{(s^n + (x_1^*)^n)^2} & b_2 \dfrac{n(x_2^*)^{n-1}}{(s^n + (x_2^*)^n)^2} - k \end{pmatrix}, \\
\boldsymbol{B} &= \begin{pmatrix} \dfrac{(x_1^*)^n}{(s^n + (x_1^*)^n)^2} - k \\ 0 \end{pmatrix}.
\end{aligned}
$$

We set the cost matrix in LQR as

$$
\begin{aligned}
\boldsymbol{Q} &= \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix}, \\
R &= (0.1)
\end{aligned}
$$

and solve the problem via `lqr` method in Matlab. The obtained Riccati solution $\boldsymbol{S}$ forms the Lyapunov function $V = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{P}_1)^\top \boldsymbol{S}(\boldsymbol{x} - \boldsymbol{P}_1)$, the controller is $u = -\boldsymbol{K}(\boldsymbol{x} - \boldsymbol{P}_1)$ where $\boldsymbol{K} \in \mathbb{R}^{1\times 2}$ is returned by the lqr solver. The Lie derivative of the Lyapunov function is $-(\boldsymbol{x} - \boldsymbol{P}_1)^\top \boldsymbol{Q}_1(\boldsymbol{x} - \boldsymbol{P}_1)$ with $\boldsymbol{Q}_1 = \boldsymbol{Q} + \boldsymbol{K}^\top R\boldsymbol{K}$.

Critic-Actor NNETC. According to the implementation setting in [33], we consider the following event-triggered controller parametrized by the critic neural network W_c and the actor neural network W_a,

$$
\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x}),\\
V^*(\boldsymbol{x}) &= \min_{\boldsymbol{u}} \int_0^T \left(\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u}\right) \mathrm{d}t,\\
V^*(\boldsymbol{x}) &= \boldsymbol{x}^\top \boldsymbol{W}_c \boldsymbol{x},\\
u^*(\boldsymbol{x}) &= \boldsymbol{W}_a \boldsymbol{x},\\
\boldsymbol{e}_a &= \boldsymbol{W}_a \boldsymbol{x} + \frac{1}{2}\boldsymbol{g}^\top \nabla V^*(\boldsymbol{x}),\\
K_a &= \frac{1}{2}\boldsymbol{e}_a^\top \boldsymbol{e}_a,\\
\dot{\boldsymbol{W}}_a &= -\frac{\partial K_a}{\partial \boldsymbol{W}_a},\\
\boldsymbol{e}_c &= \nabla V^*(\boldsymbol{x}) \cdot [\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x})] + \boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u},\\
K_c &= \frac{1}{2}\boldsymbol{e}_c^\top \boldsymbol{e}_c,\\
\dot{\boldsymbol{W}}_c &= -\frac{\partial K_c}{\partial \boldsymbol{W}_c},
\end{aligned}
$$

here $\boldsymbol{f}$ is the original dynamics described above, $\boldsymbol{g}$ is the actuator taking the form,

$$
\boldsymbol{g} = \begin{pmatrix} \dfrac{x_1^n}{(s^n + x_1^n)^2} \\ 0 \end{pmatrix}.
$$

The cost matrix $\boldsymbol{Q}$, $\boldsymbol{R}$ are the same as that in LQR. In the event-triggered mode, the weights of critic and actor NN, $\boldsymbol{W}_c$ and $\boldsymbol{W}_a$, obeying the evolution dynamics as follows,

$$
\begin{aligned}
\dot{\boldsymbol{W}}_a &= \boldsymbol{0},\ t \in [t_k, t_{k+1}),\\
\boldsymbol{W}_a^+ &= \boldsymbol{W}_a - \alpha_a \frac{\partial K_a}{\partial \boldsymbol{W}_a}, t = t_{k+1},\\
\dot{\boldsymbol{W}}_c &= \boldsymbol{0},\ t \in [t_k, t_{k+1}),\\
\boldsymbol{W}_c^+ &= \boldsymbol{W}_c - \alpha_c \frac{\partial K_c}{\partial \boldsymbol{W}_c}, t = t_{k+1},
\end{aligned}
$$

where α_a and α_c are the learning rates of the critic and actor NNs, respectively. For the initial value of $\boldsymbol{W}_c$ and $\boldsymbol{W}_a$, we employ the solutions from the above LQR solver as $\boldsymbol{W}_c = \boldsymbol{S}$, $\boldsymbol{W}_a = -\boldsymbol{K}$. We set the learning rate as $\alpha_c = \alpha_a = 1e-2$, the event function is set as $h = |\boldsymbol{e}| - e_{\text{thres}}$, $e_{\text{thres}} = 0.2$ according to [33], here $\boldsymbol{e} = (e_{x_1}, e_{x_2})$ are the variables of error dynamics. The event function here is different to our proposed stability guaranteed function because of the lack of Lyapunov function in this method, we note that V^* is only an auxiliary function used to find the dynamics of $\boldsymbol{W}_c$, $\boldsymbol{W}_a$ and cannot be verified as a Lyapunov function. We have tuned the hyperparameters $\alpha_c, \alpha_a \in \{5e-4, 1e-3, 1e-2, , 1e-1\}$, $e_{\text{thres}} \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and fix the parameters with the best performance.

IRL ETC. Similarly to the Critic-Actor NNETC, [32] transformed the optimization control problem to a RL problem via abstracting the Hamilton-Jacobi-Bellman equation as the value function and approximating the optimal value function based on a preset basis activation function. Specifically, we consider the control problem parametrized by the critic neural network W as follows,

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x}), \\
V^*(\boldsymbol{x}) &= \min_{\boldsymbol{u}} \int_0^T \left(\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u}\right) \mathrm{d}t, \\
V^*(\boldsymbol{x}) &= \boldsymbol{x}^\top \boldsymbol{W}\boldsymbol{x}, \\
u^*(\boldsymbol{x}) &= \eta\sigma\left(-\frac{1}{2\eta}\boldsymbol{R}^{-1}\boldsymbol{g}^\top \nabla V^*(\boldsymbol{x})\right), \\
E &= \int_t^{t+l} e^{-\alpha(\tau - t)}\left[\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \sum_i \int_0^{u_i} 2\eta\sigma^{-1}(\eta^{-1}s) r_i \mathrm{d}s\right] \mathrm{d}\tau,\ \mathrm{diag}(R) = (r_1, \cdots, r_m), \\
K &= \frac{1}{2}E^2, \\
\dot{\boldsymbol{W}} &= -\frac{\partial K}{\partial \boldsymbol{W}},
\end{aligned}$$

here $\boldsymbol{f}$ is the original dynamics described above, $\boldsymbol{g}$ is the actuator taking the form,

$$\boldsymbol{g} = \begin{pmatrix} \frac{x_1^n}{(s^n + x_1^n)^2} \\ 0 \end{pmatrix}.$$

The cost matrix $\boldsymbol{Q}$, $\boldsymbol{R}$ are the same as that in LQR. In the event-triggered mode, the weight $\boldsymbol{W}$ of critic NN is updated as,

$$\begin{aligned}
\dot{\boldsymbol{W}} &= \boldsymbol{0},\ t \in [t_k, t_{k+1}), \\
\boldsymbol{W}^+ &= \boldsymbol{W} - \beta\frac{\partial K}{\partial \boldsymbol{W}}, t = t_{k+1},
\end{aligned}$$

with β being the learning rates of the weight. We initialize the weight as $\boldsymbol{W} = (W_{ij} = 4)_{2\times 2}$. We set the learning rate as $\beta = 1e - 2$, the event function is set as $h = \|\boldsymbol{e}\|^2 - \frac{(1-\lambda_y^2)\underline{\lambda}(\boldsymbol{Q})}{\eta^2\lambda_x^2}\|\boldsymbol{x}\|^2$, $\lambda_y^2 = 0.6$ according to [33]. In the original work [32], historical data is considered as multiple integral on time interval $[t^j, t^j + l] \subset [t_k, t_{k+1})$ like $E = E_{[t,t+l]}$. To simplify the calculation, here we merge the multiple historical data to a single integral on time interval $[t_k, t_k + l]$ with $l = \min(l^*, t_{k+1} - t_k)$, here $l^* = 1.2$ is a predefined length of historical data. We tuned the hyperparameters in the same way with Critic-Actor NNETC, and the final results are $\alpha = 0.1$, $\eta = 1.0$, $\lambda_x = 0.1$, $\sigma(\cdot) = \mathrm{Id}(\cdot)$.

Test configurations. For implementing the controller in the event-triggered mode, we set the event function for Neural ETC-PI, Quad-NLC, BALSA as

$$\nabla V \cdot (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x} + \boldsymbol{e})) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))) - \sigma V(\boldsymbol{x}),$$

the event function for Neural ETC-MC as

$$\nabla V \cdot (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x} + \boldsymbol{e})) - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))) - \sigma\alpha(\|\boldsymbol{x}\|),$$

the event function for NLC as

$$\nabla V \cdot (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x} + \boldsymbol{e})) - \sigma\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))),$$

the event function for LQR as [12]

$$(\sigma - 1)(\boldsymbol{x} - \boldsymbol{P}_1)^\top \boldsymbol{Q}_1(\boldsymbol{x} - \boldsymbol{P}_1) + 2(\boldsymbol{x} - \boldsymbol{P}_1)^\top \boldsymbol{S}\boldsymbol{B}\boldsymbol{K}\boldsymbol{e},$$

where the σ is set as 0.5 for all models. For the initial value, we set $\boldsymbol{x}_0 = \boldsymbol{P}_2 + \boldsymbol{\xi}_i$, $\boldsymbol{\xi}_i \sim \mathcal{U}[-1, 1]$, $i = 1, \cdots, 5$, the random seed is $(2, 4, 5, 6, 7)$.

A.3.3 Lorenz System

Here we model the state of the Lorenz system under fully actuated control $\boldsymbol{u} = (u_1, u_2, u_3)$ as $\boldsymbol{x} = (x, y, z)^\top$,

$$\begin{aligned}
\dot{x} &= \sigma(y - x) + u_1, \\
\dot{y} &= \rho x - y - xz + u_2, \\
\dot{z} &= xy - \beta z + u_3.
\end{aligned}$$

We aim to stabilize the zero solution of this chaotic system. We consider $\sigma = 10$, $\rho = 28$, $\beta = 8/3$. For training controller $\boldsymbol{u}$, we uniformly sample 5000 data from the state region $[-10, 10]$. We construct the controllers as follows.

Neural ETC-PI. We parameterize $V(\boldsymbol{x})$ as ICNN$(3, 64, 1)$, $\boldsymbol{u}(\boldsymbol{x})$ as Control$(3, 64, 64, 3)$ with $\mathcal{F} =$ ReLU. Since the Ode solver in the training process require high computational resources, we down-sample 2000 data from the original dataset for training. We set the iterations for warm up as 500, the iterations and batch size for calculating the triggering times as 100 and 10, the learning rate as lr $= 0.05$, the weight factor for event loss as $\lambda_2 = \frac{100}{2000}$.

Neural ETC-MC. We parameterize $V(\boldsymbol{x})$ as ICNN$(3, 64, 1)$, $\boldsymbol{u}(\boldsymbol{x})$ as Control$(3, 64, 64, 3)$. We set the iterations as $500 + 100$, the learning rate as lr $= 0.05$, the weight factor for event loss as $\lambda_2 = 0.1$.

NLC. We parameterize $V(\boldsymbol{x})$ as MLP$(3, 64, 64, 1)$, $\boldsymbol{u}(\boldsymbol{x})$ as MLP$(3, 64, 64, 3)$. We set the iterations as $500 + 100$, the learning rate as lr $= 0.05$, the loss function is

$$L = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_\phi}} V_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)^+ + (V_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^+\right] + (V_{\boldsymbol{\theta}}(\boldsymbol{0}))^+,$$

notice that we select the last term in the right hand side as $(V_{\boldsymbol{\theta}}(\boldsymbol{0}))^+$ instead of $V_{\boldsymbol{\theta}}(\boldsymbol{0})^2$ since the former performs better than the latter. We also resample 5000 data from $[-5, 5]$ since the NLC performs poorly in the original dataset, the similar case holds for Quad-NLC.

Quad-NLC. We parameterize $V(\boldsymbol{x})$ as $\boldsymbol{x}^\top$MLP$(3, 64, 3)^\top$MLP$(3, 64, 3)\boldsymbol{x}$, $\boldsymbol{u}(\boldsymbol{x})$ as MLP$(3, 64, 64, 3)$. We set the iterations as $500 + 100$, the learning rate as lr $= 0.05$, the loss function is

$$L = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_\phi}} V_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + V_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)^+ + V_{\boldsymbol{\theta}}(\boldsymbol{0})^2.$$

BALSA. For this QP based method, we set the object function as

$$\begin{aligned}
&\min_{\boldsymbol{u}, d_1, d_2} \frac{1}{2}\|\boldsymbol{u}\|^2 + p_1 d_1^2,\\
&\text{s.t.} \mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}}} V - V \le d_1.
\end{aligned}$$

We choose $V = \frac{1}{2}\|\boldsymbol{x}\|^2, p_1 = 20$.

LQR. We linearize the controlled dynamic near the zero solution as

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u},\\
\boldsymbol{A} &= \begin{pmatrix} -\sigma & \sigma & 0 \\ \rho & -1 & 0 \\ 0 & 0 & -\beta \end{pmatrix}\\
\boldsymbol{B} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}$$

We set the cost matrix in LQR as

$$\begin{aligned}
\boldsymbol{Q} &= \begin{pmatrix} 5 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 5 \end{pmatrix},\\
\boldsymbol{R} &= \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix}.
\end{aligned}$$

The obtained Riccati solution $\boldsymbol{S}$ forms the Lyapunov function $V = \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{S}\boldsymbol{x}$, the controller is $\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x}$ where $\boldsymbol{K} \in \mathbb{R}^{3\times 3}$ is returned by the lqr solver. The Lie derivative of the Lyapunov function is $-\boldsymbol{x}^\top\boldsymbol{Q}_1\boldsymbol{x}$ with $\boldsymbol{Q}_1 = \boldsymbol{Q} + \boldsymbol{K}^\top\boldsymbol{R}\boldsymbol{K}$.

Critic-Actor NNETC. The updating procedure is the same as that in Appendix A.3.2, we set the hyperparameters as $\alpha_c = \alpha_a = 5e-4$, $\mathrm{ethres} = 0.3$, we note that for the chaotic system, the event-triggered dynamics is easy to explode when $\alpha_{c,a}$ are slightly larger than $1e-3$. The actuator in this example is the identity matrix as $\boldsymbol{g} = I_{3\times 3}$.

IRL ETC. The updating procedure is the same as that in Appendix A.3.2, we set the hyperparameters as $\beta = 1e-2$, $\alpha = 0.1$, $\sigma(\cdot) = \tanh(\cdot)$, $\eta = 10$, $\lambda_x = 1.0$, $\lambda_y^2 = 0.6$. The actuator in this example is the identity matrix as $\boldsymbol{g} = I_{3\times 3}$.

Test configurations. We select the same event functions as those for GRN to implement the event-triggered control, except for setting $\sigma = 0.99$ for LQR since it fails in the case $\sigma = 0.5$. For the initial value, we randomly select 5 points in the original dataset using `numpy.random.choice` method in Python, and the random seeds are set as $\{3, 5, 7, 8, 9\}$.

A.3.4 Michaelis–Menten model

Consider the coupled subcellular model under topology control as

$$\dot{x}_i = -Bx_i + \sum_{i=1}^{100} A_{ij}\frac{x_j^2}{1+x_j^2} + \delta A_{ii}\frac{x_i^2}{1+x_i^2}.$$

This dynamic has two attractor, inactive state $\boldsymbol{P}_1 = \boldsymbol{0}$ represents the cell apoptosis and the active $\boldsymbol{P}_2$ represents the reviving cell state. We aim at regulating the cell state to the reviving state through only tuning the diagonal topology structure, which can be achieved experimentally via drugs or electrical stimulation. Therefore, an ideal control should be updated as little as possible since the frequent stimulation may do harm to the cells. For training controller $\boldsymbol{u} = (\delta A_{11}, \cdots, \delta A_{100,100})$, we uniformly sample 1000 data from the state region $[-10, 10]$. Similarly to that in GRN, we modify the parameterized V and $\boldsymbol{u}$ functions s.t. $V(\boldsymbol{P}_2) = 0$, $\boldsymbol{u}(\boldsymbol{P}_2) = 0$. We construct the controllers as follows.

Neural ETC-PI. We parameterize $V(\boldsymbol{x})$ as $\mathrm{ICNN}(100, 64, 1)$, $\boldsymbol{u}(\boldsymbol{x})$ as $\mathrm{Control}(100, 64, 64, 100)$ with $\mathcal{F} = \mathrm{ReLU}$. Since the dimension of the task is very high, the ODE solver has very high computational cost in solving the triggering times. We set the the iterations and batch size for calculating the triggering times as 10 and 5. If the readers have more powerful computing device, larger iterations and batch size are recommended. We set the iterations for warm up as 500, the learning rate as $\mathrm{lr} = 0.01$, the weight factor for event loss as $\lambda_2 = \frac{100}{1000}$. In the case, we try a combination of Neural ETC-PI and Neural ETC-MC by setting the stabilzaition loss as

$$L_{\text{stab}} = \frac{1}{N}\sum_{i=1}^{N}\Big(\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_{\boldsymbol{\phi}}}} V_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + \alpha_{\boldsymbol{\theta}_\alpha}(\|\boldsymbol{x}_i\|)\Big)^+$$

and we also penalize the Lipschitz constant of α^{-1} by adding term $L_{\alpha^{-1}}$ to the loss function with weight 0.1. The dataset $\{x_i\}$ for $L_{\alpha^{-1}}$ is generated by equidistant sampling on $[0, 10]$.

Neural ETC-MC. We parameterize $V(\boldsymbol{x})$ as $\mathrm{ICNN}(100, 200, 1)$, $\boldsymbol{u}(\boldsymbol{x})$ as $\mathrm{Control}(100, 200, 200, 100)$. We set the iterations as 500, the learning rate as $\mathrm{lr} = 0.05$, the weight factor for event loss as $\lambda_2 = 0.1$. The dataset $\{x_i\}$ for $L_{\alpha^{-1}}$ is generated by equidistant sampling on $[0, 5]$.

NLC. We parameterize $V(\boldsymbol{x})$ as $\mathrm{MLP}(100, 200, 200, 1)$, $\boldsymbol{u}(\boldsymbol{x})$ as $\mathrm{MLP}(100, 200, 200, 100)$. We set the iterations as 500, the learning rate as $\mathrm{lr} = 0.01$, the loss function is

$$L = \frac{1}{N}\sum_{i=1}^{N}\left[\Big(\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_{\boldsymbol{\phi}}}} V_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\Big)^+ + (V_{\boldsymbol{\theta}}(\boldsymbol{x}_i))^+\right] + V_{\boldsymbol{\theta}}(\boldsymbol{0})^2.$$

Quad-NLC. We parameterize $V(\boldsymbol{x})$ as $(\boldsymbol{x} - \boldsymbol{P}_2)^\top \mathrm{MLP}(100, 200, 100)^\top \mathrm{MLP}(100, 200, 100)(\boldsymbol{x} - \boldsymbol{P}_2)$, $\boldsymbol{u}(\boldsymbol{x})$ as $\mathrm{MLP}(100, 200, 200, 100)$. We set the iterations as 500, the learning rate as $\mathrm{lr} = 0.01$, the loss function is

$$L = \frac{1}{N}\sum_{i=1}^{N}\Big(\mathcal{L}_{\boldsymbol{f}_{\boldsymbol{u}_{\boldsymbol{\phi}}}} V_{\boldsymbol{\theta}}(\boldsymbol{x}_i) + V_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\Big)^+ + V_{\boldsymbol{\theta}}(\boldsymbol{0})^2.$$

BALSA. For this QP based method, we set the object function as

$$\min_{\boldsymbol{u},d_1,d_2} \frac{1}{2}\|\boldsymbol{u}\|^2 + p_1 d_1^2,$$
$$\text{s.t.}\mathcal{L}_{\boldsymbol{f_u}} V - V \leq d_1.$$

We choose $V = \frac{1}{2}\|\boldsymbol{x}\|^2, p_1 = 50$.

LQR. We linearize the controlled dynamic near the $\boldsymbol{P}_2$ solution as

$$\begin{aligned}&\dot{\boldsymbol{x}} = \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{P}_2) + \boldsymbol{B}\boldsymbol{u},\\ &\rightarrow \dot{x}_i = -B + \sum_{i=1}^{100} A_{ij}\frac{2x_j^*}{(1+(x_j^*)^2)^2} + \delta A_{ii}\frac{(x_i^*)^2}{(1+(x_i^*)^2)^2}\end{aligned}$$

We set the cost matrix in LQR as

$$\begin{aligned}&\boldsymbol{Q} = 10\boldsymbol{I}_{100\times 100},\\ &R = 0.01\boldsymbol{I}_{100\times 100}.\end{aligned}$$

The obtained Riccati solution $\boldsymbol{S}$ forms the Lyapunov function $V = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{P}_2)^\top \boldsymbol{S}(\boldsymbol{x} - \boldsymbol{P}_2)$, the controller is $\boldsymbol{u} = -\boldsymbol{K}(\boldsymbol{x} - \boldsymbol{P}_2)$ where $\boldsymbol{K} \in \mathbb{R}^{100\times 100}$ is returned by the lqr solver. The Lie derivative of the Lyapunov function is $-(\boldsymbol{x} - \boldsymbol{P}_2)^\top \boldsymbol{Q}_1(\boldsymbol{x} - \boldsymbol{P}_2)$ with $\boldsymbol{Q}_1 = \boldsymbol{Q} + \boldsymbol{K}^\top \boldsymbol{R}\boldsymbol{K}$.

Critic-Actor NNETC. Similarly, we set the hyperparameters as $\alpha_c = \alpha_a = 1e-2$, $e\text{thres} = 0.2$. The actuator in this example is

$$\boldsymbol{g} = \text{diag}(\frac{x_1^2}{(1+x_1^2)^2}, \cdots, \frac{x_{100}^2}{(1+x_{100}^2)^2}). \tag{16}$$

Since the dynamics of $\boldsymbol{W}_c$ and $\boldsymbol{W}_a$ are both 100^2-D, leading to a significantly high dimensional system, we reduce the dynamics as

$$\begin{aligned}&V^*(\boldsymbol{x}) = \boldsymbol{W}_c^\top (x_1^2, \cdots, x_{100}^2)^\top,\ \boldsymbol{W}_c \in \mathbb{R}^{100},\\ &\boldsymbol{u}^*(\boldsymbol{x}) = \text{diag}(\boldsymbol{W}_a)(x_1, \cdots, x_{100})^\top,\ \boldsymbol{W}_a \in \mathbb{R}^{100},\end{aligned}$$

to the 100-D systems, for the sake of limited computational resources.

IRL ETC. We set the hyperparameters as $\beta = 1e-2$, $\alpha = 0.1$, $\sigma(\cdot) = \text{Id}(\cdot)$, $\eta = 1$, $\lambda_x = 0.1$, $\lambda_y^2 = 0.6$. The actuator in this example is

$$\boldsymbol{g} = \text{diag}(\frac{x_1^2}{(1+x_1^2)^2}, \cdots, \frac{x_{100}^2}{(1+x_{100}^2)^2}). \tag{17}$$

We reduce the dimension of the dynamics as the same with that in Critic-Actor NNETC above.

Test configurations. We select the same event functions as those for GRN to implement the event-triggered control. For the initial value, we set $\boldsymbol{x}_0 = \boldsymbol{P}_1 + \boldsymbol{\xi}_i$, $\boldsymbol{\xi}_i \sim \mathcal{U}[-0.5, 0.5]$, $i = 1, c \ldots, 5$, and the random seeds are set as $\{0, 3, 4, 5, 6\}$.

A.3.5 Motivation of selecting the benchmark systems

In [36], a geometrical approach for switching the system from ROA of one equilibrium to another, through finite changes of the experimentally feasible parameters, wherein GRN system is investigated in their paper. Since our Neural ETC has similarity to the geometrical approach in terms of adding finite non-invasive control to the system, we also study GRN in our work. The Lorenz system is a classic chaotic systems possessing plentiful shapes of dynamical trajectories, hence, the control of Lorenz (or control of chaos in a more common sense) is of important position in control literature [41, 42], and the control of Lorenz system under event-triggered implementation is also investigated in [43]. In [30], a topological reconstruction method to the structure of complex dynamics is proposed to revive the degenerate complex system via minimal interventions, i.e., reconstructing links or nodes as small as possible, and the Michaelis–Menten model describing the evolution dynamics of sub-cellular behavior is considered as an illustration. Since the event-triggered control aims at adding feasible control to the complex system intermittently, e.g., changing the network structure slowly in time, we think it's meaningful to consider the Michaelis–Menten model in our work to see if there are essentially same parts between our method with the topological reconstruction method.

A.4 More Related Works

Neural control with certificate functions. Previous works in neural control establish the performance guarantee via using the certificate functions, including Lyapunov function for stability [44, 4], barrier function for safety [6, 45, 46, 47, 48, 49], and contraction metrics for stability in trajectory tracking [50, 3]. However, all these feedback controllers require impractically high communication cost for updating the controller continuously when deployed on the digital platforms. We solve this challenge in limited communication resources and improve the performance guarantee at the same time.

Event-triggered control. The pioneering works [21, 13] highlighted the advantages of event-based control against the periodic implementation in reducing the communication cost. Since then, [14] investigates the sufficient conditions for avoiding the Zeno behavior in event-triggered implementations of stabilizing feedback control laws, [16] extends the event-triggered control to the linear stochastic system and [15] gives the system theory of event-triggered control scheme for perturbed linear systems. Machine learning methods have also been introduced to the ETC settings, [32, 33] employ the critic-actor RL structure to solve the dynamic Hamilton-Jacobi-Bellman equation under the ETC, [37] cultivates a model-free hierarchical RL method to optimize both the control and communication policies for discrete dynamics, and [38] applies deep RL to ETC in the nonlinear systems. All the previous works focus on the stabilization analysis of the controlled systems, the existence of the minimal inter-event time (and hence avoids the Zeno behavior), and directly introducing machine learning methods to ETC. To our knowledge, we are the first to study the optimization scheduling problem of ETC in the continuous dynamics.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   The novel method we propose is summarized in Section 3, and the experimental results are in Section 5.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in Section 7 in the main text.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: See Section A.1 in Appendix for all the theoretical assumptions and proofs.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Please see Section A.3 for detailed information of training process, the corresponding code is also provided in the supplementary material.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our source code and data is released at `https://anonymous.4open.science/r/Neural-Event-triggered-Control-628A/README.md`.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Refer to Section A.2 for training and test details.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The main results in Table 1 and Fig. 3 are averaged over different samples, and the corresponding standard errors are reported.

   Guidelines:

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

See Section A.2 in Appendix.

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work conforms with the NeurIPS Code of Ethics.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work addresses the optimization schedule problem in the event-triggered control field, hence, the impact of the work is limited to the technical aspect.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work does not pose such risks.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited all the existing assets used in our code.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide all the documentation in the attached code repository.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.