



Title	Virtual Dynamics for Systems with Fixed Controllers through Signal Shaping
Author(s)	田中, 健太
Citation	大阪大学, 2026, 博士論文
Version Type	VoR
URL	https://doi.org/10.18910/105437
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

Doctoral Dissertation

Virtual Dynamics for Systems
with Fixed Controllers through Signal Shaping

Kenta Tanaka

January 2026

Graduate School of Engineering,
The University of Osaka

Abstract

This dissertation examines control systems in which the controller is fixed. Controllers are frequently fixed in practical contexts, including certified and safety-critical systems where stability and performance must be maintained, or when humans act as controllers within the loop. Consequently, this problem setting is of considerable practical relevance.

Motivated by the above, this dissertation investigates an approach that introduces systems designed to shape information, termed *Alterers*. The series interconnection of an Alterer, the plant, and another Alterer is treated as a new plant, designated as *Virtual Dynamics*. The objective is to enhance overall control system performance by constructing Virtual Dynamics that the fixed controller handles more effectively, utilizing Alterers to obscure problematic plant characteristics. This dissertation addresses three central questions: (i) which types of dynamics are more effectively managed by the fixed controller, (ii) which Virtual Dynamics arise from specific Alterers, and (iii) whether control performance is improved through Virtual Dynamics.

Chapter 3 positions dynamic quantizers for discrete-valued input control as Alterers that conceal the nonlinear characteristics of discrete-valued input constraints. The Virtual Dynamics design problem is formulated as minimizing deviation from an ideal continuous-valued system, subject to implementability constraints including stability, gain, and order. An open-source Python library, NQLib, has been developed to integrate the design and evaluation of dynamic quantizers. NQLib is used to demonstrate a procedure for designing Virtual Dynamics and to validate its effectiveness.

Chapter 4 introduces an Alterer that presents the agent with altered states of the plant, aiming to improve learning efficiency and final performance. The Virtual Dynamics design problem is formulated as finding an Alterer that minimizes the time required to achieve a target score. Numerical experiments on inverted-pendulum stabilization tasks demonstrate that prediction-type Alterers accelerate learning and improve final control performance. Additionally, experiments on second-order linear systems indicate that replacing an unstable zero with a stable zero through prediction enhances learning efficiency. These findings suggest that minimum-phase characteristics provide useful design guidelines.

Chapter 5 introduces an Alterer that alters and presents visual information in human-machine systems, aiming to improve operability. The Virtual Dynamics design problem is formulated as the problem of finding an Alterer that maximizes task performance. A simulator, The Inverted Pendulum Game, is developed, enabling information presentation via Alterers for human stabilization tasks. Experiments with 32 subjects compare a mode

in which the Alterer presents a prediction of the inverted pendulum state 0.6 seconds ahead with a mode presenting the actual state. Results indicate that the median score in the future-prediction mode exceeds that in the real-presentation mode by more than a factor of two for all subjects, with reductions in both input and angle costs. To further understand the impact of this operability improvement, an analysis is conducted using a linearized model of The Inverted Pendulum Game and a human. This analysis demonstrates that prediction-based Alterers can eliminate the need for velocity and angular velocity information for system stabilization, providing guidance for Alterer design.

In summary, this dissertation presents a cross-domain framework for Alterer-based Virtual Dynamics design to enhance performance without modifying a fixed controller, validated through case studies in quantized control, reinforcement learning, and human operation.

Acknowledgements

I beg your indulgence to write this section in Japanese, as most of the people who supported this work are Japanese.

本論文は、著者が大阪大学工学部応用理工学科機械工学コースと大阪大学大学院工学研究科機械工学専攻博士前/後期課程に在籍した間に従事した研究を集成したものです。本論文の執筆にあたり、石川 将人 教授、南 裕樹 教授（兵庫県立大学）、増田 容一 助教、鈴木 朱羅 助教（東北大学）には多大なるご指導とご助言を賜りました。特に、6年間にわたり研究チームのリーダーとして、研究の基礎からスライドの作り方まで丁寧にご指導くださった南 裕樹 教授には、心より感謝申し上げます。私には至らない点が多々あったかと存じますが、最後まで諦めずにご指導くださったおかげで、無事に博士論文を完成させることができました。ありがとうございました。また、石川 将人 教授には、研究会をはじめとする様々な場面で、研究の方向性や論文執筆に関する的確なアドバイスを頂きました。さらに、より近い目線ながらも頼れる存在であった増田 容一 助教と鈴木 朱羅 助教には、多くの励ましや具体的なアドバイスを頂きました。

お忙しい中、本論文の審査をお引き受けくださった井野 秀一 教授、東森 充 教授、石川 将人 教授に深く感謝申し上げます。井野 秀一 教授には、研究の将来の方向性について、鋭いご指摘と建設的なご助言を頂きました。東森 充 教授には、研究内容に関する詳細なご質問を頂き、より深く考える契機となりました。

本研究を進めるにあたり、周りの方々の温かいご支援とご協力は、欠かせないものでした。研究室の先輩方には、研究に関する様々なアドバイスや激励を頂きました。研究室の同期には、研究活動に大きな刺激を与えてくれたとともに、楽しい時間を共有してくれました。研究室の後輩には、スマブラや飲み会などの交流を通じて、若さと元気を分けてくれました。楽器という趣味を共有する友人たちは、リフレッシュの場を提供しながら、研究を続ける原動力を与えてくれました。また、家族や親戚は、長く学生生活続ける私を快く理解し、支えてくれました。

研究室の先生方、先輩方、同期、後輩、友人、家族や親戚に感謝の意を表します。

最後に、本研究に対してご支援を頂いた JST 次世代挑戦的研究者育成プロジェクト様、公益財団法人 立石科学技術振興財団様に感謝を申し上げます。

Contents

Chapter 1	Introduction	3
1.1	Constraints on Controller Frameworks in Control Systems	3
1.2	Prior Approaches Using Add-on Alterers	3
1.3	Virtual Dynamics Favorable for Controllers	4
1.4	Contributions and Organization of This Dissertation	6
1.4.1	Contributions	6
1.4.2	Organization of This Dissertation	8
Chapter 2	Design Problem of Virtual Dynamics	9
2.1	Designing Virtual Dynamics to Enhance Ease of Control	9
2.2	Subproblems in Subsequent Chapters	10
Chapter 3	Virtual Dynamics Design for Discrete-Valued Input Control	12
3.1	Design Problem	12
3.1.1	Discrete-Valued Input Control	12
3.1.2	Dynamic Quantizer and Its Design Problem	15
3.2	Virtual Dynamics Design Methods	18
3.2.1	Analysis of the Evaluation Function and Constraints	19
3.2.2	Optimal Dynamic Quantizer	19
3.2.3	Relaxing to a Linear Programming Problem	21
3.2.4	Nonlinear Optimization for SISO Systems	23
3.3	Dynamic Quantizer Design Toolbox NQLib	24
3.3.1	Overview of NQLib	24
3.3.2	Workflow Using NQLib	25
3.3.3	Illustrative Design Example	26
3.4	Conclusion	33
Chapter 4	Virtual Dynamics Design for Systems with Reinforcement Learning	34
4.1	Design Problem	35

4.2	Preliminary Example: Improving Learning Efficiency via Virtual Dynamics	36
4.2.1	Plant: Cart-Type Inverted Pendulum	37
4.2.2	Alterer: Prediction	38
4.2.3	Controller: Reinforcement Learning with Deep Q Network . .	38
4.2.4	Numerical Experiments and Results	39
4.2.5	Discussion	40
4.3	Design Guidelines via Minimum-Phase Considerations	40
4.3.1	Plant: Second-Order Linear System	42
4.3.2	Alterer: Prediction	43
4.3.3	Controller: Reinforcement Learning with Proximal Policy Optimization	44
4.3.4	Numerical Experiments and Results	44
4.3.5	Discussion	45
4.4	Conclusion	46
Chapter 5	Virtual Dynamics Design for Human-Machine Systems	47
5.1	Design Problem	47
5.2	The Inverted Pendulum Game: A Pioneering Example of Operability Enhancement via Virtual Dynamics	49
5.2.1	The Inverted Pendulum Game	49
5.2.2	Alteration: Prediction Based on the Current State	50
5.2.3	Experimental Method	51
5.2.4	Experimental Results: Example of Time Evolution	54
5.2.5	Experimental Results: Comparison and Analysis of Scores . .	54
5.2.6	Experimental Results: Comparison of Input Cost	55
5.2.7	Experimental Results: Comparison of Angle Cost	57
5.2.8	Questionnaire Results	58
5.3	Design Guidelines for Virtual Dynamics to Enhance Human-Machine Operability	59
5.3.1	Alterer: Linear Future Prediction	59
5.3.2	Models of the Inverted Pendulum and the Information Alterer	60
5.3.3	Controller: Static Output Feedback	60
5.3.4	Numerical Example	61
5.4	Conclusion	63
Chapter 6	Conclusion	64

A	Appendix A	66
A.1	Documentation of NQLib	66
A.1.1	What Can I Do with NQLib?	66
A.1.2	Documentation	66
A.1.3	References	66
A.1.4	License	67
A.2	Classes	67
A.2.1	nqlib.Controller	67
	Example	67
	Methods	68
A.2.2	nqlib.DynamicQuantizer	68
	Example	68
	Properties	68
	Methods	69
	Methods	69
	Properties	74
A.2.3	nqlib.Plant	76
	Methods	76
A.2.4	nqlib.StaticQuantizer	76
	Properties	76
	Methods	76
A.2.5	nqlib.System	78
	Methods	78
A.3	Functions	81
A.3.1	nqlib.installed_solvers	81
A.3.2	nqlib.order_reduced	81
	Bibliography	83

List of Figures

1.1	Conceptual diagram of the control system with an inserted Alterer targeted in this study	4
1.2	Block diagram illustrating the concept of considering Alterers combined with the controller as a new controller	5
1.3	Block diagram illustrating the concept of considering Alterers combined with the plant as a new plant (Virtual Dynamics)	5
2.1	Block diagram of a control system with Alterers forming Virtual Dynamics	9
2.2	Systems considered in this dissertation	10
3.1	Discrete-valued input control system	13
3.2	Simulation results of a control system with and without discrete-valued input constraints	14
3.3	Mid-tread-type static quantizer	15
3.4	Discrete-valued input control system with a dynamic quantizer inserted before the plant	16
3.5	Parallel connection of the discrete-valued input control system $\mathcal{S}_D$ with a dynamic quantizer and the continuous-valued input control system $\mathcal{S}_I$. . .	17
3.6	Generalized system with a dynamic quantizer	18
3.7	Simulation result of a control system with the dynamic quantizer designed by the algebraic method described in Section 3.2.2	29
3.8	Simulation result of a control system with the dynamic quantizer designed by the LP-based method described in Section 3.2.3	30
3.9	Simulation result of a control system with the dynamic quantizer designed by the LP-based method with order reduction described in Section 3.2.3 . .	31
3.10	Simulation result of a control system with the dynamic quantizer designed by the gradient-based method described in Section 3.2.4	32
4.1	Block diagram of a control system where the RL agent serves as the controller	35

4.2	Block diagram of the control system where the DQN agent serving as the controller receives altered information from the Alterer	36
4.3	Cart-type inverted pendulum	37
4.4	Score evolution over 20 trials of learning inverted-pendulum stabilization, plotted with minimum, 25th percentile, median, 75th percentile, and maximum values	41
4.5	Block diagram of the control system where the PPO agent serving as the controller receives altered information from the Alterer	42
5.1	Block diagram of a control system where a human operator serves as the controller	48
5.2	Game screen of The Inverted Pendulum Game	50
5.3	Illustration of state alteration in The Inverted Pendulum Game	51
5.4	Photo of an experiment using The Inverted Pendulum Game	52
5.5	Time evolution of input, position, and angle in a performance in which one subject is shown real visual information	54
5.6	Time evolution of input, position, and angle in a performance in which the same subject as in Fig. 5.5 is shown altered visual information	55
5.7	Comparison of scores when real and altered information was presented	56
5.8	Comparison of input costs when presented with real versus altered information	57
5.9	Comparison of angle costs when presented with real versus altered information	58
5.10	Block diagram of the Inverted Pendulum Game model to investigate factors for operability improvement	60
5.11	Stability of the eigenvalues for each T and k_θ when $K = [0.01 \ k_\theta]$	62
5.12	Stability of $\tilde{A}$ for each T and k_θ	62

Notation and definitions

The following notation is used in this dissertation.

$\mathbb{R}$	Set of all real numbers
$\mathbb{R}_{>0}$	Set of all positive real numbers
$\mathbb{R}_{\geq 0}$	Set of all non-negative real numbers
$\mathbb{C}$	Set of all complex numbers
$\mathbb{N}$	Set of all natural numbers
$\mathbb{Z}_{\geq 0}$	Set of all non-negative integers
$\mathbb{S}$	Set of all systems
$\in$	Element of
$\subset$	Strict subset of
$\subseteq$	Subset of (strict or not)
$\cup$	Union
$M^\top$	Transpose of the matrix M
$M^\dagger$	Moore-Penrose pseudoinverse of M
M^{-1}	Inverse of the matrix M
$\lfloor x \rfloor$	Floor function

The following notation is also defined.

- For a quantization width $d \in \mathbb{R}_{>0}$, the set of quantization steps is denoted as $\mathbb{V}_d := \{0, \pm d, \pm 2d, \dots\}$.
- For a vector $x \in \mathbb{R}^n$, its i -th ($1 \leq i \leq n$) element is denoted as x_i .
- For a set of systems $\mathbb{X} \subseteq \mathbb{S}$, the set of all static systems in $\mathbb{X}$ is denoted by $\mathbb{X}_{\text{static}}$.
- For systems $\mathcal{S}_1, \mathcal{S}_2 \in \mathbb{S}$, their series connection (i.e., the system whose input is that of $\mathcal{S}_1$ and output is that of $\mathcal{S}_2$, and the input of $\mathcal{S}_2$ is the output of $\mathcal{S}_1$) is denoted by $\mathcal{S}_2 \circ \mathcal{S}_1$.
- For matrices A, B, C , and D with appropriate dimensions, (A, B, C, D) denotes a

linear system represented in state-space form as

$$\begin{cases} x[k+1] = Ax[k] + Bu[k], \\ y[k] = Cx[k] + Du[k], \end{cases}$$

where k is the discrete-time index and $x[k]$, $u[k]$, and $y[k]$ are the state, input, and output signals with appropriate dimensions, respectively.

- $0_{n \times m}$ denotes the zero matrix of size $n \times m$.
- I_n denotes the identity matrix of size $n \times n$.
- For a matrix M , a vector v , and a vector signal x , $\|M\|$, $\|v\|$, and $\|x\|$ denote their infinity norms, respectively.

Chapter 1

Introduction

1.1 Constraints on Controller Frameworks in Control Systems

Control engineering is a discipline that provides means for constructing control systems to guide various objects to desirable states. Through long years of development, theories for mathematically analyzing and designing diverse control systems have been established, including transfer functions, state-space models, optimal control, robust control, and adaptive control. The design of control systems using the above techniques typically involves designing a controller for a given plant. However, in practical control system design, the controller is often fixed. For example, when controlling commercialized systems, the controller is often already determined and cannot be changed. Such systems include motor test machines [1]. Furthermore, there are cases where the controller should not be changed to preserve certain properties. For example, there is a demand to avoid destabilization due to redesign [2,3] or to utilize the performance of existing controllers [4,5]. Also, systems that include a human as a controller are one example where the controller is fixed. Such systems are common in the real world, including automobiles [6, 7], airplanes [8–10], drones [11, 12], heavy machinery [13], and remote surgical systems [14]. These facts indicate that not only methods for designing controllers but also methods for improving the overall performance of control systems that can be applied even when the controller is fixed are important.

1.2 Prior Approaches Using Add-on Alterers

An approach to improving performance in control systems with a fixed controller involves inserting a system that shapes signals before and after the controller, as shown in Fig. 1.1, and

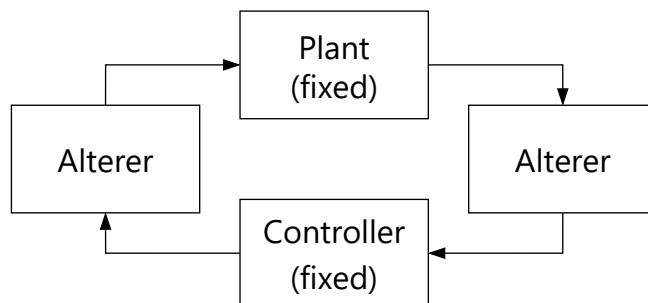


Fig. 1.1: Conceptual diagram of the control system with an inserted Alterer targeted in this study

designing it. These signal-shaping systems are referred to as *Alterers* in this dissertation. This study focuses on this approach. Previous research on Alterers includes the following.

First, we discuss Alterers inserted after the controller to shape control inputs. Control barrier functions [15, 16] are methods that process control inputs generated by the controller to guarantee safety and stability. Random dither quantizers [17] are methods that intentionally add noise to control inputs before quantization for plants that only accept discrete-valued inputs, thereby reducing the impact of quantization errors on the plant.

Next, we discuss Alterers inserted before the controller to shape inputs to the controller. First, in networked control systems, such as remote control systems, whether manual or automatic, Alterers are sometimes inserted to compensate for communication delays or data rate constraints [6]. Additionally, there are methods to shape commands to the plant to reduce unwanted vibrations in the system [18]. Furthermore, displays that show bullet trajectory predictions to the human operating fighter jets [8, 9] can also be considered examples of Alterers that shape inputs to the controller. Also, disturbance observers [19, 20] are Alterers that estimate disturbances based on outputs from the plant and correct inputs to the controller.

There are also methods that insert Alterers both before and after the plant. For example, exact linearization of nonlinear control systems [21, 22] is a method that linearizes the plant dynamics as seen from the controller by inserting nonlinear transformers before and after the plant.

1.3 Virtual Dynamics Favorable for Controllers

Previous research on Alterers has primarily focused on the controller (as shown in Fig. 1.2), with Alterers designed to add advanced functions to it. For example, control barrier functions can be viewed as adding functions to the controller that guarantee the state of the plant remains within a certain region. Also, Alterers that predict trajectories or perform noise reduction can

1.3. VIRTUAL DYNAMICS FAVORABLE FOR CONTROLLERS

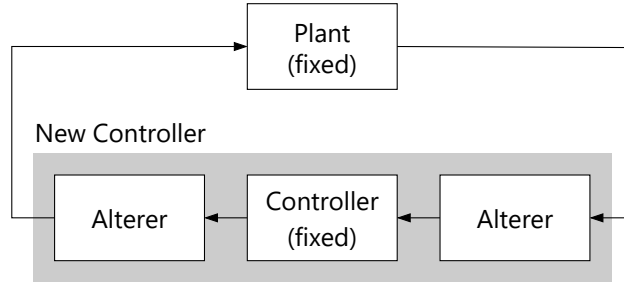


Fig. 1.2: Block diagram illustrating the concept of considering Alterers combined with the controller as a new controller

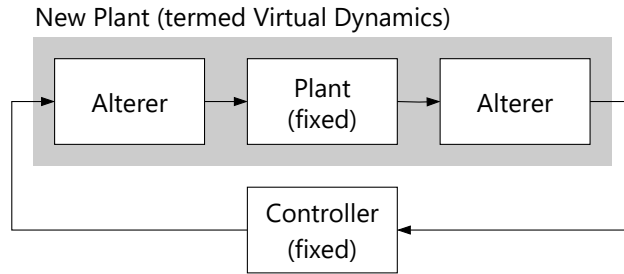


Fig. 1.3: Block diagram illustrating the concept of considering Alterers combined with the plant as a new plant (Virtual Dynamics)

be said to add functions that enable the controller or the human to more accurately grasp the state of the plant. The design of Alterers from this perspective can be considered a policy that treats the series connection of an Alterer, the controller, and another Alterer as a new controller, as shown in Fig. 1.2, thereby enabling effective plant control. This policy can naturally extend the controller and has the advantage of being easy to integrate with the functions of the existing controller.

In contrast, this study takes the approach of designing Alterers by treating the series connection of an Alterer, a plant, and another Alterer as a new plant, as shown in the shaded area of Fig. 1.3. Specifically, we consider the following. First, the series connection of Alterer, plant, and Alterer, as shown in the shaded area of Fig. 1.3, has dynamics different from those of the original plant. Since this dynamics differs from the original dynamics, even though the plant itself is not modified, we refer to this dynamics as *Virtual Dynamics*. The plant itself sometimes contains characteristics that hinder the controller or its design. For example, when designing a controller using linear control theory, the nonlinearity of the plant may hinder the analysis of the control system and the design of the controller [23]. Also, when a human or machine learning operates a machine, it is known that delays or unstable zeros in the machine adversely affect operability [24, 25]. If the hindering characteristics of the plant for

1.4. CONTRIBUTIONS AND ORGANIZATION OF THIS DISSERTATION

the controller or its designer are not included in the Virtual Dynamics, there is a possibility of improving the overall performance of the control system. This study aims to improve the overall performance of the control system by designing Alterers that prevent the hindering properties of the plant from appearing in the Virtual Dynamics.

Considering Virtual Dynamics is significant for two reasons. The first is that design guidelines for Alterers that cannot be obtained from the perspective of adding functions to the controller may be obtained. For example, Alterers that deceive the controller in a certain way to hide characteristics that hinder control of the plant may be effective. Such Alterers are difficult to design from the perspective of adding new functions to the controller. The second is that Virtual Dynamics can be analyzed even if the controller contains black-box elements. For example, controllers for systems that interact with complex environments, such as automobiles, currently require elements such as humans and machine learning. These controllers are black-box-like and are difficult to analyze mathematically. The design policy using Virtual Dynamics can be applied to such control systems because the controller is not included in the Virtual Dynamics.

There are two points to consider for this approach. The first is: which Alterer is more favorable for controller design or for the controller itself? The second is: how to design Alterers to realize such Virtual Dynamics? In this dissertation, we aim to construct a design theory for the control performance improvement approach using Virtual Dynamics while considering these two points.

1.4 Contributions and Organization of This Dissertation

1.4.1 Contributions

This dissertation focuses on an approach to designing control systems in which the controller is fixed by inserting Alterers before and after the plant, and by designing Alterers so that the Virtual Dynamics are favorable to the controller. Specifically, the following three contributions are made.

The first is demonstrating the effectiveness of Virtual Dynamics for a class called discrete-valued input systems. Discrete-valued input systems are control systems in which the inputs to the plant are constrained to discrete values. Discrete-valued input systems often appear in the real world, such as networked control systems with limited communication capacity and control systems with ON/OFF actuators. Therefore, control system design methods for discrete-valued input systems are important and have been actively researched. In this study, we focus on dynamic quantizers [26–34] as an approach to control system design for discrete-valued input systems. Dynamic quantizers are Alterers that can transform discrete-valued

1.4. CONTRIBUTIONS AND ORGANIZATION OF THIS DISSERTATION

input systems, which are nonlinear systems, into Virtual Dynamics close to linear systems. In this study, we have developed a library, NQLib, for designing dynamic quantizers and demonstrated its effectiveness.

The second is demonstrating the effectiveness of Virtual Dynamics for control systems that use deep reinforcement learning (hereafter, simply RL) as a controller, and presenting candidates for design guidelines for Alterers. RL is a machine learning method that learns actions based on rewards. It is a machine learning method that has attracted attention across various fields, following the success of AlphaGo [35]. In control engineering, research using RL as a controller has been actively conducted, and high performance has been reported [36]. On the other hand, RL requires a large amount of trial data for learning, which consumes significant time and computational resources [37]. To solve this problem, this study aims to improve the efficiency and performance of RL by realizing Virtual Dynamics that are easy to learn through the design of Alterers. Specifically, we presented an example in which learning is made more efficient through Alterers applied to plant outputs. This example is based on the stabilization control problem of an inverted pendulum, a benchmark problem for RL that serves as a foundation for research on Virtual Dynamics for RL. Furthermore, to develop design guidelines for Alterers, we conducted experiments to identify which Virtual Dynamics improve learning efficiency. This experiment used a second-order linear system as the plant, enabling easy parameter analysis. As a result, it was shown that learning efficiency can be improved by removing the unstable zeros of the system using Virtual Dynamics.

The third is demonstrating the effectiveness of Virtual Dynamics for control systems that include a human as a controller, as well as presenting candidates for design guidelines for Alterers. Control systems in which a human is a controller are very common in the real world, including vehicle operation and remote surgical robots. Humans can make appropriate judgments even in complex environments, but mistakes in operation can occur due to fatigue, lack of skills, cognitive load, etc. [38]. Since such mistakes can lead to serious accidents, such as loss of life, improving operability, including the prevention of mistakes, is important. In this study, we aimed to improve operability by realizing Virtual Dynamics that are easy for humans to operate through the design of Alterers. Specifically, we presented an example of how operability can be improved through Alterers for plant outputs, using the inverted pendulum as the subject. The stabilization problem of the inverted pendulum is simple enough to analyze, is often used in control engineering, and is important due to its relation to human walking models. Therefore, this result provides a foundation for research on Virtual Dynamics across all human-machine systems. Furthermore, we conducted numerical analyses to determine which Virtual Dynamics improve operability in order to obtain design guidelines for Alterers. This numerical analysis also used the inverted pendulum as the subject. As a result,

1.4. CONTRIBUTIONS AND ORGANIZATION OF THIS DISSERTATION

it was shown that if stabilization can be achieved even with a simple static output feedback controller, operability for humans may also be improved.

1.4.2 Organization of This Dissertation

The organization of this dissertation is as follows. In Chapter 2, we describe the problem setting of Virtual Dynamics and formulate the design problem. In Chapter 3, we discuss Virtual Dynamics for discrete-valued input systems. In Chapter 4, we discuss Virtual Dynamics for control systems that include RL as a controller. In Chapter 5, we discuss Virtual Dynamics for control systems that include a human as a controller. Finally, in Chapter 6, we summarize this dissertation and discuss future work.

Chapter 2

Design Problem of Virtual Dynamics

In this chapter, we describe the problem setting addressed in this dissertation. To this end, we define the systems and evaluation functions considered in this dissertation. Furthermore, we explain how this problem setting is treated in each chapter.

2.1 Designing Virtual Dynamics to Enhance Ease of Control

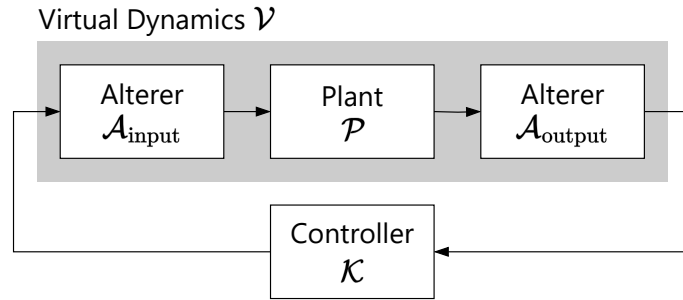


Fig. 2.1: Block diagram of a control system with Alterers forming Virtual Dynamics

In this study, we consider a system shown in Fig. 2.1. The system consists of a plant $\mathcal{P} \in \mathbb{S}$ and Alterers $\mathcal{A}_{\text{input}}, \mathcal{A}_{\text{output}} \in \mathbb{S}$ that shape the input and output signals of $\mathcal{P}$, respectively. The series interconnection $\mathcal{V} := \mathcal{A}_{\text{output}} \circ \mathcal{P} \circ \mathcal{A}_{\text{input}}$ behaves, from the controller's viewpoint, as a system different from the original plant $\mathcal{P}$. Since the dynamics are altered indirectly without directly modifying $\mathcal{P}$, we refer to $\mathcal{V}$ as the Virtual Dynamics in this dissertation. The

2.2. SUBPROBLEMS IN SUBSEQUENT CHAPTERS

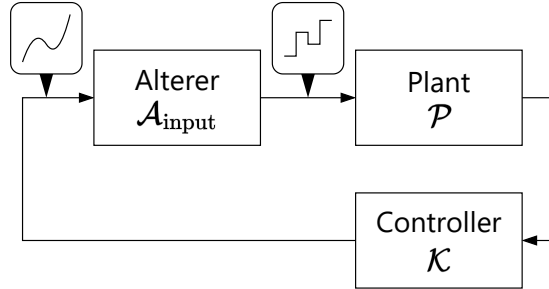
Virtual Dynamics is controlled by a controller $\mathcal{K} \in \mathbb{S}$. If $\mathcal{A}_{\text{input}}$ and $\mathcal{A}_{\text{output}}$ are designed appropriately so that $\mathcal{V}$ does not possess the detrimental characteristics of $\mathcal{P}$ for control, it is expected that the overall performance of the control system can be improved. Here, detrimental characteristics for control include, for example, nonlinearities that cannot be handled by linear control theory or complexities that hinder RL or human control.

In this study, we consider an evaluation function $J_{\mathcal{K}} : \mathbb{S} \rightarrow \mathbb{R}$ representing the ease of control. $J_{\mathcal{K}}(\mathcal{S})$ takes smaller values as the plant $\mathcal{S} \in \mathbb{S}$ becomes easier to control by the controller $\mathcal{K}$. Then, we formulate the problem of finding $\mathcal{A}_{\text{input}}, \mathcal{A}_{\text{output}}$ that improve the ease of control as follows.

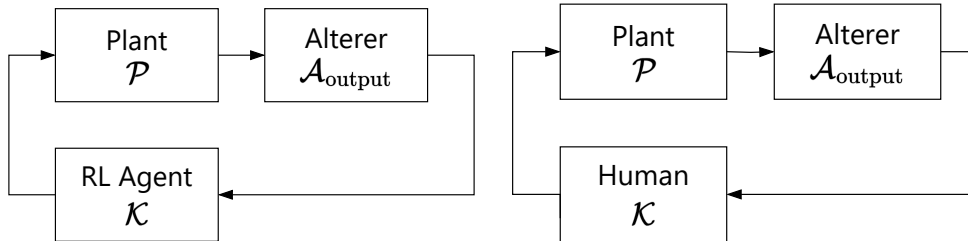
Problem 2.1 Given a plant $\mathcal{P}$ and a controller $\mathcal{K}$. Find Alterers $\mathcal{A}_{\text{input}}$ and $\mathcal{A}_{\text{output}}$ that minimize the evaluation function $J_{\mathcal{K}}(\mathcal{A}_{\text{output}} \circ \mathcal{P} \circ \mathcal{A}_{\text{input}})$ representing the ease of control by $\mathcal{K}$.

2.2 Subproblems in Subsequent Chapters

Specifically, the following chapters present concrete instances of Problem 2.1.



(a) System dealt with in Chapter 3



(b) System dealt with in Chapter 4

(c) System dealt with in Chapter 5

Fig. 2.2: Systems considered in this dissertation

In Chapter 3, we consider Problem 2.1 for the class of control systems known as discrete-

2.2. SUBPROBLEMS IN SUBSEQUENT CHAPTERS

valued input systems, as shown in Fig. 2.2 (a). In such systems, the signal applied to the plant $\mathcal{P}$ is constrained to discrete values. A controller $\mathcal{K}$ designed for a system without discrete-input constraints generally fails to attain its nominal performance when applied to a plant $\mathcal{P}$ with discrete-valued input constraints [26]. Accordingly, in Chapter 3 we choose an evaluation function $J_{\mathcal{K}}$ whose value decreases as the behavior of the ideal system without discrete input constraints becomes closer to that of the actual system $\mathcal{P} \circ \mathcal{A}_{\text{input}}$ with discrete-valued input constraints. By solving Problem 2.1 under this criterion, we design Alterers that realize control performance close to that of a continuous-valued input system even under discrete-valued input constraints.

In Chapter 4, we address Problem 2.1 for control systems employing RL as in Fig. 2.2 (b). RL is a powerful approach that learns to control $\mathcal{P}$ from large amounts of data, but it often requires substantial time, motivating intensive research to improve efficiency [37]. Therefore, we take $J_{\mathcal{K}}$ to be a function whose value decreases as the agent acquires the target score more rapidly. Solving Problem 2.1 with this choice enables the design of Virtual Dynamics that facilitate and accelerate RL.

In Chapter 5, we study Problem 2.1 for human-in-the-loop control systems as in Fig. 2.2 (c). Humans can skillfully operate many systems. Indeed, countless systems such as automobiles, bicycles, aircraft, and heavy machinery are operated by humans. However, humans have limitations in reaction time and processing capacity, and are prone to errors due to fatigue or inattention [38]. Therefore, the goal is to design Alterers that induce Virtual Dynamics, making the operation simpler and less error-prone for the human. Accordingly, we consider Problem 2.1 with an evaluation function $J_{\mathcal{K}}$ whose value decreases as human task performance improves.

Chapter 3

Virtual Dynamics Design for Discrete-Valued Input Control

In this chapter, we consider discrete-valued input control systems in which the control input to the plant is constrained to discrete values. In discrete-valued input control systems, their non-linearity often leads to performance degradation when classical linear control design methods are applied directly. To mitigate this performance degradation, we aim to solve the design problem stated in Problem 2.1.

Section 3.1 first explains the performance degradation caused by discrete-valued input constraints, then formulates the design problem of Virtual Dynamics and Alterer to address it. Section 3.2 then describes the design methods for dynamic quantizers, which are Alterers proposed in previous studies. Finally, Section 3.3 introduces NQLib, a dynamic quantizer design toolbox developed by the author.

3.1 Design Problem

3.1.1 Discrete-Valued Input Control

In this chapter, we consider a discrete-valued input control system shown in Fig. 3.1. In Fig. 3.1, the plant $\mathcal{P} \in \mathbb{S}$ is given by

$$\mathcal{P} : \begin{cases} x_{\mathcal{P}}[k+1] = A_{\mathcal{P}}x_{\mathcal{P}}[k] + B_{\mathcal{P}}v[k], \\ z[k] = C_{\mathcal{P}1}x_{\mathcal{P}}[k], \\ y[k] = C_{\mathcal{P}2}x_{\mathcal{P}}[k], \end{cases} \quad (3.1)$$

3.1. DESIGN PROBLEM

and the controller $\mathcal{K} \in \mathbb{S}$ is given by

$$\mathcal{K} : \begin{cases} x_{\mathcal{K}}[k+1] = A_{\mathcal{K}}x_{\mathcal{K}}[k] + B_{\mathcal{K}1}r[k] + B_{\mathcal{K}2}y[k], \\ u[k] = C_{\mathcal{K}}x_{\mathcal{K}}[k] + D_{\mathcal{K}1}r[k] + D_{\mathcal{K}2}y[k]. \end{cases} \quad (3.2)$$

Here, $k \in \mathbb{Z}_{\geq 0}$ is discrete-time, $x_{\mathcal{P}} \in \mathbb{R}^{N_{\mathcal{P}}}$ and $x_{\mathcal{K}} \in \mathbb{R}^{N_{\mathcal{K}}}$ are states, $u \in \mathbb{R}^m$ is the control input, $y \in \mathbb{R}^{N_y}$ is the measured output, $z \in \mathbb{R}^{N_z}$ is the performance output, $r \in \mathbb{R}^{N_r}$ is the reference input, and $A_{\mathcal{P}} \in \mathbb{R}^{N_{\mathcal{P}} \times N_{\mathcal{P}}}$, $B_{\mathcal{P}} \in \mathbb{R}^{N_{\mathcal{P}} \times m}$, $C_{\mathcal{P}1} \in \mathbb{R}^{N_z \times N_{\mathcal{P}}}$, $C_{\mathcal{P}2} \in \mathbb{R}^{N_y \times N_{\mathcal{P}}}$, $A_{\mathcal{K}} \in \mathbb{R}^{N_{\mathcal{K}} \times N_{\mathcal{K}}}$, $B_{\mathcal{K}1} \in \mathbb{R}^{N_{\mathcal{K}} \times N_r}$, $B_{\mathcal{K}2} \in \mathbb{R}^{N_{\mathcal{K}} \times N_y}$, $C_{\mathcal{K}} \in \mathbb{R}^{m \times N_{\mathcal{K}}}$, $D_{\mathcal{K}1} \in \mathbb{R}^{m \times N_r}$, $D_{\mathcal{K}2} \in \mathbb{R}^{m \times N_y}$ are constant matrices. Furthermore, $v \in \mathbb{V}_d^m$ is the control input to the plant $\mathcal{P}$ with discrete-valued constraints and quantization width $d \in \mathbb{R}_{>0}$.

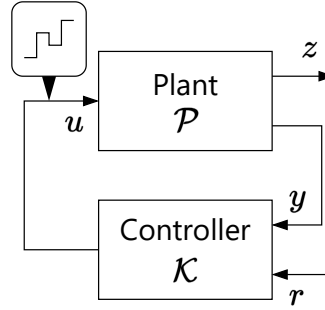


Fig. 3.1: Discrete-valued input control system

Classical linear control design typically assumes that all signals in the loop are continuous-valued. However, there are many practical discrete-valued input systems (Fig. 3.1) where the input signals to the plant are constrained to discrete values. Examples of discrete-valued input systems include control systems with ON/OFF actuators and networked control systems over digital channels with limited data rates.

When the controller $\mathcal{K}$ is linear, it cannot be directly applied to the plant $\mathcal{P}$ due to the discrete-valued input constraints. Therefore, the output u of the controller $\mathcal{K}$ must pass through a system that quantizes it to satisfy the discrete-valued constraints. In discrete-valued input control systems, this quantization introduces nonlinearity, leading to performance degradation. Let us see an example. Figure 3.2 shows the simulation results of the output $z[k]$ of the control system with and without discrete-valued input constraints on v under the following

3.1. DESIGN PROBLEM

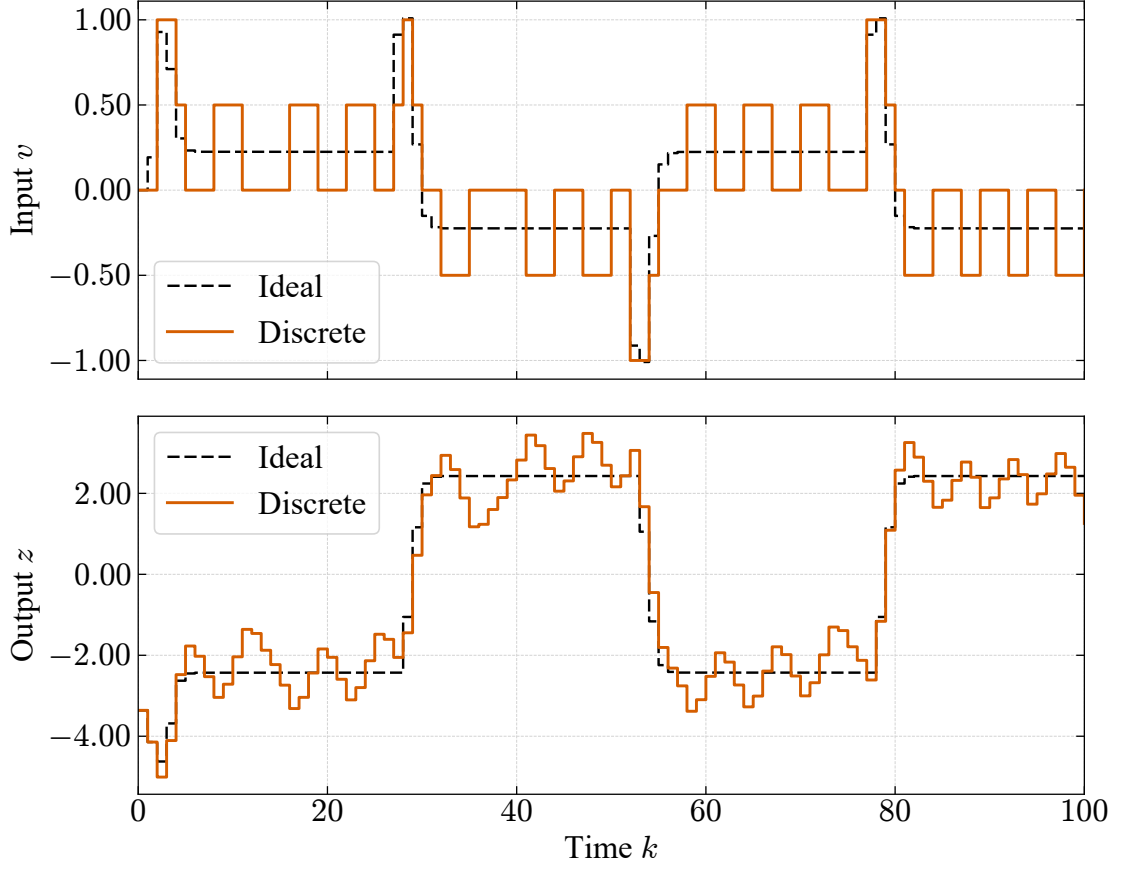


Fig. 3.2: Simulation results of a control system with and without discrete-valued input constraints

parameters.

$$A_{\mathcal{P}} = \begin{bmatrix} 1.19 & 2.22 \\ 0.02 & -0.49 \end{bmatrix}, \quad B_{\mathcal{P}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad C_{\mathcal{P}1} = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad C_{\mathcal{P}2} = \begin{bmatrix} 1 & 1 \end{bmatrix}, \quad (3.3)$$

$$A_{\mathcal{K}} = \begin{bmatrix} 0.833 & 1.95 \\ -0.928 & -1.351 \end{bmatrix}, \quad B_{\mathcal{K}1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad B_{\mathcal{K}2} = \begin{bmatrix} 0.013 \\ 0.604 \end{bmatrix}, \quad (3.4)$$

$$C_{\mathcal{K}} = \begin{bmatrix} -0.344 & -0.257 \end{bmatrix}, \quad D_{\mathcal{K}1} = 0, \quad D_{\mathcal{K}2} = 0, \quad (3.5)$$

$$x_{\mathcal{P}}[0] = \begin{bmatrix} -3.21 \\ -0.15 \end{bmatrix}, \quad x_{\mathcal{K}}[0] = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad (3.6)$$

$$d = 0.5, \quad (3.7)$$

$$r[k] = \begin{cases} 1 & \text{if } \sin(\frac{2\pi k}{50}) \geq 0, \\ -1 & \text{otherwise.} \end{cases} \quad (3.8)$$

3.1. DESIGN PROBLEM

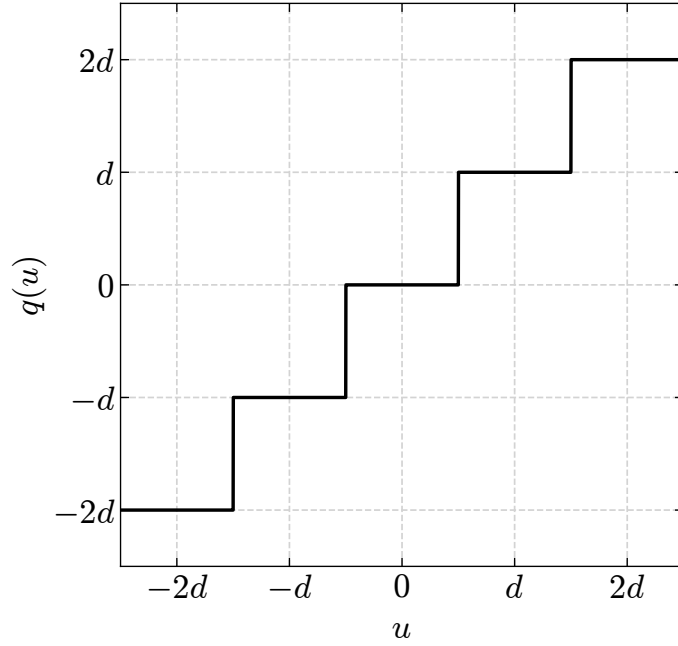


Fig. 3.3: Mid-tread-type static quantizer

Here, when converting the continuous-valued control input u to the discrete-valued v , we used a function $q : \mathbb{R}^m \rightarrow \mathbb{V}_d^m$ defined as

$$q(u) = d \cdot \left\lfloor \frac{u}{d} + \frac{1}{2} \right\rfloor, \quad (3.9)$$

The function q is a rounding function as shown in Fig. 3.3, known as a mid-tread-type quantizer. From Fig. 3.2, it can be seen that when discrete-valued input constraints are present, the output $z[k]$ of the control system exhibits significant oscillations, indicating a marked degradation in control performance. Thus, applying a linear controller to a plant with discrete-valued input constraints via simple quantization results in performance degradation.

3.1.2 Dynamic Quantizer and Its Design Problem

Numerous studies have sought to mitigate the performance degradation of discrete-valued input systems relative to their continuous-valued counterparts. References [39, 40] improve control performance using quantization methods such as pulse-width modulation and Δ - Σ modulation. Reference [41] analyzes and improves control performance using variable quantization width. Reference [17] improves control performance using quantization with random dithering. Furthermore, reference [42–44] proposes a method for directly generating discrete-valued control inputs using model predictive control.

In this chapter, we focus on inserting a dynamic quantizer before the plant as shown in

3.1. DESIGN PROBLEM

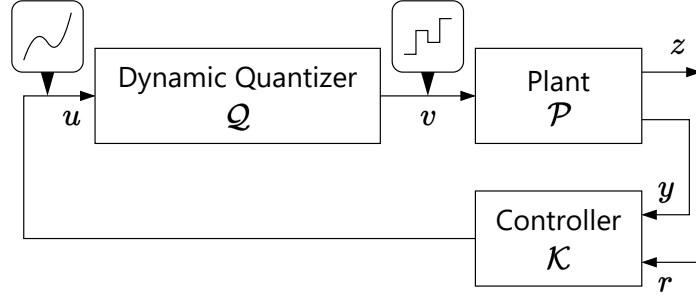


Fig. 3.4: Discrete-valued input control system with a dynamic quantizer inserted before the plant

Fig. 3.4. A dynamic quantizer $Q \in \mathbb{S}$ is given by

$$Q : \begin{cases} x_Q[k+1] = A_Q x_Q[k] + B_Q (v[k] - u[k]), \\ v[k] = q(C_Q x_Q[k] + u[k]). \end{cases} \quad (3.10)$$

A dynamic quantizer is a dynamical system with state x_Q and quantizer q . Here, $x_Q[k] \in \mathbb{R}^{N_Q}$ is the state of the dynamic quantizer with initial condition $x_Q[0] = 0$. Furthermore, $A_Q \in \mathbb{R}^{N_Q \times N_Q}$, $B_Q \in \mathbb{R}^{N_Q \times m}$, and $C_Q \in \mathbb{R}^{m \times N_Q}$ are constant matrices parameterizing Q . The dynamic quantizer converts continuous-valued signals to discrete-valued signals using the information of its state and the difference between its input and output. It is known that with appropriate design of A_Q , B_Q , and C_Q , the behavior of continuous- and discrete-valued input systems can be closely matched, thereby attenuating the nonlinearity introduced by discrete input constraints [45].

Therefore, in this chapter, we consider the following evaluation function J_K in Problem 2.1 that represents the ease of control by K . First, we consider the system shown in Fig. 3.5. This system consists of the ideal continuous-valued input control system $S_I \in \mathbb{S}$ without discrete-valued constraints, and the discrete-valued input control system $S_D \in \mathbb{S}$ with the dynamic quantizer Q inserted, connected in parallel. Using the difference $e := z_D - z_I$ between the outputs z_I of S_I and z_D of S_D , we define the evaluation function J_K as

$$J_K(\mathcal{P} \circ Q) = \sup_{(x_P[0], x_K[0], r) \in \mathbb{R}^{N_P} \times \mathbb{R}^{N_K} \times \ell_\infty^{N_r}} \|e\|_\infty. \quad (3.11)$$

A smaller value of this function indicates a smaller output discrepancy and thus a behavior of S_D closer to that of the ideal S_I .

For the practical implementation of the dynamic quantizer, the following three constraints must be considered. First, it is desirable for the dynamic quantizer to be bounded-input bounded-output (BIBO) stable. Second, it is also necessary to prevent $\|v\|_\infty$ from becoming

3.1. DESIGN PROBLEM

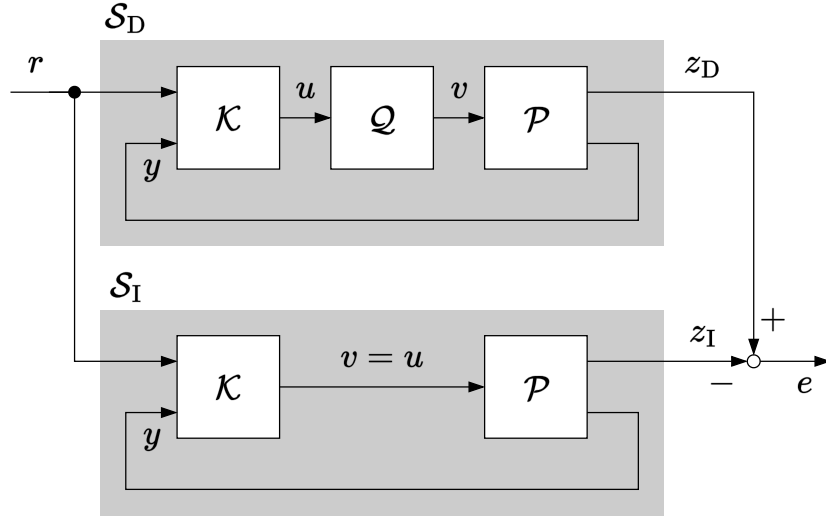


Fig. 3.5: Parallel connection of the discrete-valued input control system $\mathcal{S}_D$ with a dynamic quantizer and the continuous-valued input control system $\mathcal{S}_I$

ing excessively large. To this end, by introducing the quantization noise

$$w := q(C_Q x_Q + u) - (C_Q x_Q + u), \quad (3.12)$$

we define the gain from w to v of Q by

$$\|Q\|_{wv} := \sup_{\substack{w \in \ell_\infty^m \\ \text{s.t. } \|w\|_\infty \neq 0}} \frac{\|v\|_\infty}{\|w\|_\infty}, \quad (3.13)$$

and impose a constraint that this gain be below a certain threshold. Finally, in some cases, it is desirable to keep the order N_Q of the dynamic quantizer small due to implementation constraints.

Based on the above, we formulate the following design problem.

Problem 3.1 Suppose the plant $\mathcal{P} \in \mathbb{S}$ and the controller $\mathcal{K} \in \mathbb{S}$ are given by Eq. (3.1) and Eq. (3.2), respectively. Assume the evaluation function $J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q})$ is defined by Eq. (3.11). Let the maximum gain $\gamma_{wv} \in \mathbb{R}_{>0} \cup \{\infty\}$ and the order of the dynamic

3.2. VIRTUAL DYNAMICS DESIGN METHODS

quantizer $N_{\max} \in \mathbb{N} \cup \{\infty\}$ be given. Then solve the following optimization problem.

$$\underset{\mathcal{Q}}{\text{minimize}} \quad J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q}) \quad (3.14)$$

$$\text{subject to} \quad \mathcal{Q} \text{ is BIBO stable,} \quad (3.15)$$

$$\|\mathcal{Q}\|_{wv} \leq \gamma_{wv}, \quad (3.16)$$

$$N_{\mathcal{Q}} \leq N_{\max}. \quad (3.17)$$

3.2 Virtual Dynamics Design Methods

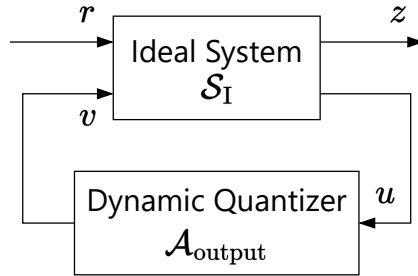


Fig. 3.6: Generalized system with a dynamic quantizer

As shown in Fig. 3.6, previous studies have considered a more generalized ideal system $\mathcal{S}_I$ given by

$$\mathcal{S}_I : \begin{cases} x[k+1] = Ax[k] + B_1r[k] + B_2v[k], \\ z[k] = C_1x[k] + D_1r[k], \\ u[k] = C_2x[k] + D_2r[k], \end{cases} \quad (3.18)$$

Here, $x \in \mathbb{R}^{N_S}$ is the system state, and $A \in \mathbb{R}^{N_S \times N_S}, B_1 \in \mathbb{R}^{N_S \times N_r}, B_2 \in \mathbb{R}^{N_S \times m}, C_1 \in \mathbb{R}^{N_z \times N_S}, D_1 \in \mathbb{R}^{N_z \times N_r}, C_2 \in \mathbb{R}^{N_y \times N_S}, D_2 \in \mathbb{R}^{N_y \times N_r}$ are constant matrices. In the following, for notational simplicity, we use this generalized ideal system $\mathcal{S}_I$. Note that the ideal system $\mathcal{S}_I$ consisting of the plant $\mathcal{P}$ given by Eq. (3.1) and the controller $\mathcal{K}$ given by Eq. (3.2) can be represented by defining it as follows:

$$A := \begin{bmatrix} A_{\mathcal{P}} & 0 \\ B_{\mathcal{K}2}C_{\mathcal{P}2} & A_{\mathcal{K}} \end{bmatrix}, \quad B_1 := \begin{bmatrix} 0 \\ B_{\mathcal{K}1} \end{bmatrix}, \quad B_2 := \begin{bmatrix} B_{\mathcal{P}} \\ 0 \end{bmatrix}, \quad (3.19)$$

$$C_1 := \begin{bmatrix} C_{\mathcal{P}1} & 0 \end{bmatrix}, \quad D_1 := 0, \quad C_2 := \begin{bmatrix} D_{\mathcal{K}2}C_{\mathcal{P}2} & C_{\mathcal{K}} \end{bmatrix}, \quad D_2 := D_{\mathcal{K}1}. \quad (3.20)$$

3.2.1 Analysis of the Evaluation Function and Constraints

In this section, for the dynamic quantizer $\mathcal{Q}$, we summarize the analysis results from previous studies [26–29] regarding the evaluation function $J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q})$ given by Eq. (3.11), the BIBO stability of the dynamic quantizer, and the gain $\|\mathcal{Q}\|_{wv}$.

First, we discuss the evaluation function $J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q})$. We assume that the static quantizer q satisfies

$$\|q(\mu) - \mu\|_{\infty} \leq \frac{d}{2} \quad (\forall \mu \in \mathbb{R}^m). \quad (3.21)$$

This property is satisfied by functions such as the mid-tread quantizer shown in Fig. 3.3. Under the above assumption, the evaluation function $J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q})$ can be derived as follows:

$$J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q}) = \left\| \sum_{j=0}^{\infty} \text{abs}(\bar{C} \bar{A}^j \bar{B}) \right\|_{\infty}, \quad (3.22)$$

$$\bar{A} := \begin{bmatrix} \tilde{A} & B_2 C_{\mathcal{Q}} \\ 0 & A_{\mathcal{Q}} + B_{\mathcal{Q}} C_{\mathcal{Q}} \end{bmatrix}, \quad \tilde{A} := A + B_2 C_2 \quad (3.23)$$

$$\bar{B} := \begin{bmatrix} B_2 \\ B_{\mathcal{Q}} \end{bmatrix}, \quad \bar{C} := \begin{bmatrix} C_1 & 0 \end{bmatrix}. \quad (3.24)$$

Next, we discuss the BIBO stability of the dynamic quantizer $\mathcal{Q}$. Using the quantization noise defined in Eq. (3.12), Eq. (3.10) can be rewritten as

$$\mathcal{Q} : \begin{cases} x_{\mathcal{Q}}[k+1] = (A_{\mathcal{Q}} + B_{\mathcal{Q}} C_{\mathcal{Q}}) x_{\mathcal{Q}}[k] + B_{\mathcal{Q}} w[k], \\ v[k] = C_{\mathcal{Q}} x_{\mathcal{Q}}[k] + u[k] + w[k]. \end{cases} \quad (3.25)$$

From this, it follows that the BIBO stability of the dynamic quantizer is equivalent to the condition that the poles of the matrix $A_{\mathcal{Q}} + B_{\mathcal{Q}} C_{\mathcal{Q}}$ lie inside the unit circle.

Finally, we discuss the gain $\|\mathcal{Q}\|_{wv}$ of the dynamic quantizer $\mathcal{Q}$. From Eq. (3.25), the gain $\|\mathcal{Q}\|_{wv}$ can be derived as follows:

$$\|\mathcal{Q}\|_{wv} = \left\| I + \sum_{k=0}^{\infty} \text{abs}(C_{\mathcal{Q}} (A_{\mathcal{Q}} + B_{\mathcal{Q}} C_{\mathcal{Q}})^k B_{\mathcal{Q}}) \right\|_{\infty}. \quad (3.26)$$

3.2.2 Optimal Dynamic Quantizer

In this section, we introduce a method for analytically designing the optimal dynamic quantizer [26, 28, 33, 45].

3.2. VIRTUAL DYNAMICS DESIGN METHODS

The solution to Problem 3.1 can be obtained analytically if either of the following conditions (C1) or (C2) is satisfied.

- (C1) All transmission zeros of $\mathcal{S}_I$ are stable, and using the smallest non-negative integer τ such that $C_1 \tilde{A}^\tau B_2 \neq 0$, both D_2 and $C_2 \tilde{A}^\tau B_2$ are row full rank.
- (C2) $m = N_r = N_z = 1$, $C_1 B_2$ is nonsingular, $D_2 \neq 0$, and there is exactly one unstable zero in the transfer function from $s := v - u$ to z .

When condition (C1) is satisfied, the optimal dynamic quantizer can be obtained as

$$A_Q^* = \tilde{A}, \quad (3.27)$$

$$B_Q^* = B_2, \quad (3.28)$$

$$C_Q^* = - \left(- \left(C_1 \tilde{A}^\tau B_2 \right)^\dagger \right) C_1 \tilde{A}^{\tau+1}, \quad (3.29)$$

and the value of the evaluation function can be expressed as

$$J_K(\mathcal{P} \circ \mathcal{Q}) = \left\| \text{abs} \left(C_1 \tilde{A}^\tau B_2 \right) \right\|_\infty \frac{d}{2}. \quad (3.30)$$

Also, when condition (C2) is satisfied, the optimal dynamic quantizer can be obtained as follows. First, by substituting $v[k] = u[k] + s[k]$ into Eq. (3.18), we have

$$\mathcal{S}_I : \begin{cases} x[k+1] = \tilde{A}x[k] + (B_1 + B_2 D_2) r[k] + B_2 s[k], \\ z[k] = C_1 x[k] + D_1 r[k]. \end{cases} \quad (3.31)$$

Let $\mathcal{G}_{zs}$ be the transfer function from s to z of $\mathcal{S}_I$, and let a be the unique unstable zero of $\mathcal{G}_{zs}$ and i the relative degree. Using these, $\mathcal{G}_{zs}(z)$ can generally be cascade-decomposed into

$$\mathcal{G}_{zs}(z) = \mathcal{L}_1(z) \mathcal{L}_2(z), \quad (3.32)$$

$$\mathcal{L}_1(z) = \frac{(z - a)}{z^i}, \quad \mathcal{L}_2 : (A_{\mathcal{L}_2}, B_{\mathcal{L}_2}, C_{\mathcal{L}_2}, 0), \quad (3.33)$$

where $\mathcal{L}_2$ is minimum-phase and $C_{\mathcal{L}_2} B_{\mathcal{L}_2} \neq 0$. Using these, the optimal dynamic quantizer can be derived as

$$A_Q^* = A_{\mathcal{L}_2}, \quad (3.34)$$

$$B_Q^* = B_{\mathcal{L}_2}, \quad (3.35)$$

$$C_Q^* = - (C_{\mathcal{L}_2} B_{\mathcal{L}_2})^{-1} C_{\mathcal{L}_2} A_{\mathcal{L}_2}. \quad (3.36)$$

The value of the evaluation function can also be derived as

$$J_K(\mathcal{P} \circ \mathcal{Q}) = (1 + |a|) |C_{\mathcal{L}_2} B_{\mathcal{L}_2}| \frac{d}{2}. \quad (3.37)$$

3.2.3 Relaxing to a Linear Programming Problem

In this section, we introduce a method to numerically solve Problem 3.1 by relaxing it to a linear programming (LP) problem [29]. As described in Section 3.2.2, the method for analytically designing the optimal dynamic quantizer is limited to cases where either condition (C1) or (C2) is satisfied. Furthermore, the optimal dynamic quantizer may not satisfy the gain constraint represented by Eq. (3.16) and the order constraint represented by Eq. (3.17). Even in such cases, the dynamic quantizer can be designed through numerical optimization. We introduce one such method in this section.

First, introduce variables $H_k \in \mathbb{R}^{N_z \times m}$ ($k = 0, 1, \dots, T-1$) corresponding to the Markov parameters of the dynamic quantizer. Then, we seek the solution $H_k^\bullet$ ($k = 0, 1, \dots, T-1$) of the following optimization problem:

Problem 3.2 Solve the following optimization problem obtained by relaxing Problem 3.1 to a linear programming problem.

$$\begin{aligned} & \underset{H_k (k=0,1,\dots,T-1)}{\text{minimize}} \quad \left\| \text{abs}(C_1 B_2) + \sum_{k=1}^{T-1} \text{abs} \left(C_1 \tilde{A}^k B_2 + \Phi_k \begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ H_{k-1} \end{bmatrix} \right) \right\|_\infty \end{aligned} \quad (3.38)$$

$$\text{subject to} \quad \left\| I + \sum_{k=0}^{T-1} \text{abs}(H_k) \right\|_\infty \leq \gamma_{wv}. \quad (3.39)$$

Here, Φ_k is a block matrix defined by

$$\Phi_k := \begin{bmatrix} C_1 \tilde{A}^{k-1} B_2 & C_1 \tilde{A}^{k-2} B_2 & \cdots & C_1 B_2 \end{bmatrix}. \quad (3.40)$$

The quantity minimized in Eq. (3.38) is the evaluation function $J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q})$ given by Eq. (3.22), truncated to a sum of T terms,

$$J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q}; T) := \left\| \sum_{j=0}^{T-1} \text{abs}(\bar{C} \bar{A}^j \bar{B}) \right\|_\infty. \quad (3.41)$$

Furthermore, Eq. (3.39) means that the gain $\|\mathcal{Q}\|_{wv}$ given by Eq. (3.26), truncated to a sum of T terms,

$$\|\mathcal{Q}\|_{wv; T} := \left\| I + \sum_{k=0}^{T-1} \text{abs}(C_{\mathcal{Q}}(A_{\mathcal{Q}} + B_{\mathcal{Q}}C_{\mathcal{Q}})^k B_{\mathcal{Q}}) \right\|_\infty, \quad (3.42)$$

3.2. VIRTUAL DYNAMICS DESIGN METHODS

is constrained to be less than or equal to γ_{wv} .

Next, using the obtained solution $H_k^\bullet$, we define a block Hankel matrix $H^\bullet \in \mathbb{R}^{mT' \times mT'}$ as

$$H^\bullet := \begin{cases} \begin{bmatrix} H_0^\bullet & H_1^\bullet & H_2^\bullet & \cdots & H_{T'-1}^\bullet \\ H_1^\bullet & H_2^\bullet & \ddots & H_{T'-1}^\bullet & \vdots \\ H_2^\bullet & \ddots & \ddots & \ddots & H_{T-3}^\bullet \\ \vdots & H_{T'-1}^\bullet & \ddots & H_{T-3}^\bullet & H_{T-2}^\bullet \\ H_{T'-1}^\bullet & \cdots & H_{T-3}^\bullet & H_{T-2}^\bullet & H_{T-1}^\bullet \end{bmatrix}, & \text{if } T \text{ is odd,} \\ \begin{bmatrix} H_0^\bullet & H_1^\bullet & H_2^\bullet & \cdots & H_{T'-1}^\bullet \\ H_1^\bullet & H_2^\bullet & \ddots & H_{T'-1}^\bullet & \vdots \\ H_2^\bullet & \ddots & \ddots & \ddots & H_{T-3}^\bullet \\ \vdots & H_{T'-1}^\bullet & \ddots & H_{T-3}^\bullet & H_{T-2}^\bullet \\ H_{T'-1}^\bullet & \cdots & H_{T-3}^\bullet & H_{T-2}^\bullet & * \end{bmatrix}, & \text{if } T \text{ is even,} \end{cases} \quad (3.43)$$

with $T' := \lfloor T/2 \rfloor + 1$. Here, when T is even, the last element $*$ can take arbitrary value. Furthermore, we perform singular value decomposition of $H^\bullet$ using matrices $W_o^\bullet$, $S^\bullet$, and $W_c^\bullet \in \mathbb{R}^{mT' \times mT'}$ as

$$H^\bullet = W_o^\bullet S^\bullet W_c^\bullet. \quad (3.44)$$

Using this, we define

$$A_Q^\circ := \left(\begin{bmatrix} I_{m(T'-1)} & 0_{m(T'-1) \times m} \end{bmatrix} W_o^\bullet (S^\bullet)^{\frac{1}{2}} \right)^\dagger \begin{bmatrix} 0_{m(T'-1) \times m} & I_{m(T'-1)} \end{bmatrix} W_o^\bullet (S^\bullet)^{\frac{1}{2}} - B_Q^\circ C_Q^\circ, \quad (3.45)$$

$$B_Q^\circ := (S^\bullet)^{\frac{1}{2}} W_c^\bullet \begin{bmatrix} I_m \\ 0_{m(T'-1) \times m} \end{bmatrix}, \quad (3.46)$$

$$C_Q^\circ := \begin{bmatrix} I_m & 0_{m \times m(T'-1)} \end{bmatrix} W_o^\bullet (S^\bullet)^{\frac{1}{2}}, \quad \tilde{A}_Q^\circ := A_Q^\circ + B_Q^\circ C_Q^\circ. \quad (3.47)$$

Then, if $\tilde{A}_Q^\circ$ is stable, the dynamic quantizer is obtained as

$$A_Q^* = A_Q^\circ, \quad (3.48)$$

$$B_Q^* = B_Q^\circ, \quad (3.49)$$

$$C_Q^* = C_Q^\circ, \quad (3.50)$$

3.2. VIRTUAL DYNAMICS DESIGN METHODS

otherwise, it is obtained as

$$A_Q^* = \begin{bmatrix} 0 & \tilde{A}_Q^\circ & 0 & \cdots & 0 \\ 0 & 0 & \tilde{A}_Q^\circ & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \tilde{A}_Q^\circ \\ -B_Q^\circ C_Q^\circ & -B_Q^\circ C_Q^\circ & \cdots & -B_Q^\circ C_Q^\circ & -B_Q^\circ C_Q^\circ \end{bmatrix}, \quad (3.51)$$

$$B_Q^* = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ B_Q^\circ \end{bmatrix}, \quad (3.52)$$

$$C_Q^* = \begin{bmatrix} C_Q^\circ & C_Q^\circ & \cdots & C_Q^\circ \end{bmatrix}. \quad (3.53)$$

3.2.4 Nonlinear Optimization for SISO Systems

In this section, we introduce a nonlinear optimization method [31] for single-input single-output (SISO) systems that explicitly handles order and gain constraints.

A minimally realized dynamic quantizer of order N_Q can be parameterized by a controllable canonical form as

$$A_Q = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ -a_{N_Q} & -a_{N_Q-1} & \cdots & -a_2 & -a_1 \end{bmatrix}, \quad (3.54)$$

$$B_Q = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad C_Q = \begin{bmatrix} c_{N_Q} & c_{N_Q-1} & \cdots & c_2 & c_1 \end{bmatrix}. \quad (3.55)$$

Here, $a_i, c_i \in \mathbb{R}$ ($i = 1, 2, \dots, N_Q$) are parameters. Next, with these collected into a parameter vector $\theta := [a_1, \dots, a_{N_Q}, c_1, \dots, c_{N_Q}]^\top \in \mathbb{R}^{2N_Q}$, we define functions representing the

3.3. DYNAMIC QUANTIZER DESIGN TOOLBOX NQLIB

constraint conditions as follows:

$$h_{\text{stability}}(\theta) := \max_i |\lambda_i(\theta)| - 1, \quad (3.56)$$

$$h_{\text{gain}}(\theta) := \|\mathcal{Q}\|_{wv; T} - \gamma_{wv}, \quad (3.57)$$

$$h_c(\theta) := \max \{h_{\text{stability}}(\theta), h_{\text{gain}}(\theta)\}. \quad (3.58)$$

Here, $\lambda_i(\theta)$ are the eigenvalues of the matrix $A_{\mathcal{Q}} + B_{\mathcal{Q}}C_{\mathcal{Q}}$. Therefore, $h_{\text{stability}}(\theta)$ takes a negative value if the dynamic quantizer $\mathcal{Q}$ is BIBO stable, and a non-negative value otherwise. Also, $h_{\text{gain}}(\theta)$ takes a negative value if the gain $\|\mathcal{Q}\|_{wv; T}$ of the dynamic quantizer $\mathcal{Q}$ is less than or equal to the constraint value γ_{wv} , and a non-negative value otherwise. Then the quantizer satisfies both BIBO stability and gain constraints if and only if $h_c(\theta)$ is negative. Thus, by minimizing the evaluation function defined as

$$E(\theta) = \begin{cases} -\exp(-J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q}; T)) & \text{if } h_c(\theta) < 0, \\ h_c(\theta) & \text{otherwise,} \end{cases} \quad (3.59)$$

and achieving a negative value, we can design a dynamic quantizer that satisfies the constraints. You can use gradient methods [46] or metaheuristics [31, 47] for the optimization.

3.3 Dynamic Quantizer Design Toolbox NQLib

In this section, we describe the dynamic quantizer design toolbox NQLib, developed to bridge the gap between existing theory and the practical prototyping needs of control engineers.

3.3.1 Overview of NQLib

Design methods for dynamic quantizers are diverse; each applies to different classes of systems and yields solutions with distinct properties. The method in Section 3.2.2 produces theoretically optimal quantizers, yet the resulting order is fixed, and some systems admit no solution. The approach in Section 3.2.3 can handle systems beyond those in Section 3.2.2 and enables fast optimization, but the resulting quantizer order can grow prohibitively large for implementation. The method in Section 3.2.4 allows explicit constraints on order and gain, but can only be used for SISO systems, requires significant optimization time, and may converge to local optima.

For control engineers, comprehending these trade-offs, selecting an appropriate method, implementing the optimization, simulating with the obtained quantizer, and conducting hardware experiments for comparison present a high barrier. To bridge this research-practice gap, we have developed the dynamic quantizer design toolbox NQLib. NQLib is a Python library

3.3. DYNAMIC QUANTIZER DESIGN TOOLBOX NQLIB

for integrated design, evaluation, and simulation of dynamic quantizers. With NQLib, we can:

- represent a plant $\mathcal{P}$, a controller $\mathcal{K}$, and an ideal system $\mathcal{S}_I$ using state-space models
- represent a static quantizer q , including mid-rise-type and mid-tread-type quantizers predefined
- represent a dynamic quantizer $\mathcal{Q}$ as a state-space model
- design dynamic quantizers using the methods described in Section 3.2
- evaluate designed dynamic quantizers in terms of order $N_{\mathcal{Q}}$, gain $\|\mathcal{Q}\|_{wv}$, BIBO stability, and the evaluation function $J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q})$
- reduce the order of designed dynamic quantizers using balanced truncation
- simulate the entire control system as an ideal system $\mathcal{S}_I$, as a system including a static quantizer, or as a system including a dynamic quantizer

There exists a MATLAB-based dynamic quantizer design tool called ODQLab [34, 48]. ODQLab features both graphical and command-line interfaces for designing, evaluating, and simulating dynamic quantizers. Compared to ODQLab, NQLib has the following characteristics. First, NQLib is implemented in Python [49]. Python is a free, open-source programming language with a vibrant global development community. Thus, NQLib is freely available, including its environment, and operates across a wide range of platforms. Python also has libraries for control system design, such as the Python Control Systems Library [50], enabling integrated control system design. Second, NQLib implements a variety of methods for dynamic quantizer design. Specifically, in addition to the method for cases satisfying condition (C1) of Section 3.2.2 and the method of Section 3.2.3 implemented in ODQLab, NQLib also implements the method for cases satisfying condition (C2) of Section 3.2.2 and the method of Section 3.2.4. Thus, NQLib can design dynamic quantizers for systems that ODQLab cannot handle.

3.3.2 Workflow Using NQLib

Typical usage consists of the following steps:

1. **Specify** the ideal system $\mathcal{S}_I$, quantization width d , and constraints γ_{wv} , $N_{\max}$. You can use `Plant`, `Controller`, and `System` classes to represent an ideal system.
2. **Attempt** some design method using methods implemented in `DynamicQuantizer`:
 - `design_AG()`: algebraic method described in Section 3.2.2.
 - `design_LP()`: LP-based method from Section 3.2.3.
 - `design_GD()`: gradient-based optimization for SISO from Section 3.2.4.

3.3. DYNAMIC QUANTIZER DESIGN TOOLBOX NQLIB

- `design_DE()`: differential evolution for SISO from Section 3.2.4.
 - `design()`: general method trying appropriate design methods based on system specifications and constraints.
3. **Evaluate** resulting $\mathcal{Q}$: compute stability, gain, surrogate bound $\sum_{j \geq 0} \|C_e \bar{A}^j \bar{B}_w\|_\infty$ (truncated at T), and simulate z_D vs z_I to approximate J_K .
 4. **Iterate** constraints or method selection based on trade-offs (e.g., reduce N_Q accepting mild performance loss).

3.3.3 Illustrative Design Example

To illustrate NQLib's utility and usage, we present a simple design example. First, we show how to install NQLib. NQLib is available as an open-source project at <https://github.com/knttnk/NQLib/>. Documentation is provided at <https://knttnk.github.io/NQLib/>, including simple examples and tutorials. The documentation is also included in Chapter A. Furthermore, a tutorial notebook running on Google Colab [51] is available, allowing users to try NQLib without setting up a local environment. The library is licensed under the MIT License, permitting commercial use. NQLib is distributed via conda-forge and PyPI [52] at <https://pypi.org/project/nqlib/>, allowing users to install it through these channels:

```
pip install nqlib
# or
conda install -c conda-forge nqlib
```

Listing 3.1: Installing NQLib

NQLib v1.0.1 depends on Python [49] 3.11 or later, NumPy [53], SciPy [54] 1.24 or later, Slycot [55], Python Control Systems Library [50, 56], and CVXY [57, 58], which are also installed automatically.

Next, we import NQLib and define the system representing the control system.

```
import nqlib
```

Listing 3.2: Importing NQLib

We use the same parameters as in the simulation of Fig. 3.3, given by Eqs. (3.3) to (3.8). Such a plant and controller can be represented in NQLib as follows:

```
P = nqlib.Plant(
    A=[[1.19, 2.22],
        [0.02, -0.49]],
```

3.3. DYNAMIC QUANTIZER DESIGN TOOLBOX NQLIB

```
B=[[1.],
    [1.]],
C1=[[1., 1.]],
C2=[[1., 1.]],
)
K = nqlib.Controller(
    A=[[0.833, 1.95],
        [-0.928, -1.351]],
    B1=[[1.],
        [0.]],
    B2=[[0.013],
        [0.604]],
    C=[[-0.344, -0.257]],
    D1=[[0]],
    D2=[[0]],
)
ideal_system = nqlib.System.from_FB_connection_with_input_quantizer(
    P, K)
```

Listing 3.3: System description

We also set the simulation time, initial state, and reference input $r[k]$ as follows:

```
import numpy as np
t = np.arange(101)
x0 = [[-3.21],
      [-0.15],
      [0.],
      [0.]]
r = np.where(np.sin(2 * np.pi * t / 50) >= 0, 1.0, -1.0)
max_gain_wv = 2
```

Listing 3.4: Simulation settings

We adopt a mid-tread quantizer q with $d = 1/2$, as in Fig. 3.3. NQLib has both mid-tread and mid-rise quantizers predefined. Thus, the quantizer in this case can be succinctly described as follows:

```
q = nqlib.StaticQuantizer.mid_tread(d=0.5)
```

Listing 3.5: Static quantizer definition

To simulate the continuous-valued input system $\mathcal{S}_I$ and discrete-valued input system $\mathcal{S}_D$ with the above settings, we write as follows:

3.3. DYNAMIC QUANTIZER DESIGN TOOLBOX NQLIB

```
response_ideal = ideal_system.response(r, x0)
response_static_q = ideal_system.response_with_quantizer(q, r, x0)
```

Listing 3.6: Simulation of continuous-valued and discrete-valued input systems

The results of this simulation have already been shown in Fig. 3.2. Furthermore, to compute the evaluation function at this time, we write as follows:

```
ideal_system.E(q)
```

Listing 3.7: Evaluation function calculation for static quantizer

This yields an evaluation function value of $J_{\mathcal{K}}(\mathcal{P} \circ q) = 2.35$.

As the first design method, let's try designing a dynamic quantizer using the algebraic method introduced in Section 3.2.2. To do this, we write as follows:

```
Q_AG, E_AG = nqlib.DynamicQuantizer.design_AG(
    ideal_system,
    q=q,
)
```

Listing 3.8: Designing dynamic quantizer using algebraic method

As a result, the dynamic quantizer is obtained as

$$A_{\mathcal{Q}} = \begin{bmatrix} 1.19 & 2.22 & -0.344 & -0.257 \\ 0.02 & -0.49 & -0.344 & -0.257 \\ 0.013 & 0.013 & 0.833 & 1.95 \\ 0.604 & 0.604 & -0.928 & -1.351 \end{bmatrix}, \quad B_{\mathcal{Q}} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad (3.60)$$

$$C_{\mathcal{Q}} = \begin{bmatrix} -0.605 & -0.865 & 0.344 & 0.257 \end{bmatrix}. \quad (3.61)$$

To simulate using this dynamic quantizer Q_AG , we write as follows:

```
response_Q_AG = ideal_system.response_with_quantizer(Q_AG, r, x0)
```

Listing 3.9: Simulation using dynamic quantizer designed by algebraic method

Figure 3.7 shows the simulation result. Comparing with Fig. 3.3, we see that the output $z[k]$ of the control system with the dynamic quantizer is closer to that of the ideal continuous-valued input control system. Also, checking the evaluation function value at this time using the code below yields $J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q}) = 0.50$:

```
ideal_system.E(Q_AG)
```

3.3. DYNAMIC QUANTIZER DESIGN TOOLBOX NQLIB

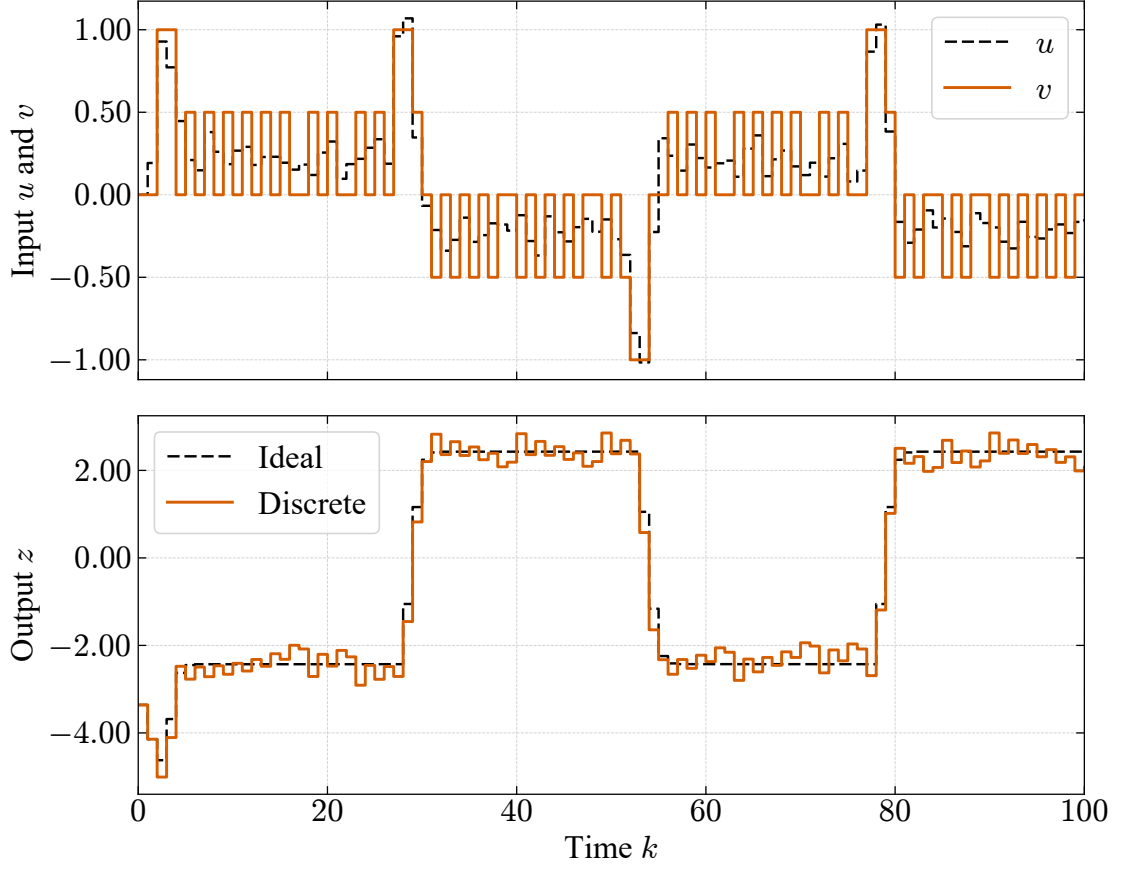


Fig. 3.7: Simulation result of a control system with the dynamic quantizer designed by the algebraic method described in Section 3.2.2

Listing 3.10: Evaluation function calculation for dynamic quantizer designed by algebraic method

Compared to $J_{\mathcal{K}}(\mathcal{P} \circ q) = 2.35$ when using the static quantizer, we see that the optimal dynamic quantizer improves the evaluation function value.

Now, suppose we impose a gain constraint of 2. To compute the gain of the obtained dynamic quantizer, we write as follows:

```
gain_AG = Q_AG.gain_wv( )
```

Listing 3.11: Gain calculation for dynamic quantizer designed by algebraic method

The gain of the obtained dynamic quantizer is 5.22, which does not satisfy the constraint. Thus, we design a dynamic quantizer satisfying the gain constraint using the LP-based method introduced in Section 3.2.3. To do this, we write as follows:

3.3. DYNAMIC QUANTIZER DESIGN TOOLBOX NQLIB

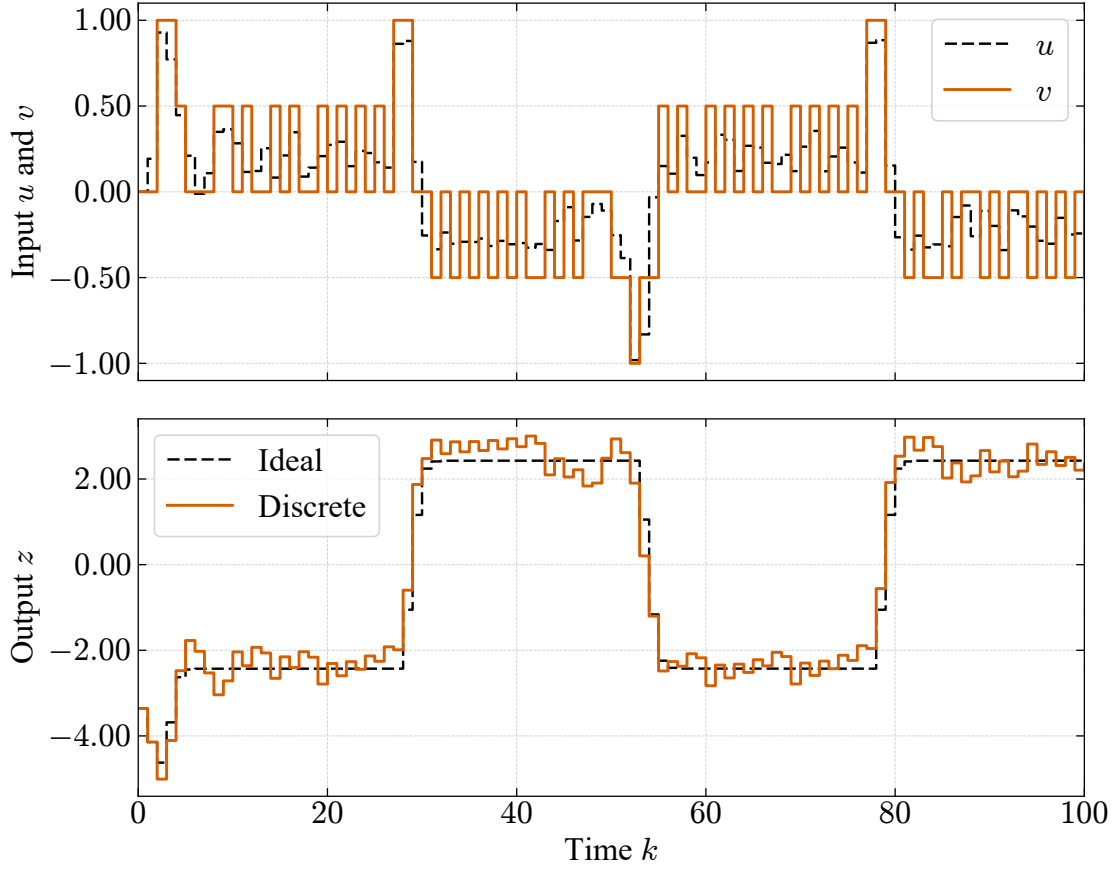


Fig. 3.8: Simulation result of a control system with the dynamic quantizer designed by the LP-based method described in Section 3.2.3

```
Q_LP, E_LP = nqlib.DynamicQuantizer.design_LP(
    ideal_system,
    q=q,
    T=100,
    max_gain_wv=max_gain_wv,
)
```

Listing 3.12: Designing dynamic quantizer using LP-based method

The quantizer obtained in this way has specifications $J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q}) = 1.17$ and gain 1.99. While the evaluation function value worsens in exchange for satisfying the gain constraint, it still achieves better performance than when using the static quantizer. Figure 3.8 shows the simulation result using this dynamic quantizer. Comparing with the case using the above optimal dynamic quantizer, the output $z[k]$ deviates somewhat from that of the ideal continuous-valued

3.3. DYNAMIC QUANTIZER DESIGN TOOLBOX NQLIB

input control system, but it still achieves better performance than when using the static quantizer.

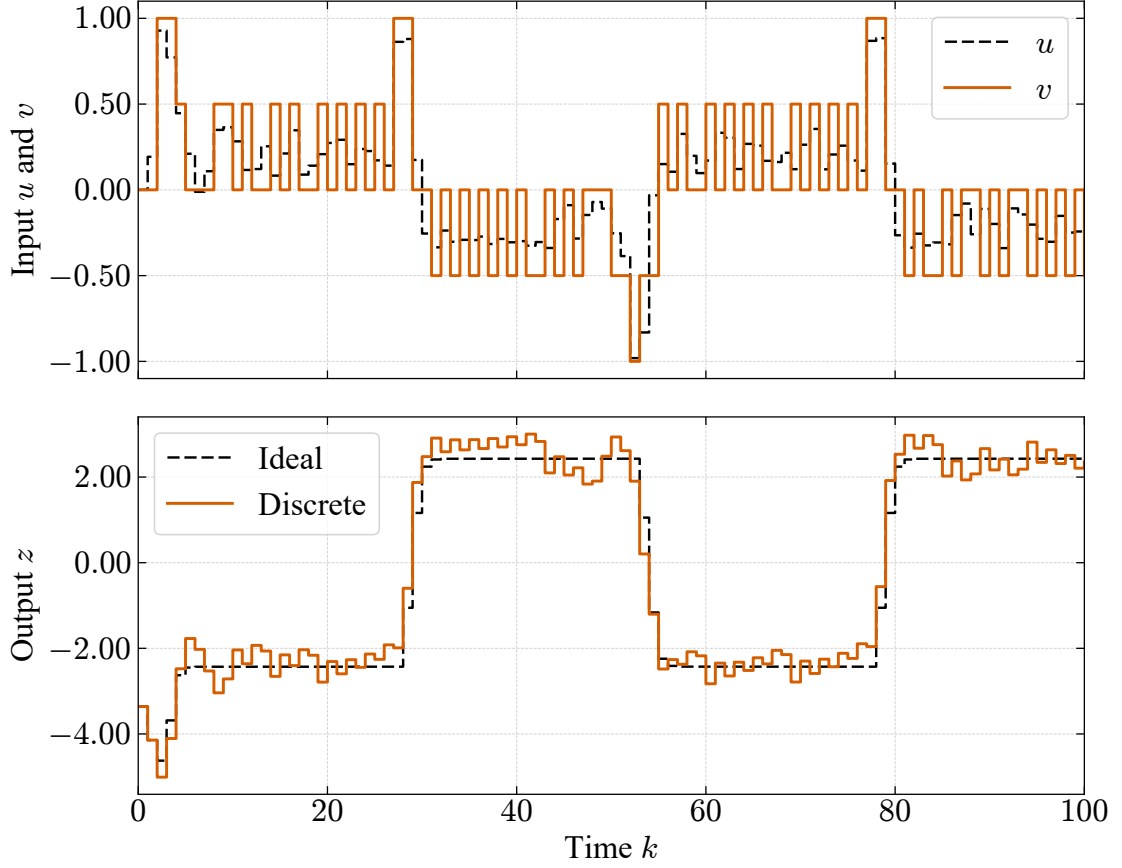


Fig. 3.9: Simulation result of a control system with the dynamic quantizer designed by the LP-based method with order reduction described in Section 3.2.3

To check the order of the obtained dynamic quantizer, we write as follows:

```
Q_LP.N
```

Listing 3.13: Order calculation for dynamic quantizer designed by LP-based method

This reveals that the obtained dynamic quantizer has an order of 5151. To enable implementation on inexpensive computers, we constrain the order to 2 and attempt order reduction using NQLib's order reduction feature. To reduce the order of `Q_LP`, we write as follows:

```
N_reduced = 2
Q_LP_reduced = Q_LP.order_reduced(N_reduced)
```

Listing 3.14: Order reduction of dynamic quantizer designed by LP-based method

3.3. DYNAMIC QUANTIZER DESIGN TOOLBOX NQLIB

The resulting dynamic quantizer has specifications $J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q}) = 1.17$, gain 2.01, and order 2. Although the gain constraint is slightly exceeded, the evaluation function value is preserved, yielding an implementable dynamic quantizer. Figure 3.9 shows the simulation result using this dynamic quantizer. Comparing with the case using the previous dynamic quantizer satisfying the gain constraint, the output $z[k]$ deviates somewhat from that of the ideal continuous-valued input control system, but it still achieves better performance than when using the static quantizer.

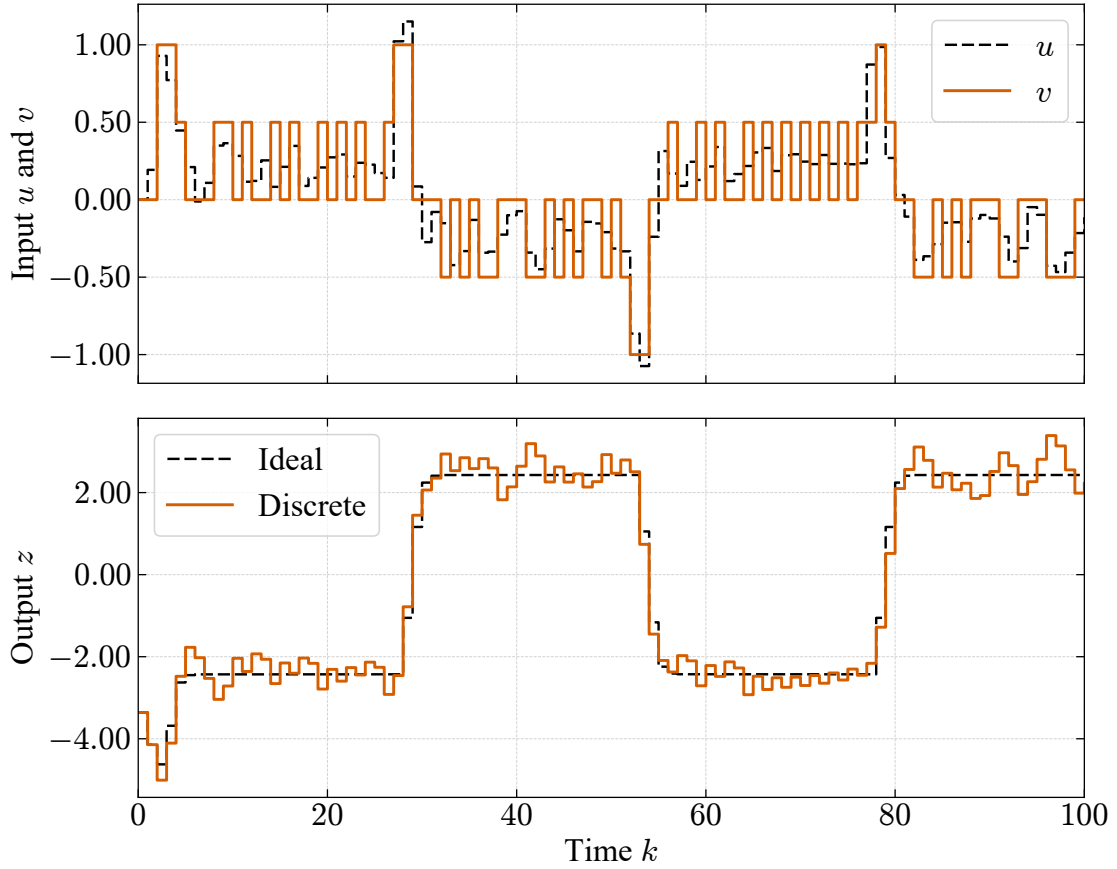


Fig. 3.10: Simulation result of a control system with the dynamic quantizer designed by the gradient-based method described in Section 3.2.4

Finally, we design a dynamic quantizer satisfying both gain and order constraints using one of the methods introduced in Section 3.2.4, specifically the gradient-based method. To do this, we write as follows:

```
Q_GD, E_GD = nqlib.DynamicQuantizer.design_GD(
    ideal_system,
    q=q,
```

3.4. CONCLUSION

```
N=N_reduced,  
steptime=100,  
max_gain_wv=max_gain_wv,  
)
```

Listing 3.15: Designing dynamic quantizer using gradient-based method

The quantizer obtained has specifications $J_{\mathcal{K}}(\mathcal{P} \circ \mathcal{Q}) = 1.19$, gain 2.00, and order 2. While the evaluation function value slightly worsens, we obtain a dynamic quantizer satisfying both gain and order constraints. Figure 3.10 shows the simulation result using this dynamic quantizer. Although the evaluation function value slightly worsens, it still achieves better performance than when using the static quantizer.

3.4 Conclusion

In this chapter, we discussed methods for designing dynamic quantizers in discrete-valued input control, viewing them as Alterers that hide discrete-valued input constraints from the controller. We explained various design methods for dynamic quantizers and summarized the strengths of each approach. Furthermore, we described the NQLib toolbox, enabling control engineers to practically design, evaluate, and simulate dynamic quantizers. Finally, by presenting a design example using NQLib, we demonstrated the effectiveness of Virtual Dynamics in controlling discrete-valued input systems.

Chapter 4

Virtual Dynamics Design for Systems with Reinforcement Learning

In this chapter, we apply Problem 2.1, the Virtual Dynamics design problem, to systems using reinforcement learning. Reinforcement learning is a framework in which an agent learns a policy for selecting actions based on observed states to maximize the resulting rewards [59]. Recently, deep reinforcement learning [60] (hereafter simply RL), which employs deep neural networks for learning, has made remarkable progress, as exemplified by the success of AlphaGo [35, 61]. Application to control systems is also active, with research in various fields such as robotics [62, 63] and autonomous driving [64]. RL often requires many trials, time, and power to achieve satisfactory performance, making improving learning efficiency an important challenge [65, 66]. As a solution to this problem, in this chapter, we insert an Alterer that alters the state obtained from the plant before the RL agent, aiming to improve learning efficiency through its design.

Section 4.1 formulates the design problem of Alterers and Virtual Dynamics to enhance RL learning efficiency. Section 4.2 then experimentally demonstrates, using RL for stabilization of an inverted pendulum as an example, that Virtual Dynamics can improve learning efficiency. Finally, Section 4.3 discusses minimum-phase considerations as design guidelines for Alterers in RL.

4.1. DESIGN PROBLEM

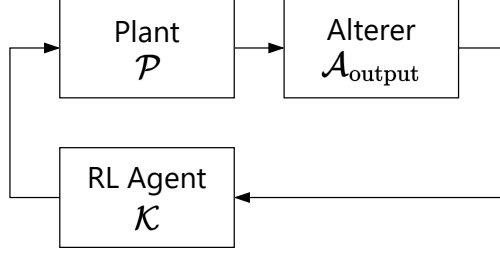


Fig. 4.1: Block diagram of a control system where the RL agent serves as the controller

4.1 Design Problem

As a method to enhance learning efficiency, reward shaping, which provides rewards different from the objective function, is known [67]. Moreover, by analogy with the decisive role of preprocessing such as normalization and standardization in supervised learning, it is plausible in RL to enhance efficiency by intentionally altering the agent's observations. Accordingly, we insert an Alterer that alters the plant state x before the controller (agent), as illustrated in Fig. 4.1, to improve learning efficiency through its design. As the evaluation function J_K in Problem 2.1, we adopt a metric that decreases when the agent reaches the target performance earlier. Specifically, let $t \in \mathbb{N}$ denote the learning time step, $S(t) \in \mathbb{R}$ the performance at time step t , and $S_{\text{target}} \in \mathbb{R}$ the target performance; we define

$$J_K(\mathcal{A}_{\text{output}} \circ \mathcal{P}) = \min\{t \mid S(t) \geq S_{\text{target}}\}. \quad (4.1)$$

This means the number of time steps required for the performance $S(t)$ to reach the target performance S_{target} .

We define the performance $S(t)$ as follows. First, specify the target state set that the plant state x must satisfy as $\mathbb{X}_{\text{target}} \subseteq \mathbb{R}^n$. Each learning episode starts from a state within $\mathbb{X}_{\text{target}}$ and terminates when x first exits $\mathbb{X}_{\text{target}}$. Next, let $t_0(t) \in \mathbb{N}$ denote the start time of the episode containing step t . Using these, define the performance $S(t)$ by

$$S(t) = t - t_0(t). \quad (4.2)$$

That is, $S(t)$ measures how long the target state has been maintained at time t .

Based on the above evaluation function and performance definition, we pose the following design problem.

4.2. PRELIMINARY EXAMPLE: IMPROVING LEARNING EFFICIENCY VIA VIRTUAL DYNAMICS

Problem 4.1 Given a plant $\mathcal{P}$, suppose the RL agent $\mathcal{K}$ controls the plant so that its state x remains within a set $\mathbb{X}_{\text{target}}$. With the performance $S(t)$ at time step t given by Eq. (4.2) and a specified target performance S_{target} , design an Alterer $\mathcal{A}_{\text{output}}$ that minimizes the evaluation function $J_{\mathcal{K}}$ defined in Eq. (4.1).

4.2 Preliminary Example: Improving Learning Efficiency via Virtual Dynamics

In this section, we consider RL for stabilization of an inverted pendulum as a concrete example where Virtual Dynamics via the Alterer $\mathcal{A}_{\text{output}}$ improves learning efficiency. It is not a priori evident that there exists an Alterer whose insertion improves learning efficiency. Our objective here is therefore to demonstrate the existence of $\mathcal{A}_{\text{output}}$ that reduces the evaluation function Eq. (4.1)—namely, achieves earlier attainment of the target performance—and to verify that Alterer design can contribute to more efficient RL.

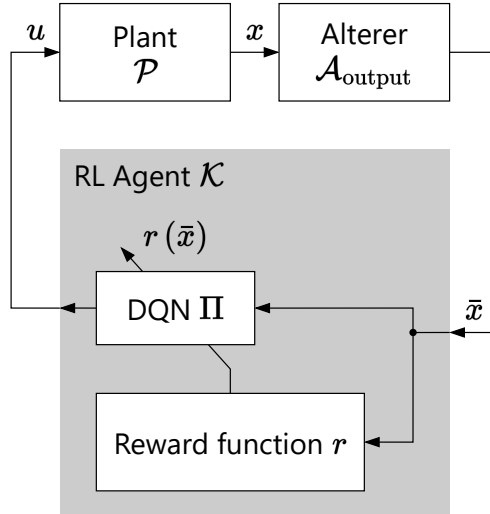


Fig. 4.2: Block diagram of the control system where the DQN agent serving as the controller receives altered information from the Alterer

We consider the control system shown in Fig. 4.2, comprising the plant $\mathcal{P}$, an Alterer $\mathcal{A}_{\text{output}}$ that transforms the plant state x into $\bar{x}$, and the RL agent $\mathcal{K}$ controlling them. The details of each block are provided below.

4.2. PRELIMINARY EXAMPLE: IMPROVING LEARNING EFFICIENCY VIA VIRTUAL DYNAMICS

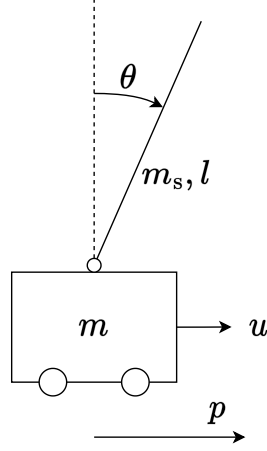


Fig. 4.3: Cart-type inverted pendulum

4.2.1 Plant: Cart-Type Inverted Pendulum

The plant $\mathcal{P}$ is a cart-type inverted pendulum model shown in Fig. 4.3, given by

$$\mathcal{P} : \dot{x}(t) = f(x(t)) + h(x(t))u(t). \quad (4.3)$$

Here,

$$f(x) := \begin{bmatrix} \dot{p} \\ \dot{\theta} \\ f_3(x) \\ f_4(x) \end{bmatrix}, \quad h(x) := \begin{bmatrix} 0 \\ 0 \\ \frac{1}{M(x)} \\ -\frac{3}{2M(x)l} \cos \theta \end{bmatrix}, \quad (4.4)$$

$$f_3(x) := -\frac{3gm}{8M(x)} \sin 2\theta - \frac{D_p}{M(x)} \dot{p} + \frac{3D_\theta}{2M(x)l} \dot{\theta} \cos \theta + \frac{lm_s}{2M(x)} \dot{\theta}^2 \sin \theta, \quad (4.5)$$

$$f_4(x) := \frac{3gm(m+m_s)}{2M(x)lm_s} \sin \theta + \frac{3D_p}{2M(x)l} \dot{p} \cos \theta - \frac{3D_\theta(m+m_s)}{M(x)l^2m_s} \dot{\theta} - \frac{3m_s}{8M(x)} \dot{\theta}^2 \sin 2\theta, \quad (4.6)$$

$$M(x) := m + \frac{m_s}{4}(1 + 3 \sin^2 \theta),$$

$t \in \mathbb{R}$ denotes time, $g \in \mathbb{R}_{>0}$ the gravitational acceleration, $m \in \mathbb{R}_{>0}$ the cart mass, $m_s \in \mathbb{R}_{>0}$ the pole mass, $l \in \mathbb{R}_{>0}$ the pole length, $D_p \in \mathbb{R}_{>0}$ the cart viscous friction coefficient, $D_\theta \in \mathbb{R}_{>0}$ the pendulum viscous friction coefficient, $p(t) \in \mathbb{R}$ the cart position, $\theta(t) \in \mathbb{R}$

4.2. PRELIMINARY EXAMPLE: IMPROVING LEARNING EFFICIENCY VIA VIRTUAL DYNAMICS

the angle from the upright vertical, $u(t) \in \mathbb{R}$ the control input (force on the cart), and $x(t) = [p, \theta, \dot{p}, \dot{\theta}]^\top \in \mathbb{R}^4$ the state vector.

4.2.2 Alterer: Prediction

Linearizing the plant $\mathcal{P}$ around the origin yields

$$\dot{x}(t) = A_{\text{lin}}x(t) + B_{\text{lin}}u(t), \quad (4.7)$$

$$A_{\text{lin}} := \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{3gm}{4M_c} & -\frac{D_p}{M_c} & \frac{3D_\theta}{2M_cl} \\ 0 & \frac{3gm(m+m_s)}{2M_clm_s} & \frac{3D_\theta}{2M_cl} & -\frac{3D_\theta(m+m_s)}{M_cl^2m_s} \end{bmatrix}, \quad (4.8)$$

$$B_{\text{lin}} := \begin{bmatrix} 0 \\ 0 \\ \frac{1}{M_c} \\ -\frac{3}{2M_cl} \end{bmatrix}, \quad M_c := m + \frac{m_s}{4}. \quad (4.9)$$

Using these, we define the Alterer as

$$\mathcal{A}_{\text{output}} : \bar{x}(t) = e^{A_{\text{lin}}T}x(t) \quad (4.10)$$

Here, $T \in \mathbb{R}_{>0}$ denotes the prediction period. Thus, $\bar{x}(t)$ is the predicted state at time $t + T$ under the linearized model with the input fixed at $u(t) = 0$. Although not the exact future state due to nonlinearity and input neglect, Eq. (4.10) enables earlier inference—e.g., from the sign of the angle component $\bar{x}_2$ —of the imminent falling direction, which is expected to facilitate value-function approximation.

4.2.3 Controller: Reinforcement Learning with Deep Q Network

The controller $\mathcal{K}$ outputs the control input u based on the altered state $\bar{x}$ as follows:

$$\mathcal{K} : u(t) = \Pi(\bar{x}(t)). \quad (4.11)$$

Here, $\Pi : \mathbb{R}^4 \rightarrow \mathbb{R}$ is the policy learned by the RL agent. This policy Π is a Deep Q Network (DQN) that selects control inputs from the discrete action set $\mathbb{U} = \{-u_{\max}, 0, u_{\max}\}$ with $u_{\max} \in \mathbb{R}_{>0}$. With control period $T_c \in \mathbb{R}_{>0}$, at time step $k \in \mathbb{N}$, the following operations are performed:

1. Observe the altered state $\bar{x}((k-1)T_c)$.

4.2. PRELIMINARY EXAMPLE: IMPROVING LEARNING EFFICIENCY VIA VIRTUAL DYNAMICS

2. Select an action $a_k \in \mathbb{U}$.
3. Apply $u(t) = a_k$ for $t \in \{t \in \mathbb{R} | (k-1)T_c \leq t < kT_c\}$.
4. Obtain the reward $r(\bar{x}(kT_c))$.

Here, $r : \mathbb{R}^4 \rightarrow \mathbb{R}$ is the reward function.

The target state set $\mathbb{X}_{\text{target}}$ that the state x must satisfy is defined as

$$\mathbb{X}_{\text{target}} = \{x \in \mathbb{R}^4 \mid |x_1| \leq p_{\max}, |x_2| \leq \theta_{\max}\}. \quad (4.12)$$

Here, $p_{\max}, \theta_{\max} \in \mathbb{R}_{>0}$ are the allowable maxima for cart position and pendulum angle. As the reward function guiding DQN learning, we set

$$r(\bar{x}) = 1 - c_p \left(\frac{\bar{x}_1}{p_{\max}} \right)^2 - c_\theta \left(\frac{\bar{x}_2}{\theta_{\max}} \right)^2 \quad (4.13)$$

where $c_p, c_\theta \in \mathbb{R}_{>0}$ are weighting coefficients. We consider c_p and c_θ to be small. The DQN receives a reward of approximately 1 per step, which slightly decreases as the altered state $\bar{x}$ approaches the boundary of $\mathbb{X}_{\text{target}}$.

The DQN uses 4 input nodes, two hidden layers with 128 nodes each, and 3 output nodes. The hidden-layer activation is ReLU. The discount factor γ is chosen so that responsibility halves over 0.5 s:

$$\gamma = \left(\frac{1}{2} \right)^{\frac{T_c}{0.5}}. \quad (4.14)$$

We adopt the ε -greedy exploration policy, with ε in episode s given by

$$\varepsilon = 0.01 + (0.8 - 0.01) \exp\left(-\frac{s}{1000}\right). \quad (4.15)$$

The total number of episodes is set to 301, and the replay memory capacity is 10^4 . Furthermore, we employ the soft update of the target network from [68] with $\tau = 0.005$.

4.2.4 Numerical Experiments and Results

We conduct numerical experiments to assess the effect of Alterer $\mathcal{A}_{\text{output}}$ in Eq. (4.10) on RL.

The initial state x_0 for each episode is drawn independently from a uniform distribution with each component bounded in absolute value by 0.01. The task score σ is defined by

$$\sigma = \begin{cases} 60 & \text{if } \bar{\sigma} \geq 60, \\ \bar{\sigma} & \text{else.} \end{cases} \quad (4.16)$$

4.3. DESIGN GUIDELINES VIA MINIMUM-PHASE CONSIDERATIONS

Here, $\bar{\sigma} \in \mathbb{R}$ denotes the time when $x(t)$ first exits $\mathbb{X}_{\text{target}}$, indicating when the pendulum falls, or the cart exceeds a certain range. We use σ only for evaluation, not for training. In other words, training aims to maximize σ , while the reward supplied to the agent is r from Eq. (4.13). We set $p_{\max} = 12$, $\theta_{\max} = \pi/4$, $c_p = 4 \times 10^{-4}$, $c_\theta = 0.05$, $u_{\max} = 3$, and $T_c = 1/30$. The simulation parameters are $g = 9.81$, $m = 0.5$, $m_s = 1$, $l = 2$, $D_p = 0.2$, $D_\theta = 0.06$.

Under these settings, we recorded the performance evolution over learning for 20 trials with prediction periods $T = 0$ and $T = 1$. For each configuration, Fig. 4.4 plots the minimum, 25th percentile, median, 75th percentile, and maximum scores versus the episode index. We observe that with $T = 1$, the score tends to reach 60 s earlier, indicating better learning efficiency and yielding $J_{\mathcal{K}}(\mathcal{A}_{\text{output}} \circ \mathcal{P}) < J_{\mathcal{K}}(\mathcal{P})$. In addition, the final performance also tends to be higher in this experiment, indicating improved control performance.

4.2.5 Discussion

Alterer Eq. (4.10) does not yield the exact future state due to nonlinearity and neglect of inputs. However, short-term prediction likely accelerates discrimination of falling direction, contributing to reduced variance in value updates and improved exploration efficiency.

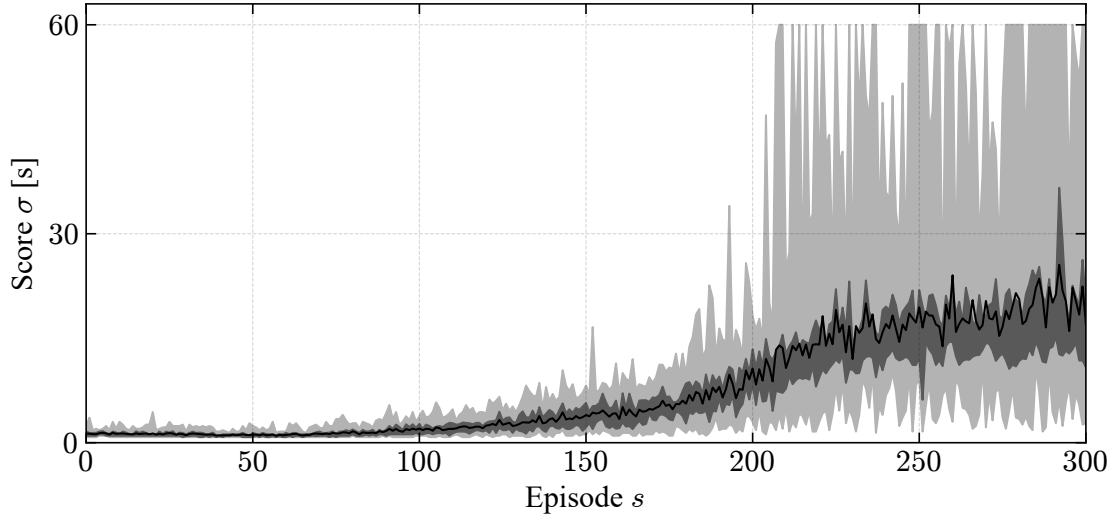
The results of this experiment provide a pioneering example of enhancing the sample efficiency of learning algorithms by altering the information presented to the agent via Virtual Dynamics. In summary, inserting a prediction-type static Alterer as Virtual Dynamics can improve the evaluation function Eq. (4.1) in control systems using RL. Extensions using longer prediction periods or dynamic (internally stateful) predictors, designs of Alterers incorporating nonlinear model prediction or uncertainty estimation, and applications to other plants remain future work.

4.3 Design Guidelines via Minimum-Phase Considerations

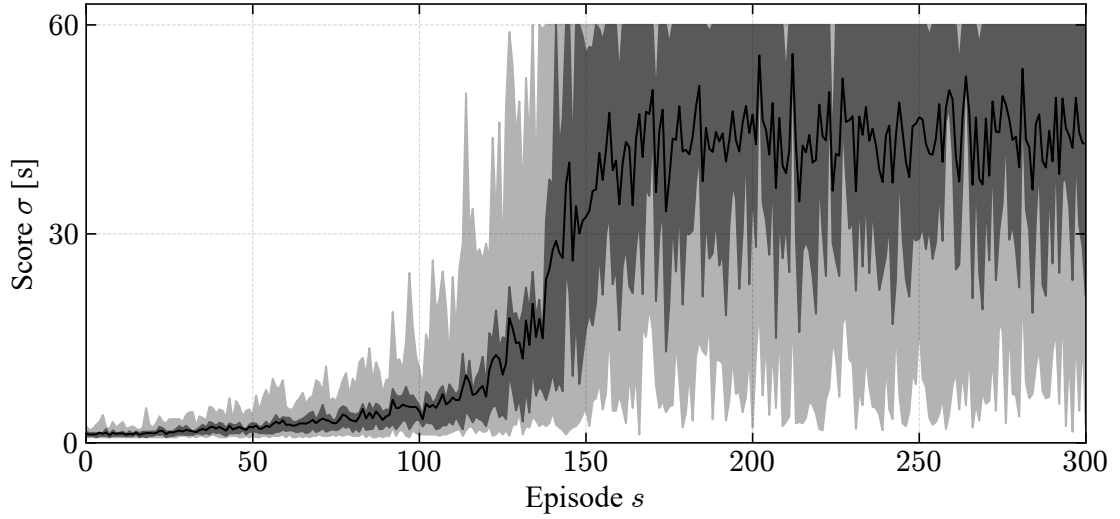
In Section 4.2, we demonstrated experimentally, using an inverted pendulum task, that Virtual Dynamics can improve RL efficiency. However, the principles underlying learning-efficiency improvements via Virtual Dynamics, which are necessary for deriving design guidelines, remain unclear. Therefore, in this section, we investigate the relationships among properties of the plant, properties acquired by Virtual Dynamics, and learning efficiency using a simpler plant to derive design guidelines for $\mathcal{A}_{\text{output}}$.

Specifically, we consider the control system shown in Fig. 4.5. We investigate how the

4.3. DESIGN GUIDELINES VIA MINIMUM-PHASE CONSIDERATIONS



(a) With $T = 0$



(b) With $T = 1$

Fig. 4.4: Score evolution over 20 trials of learning inverted-pendulum stabilization, plotted with minimum, 25th percentile, median, 75th percentile, and maximum values

4.3. DESIGN GUIDELINES VIA MINIMUM-PHASE CONSIDERATIONS

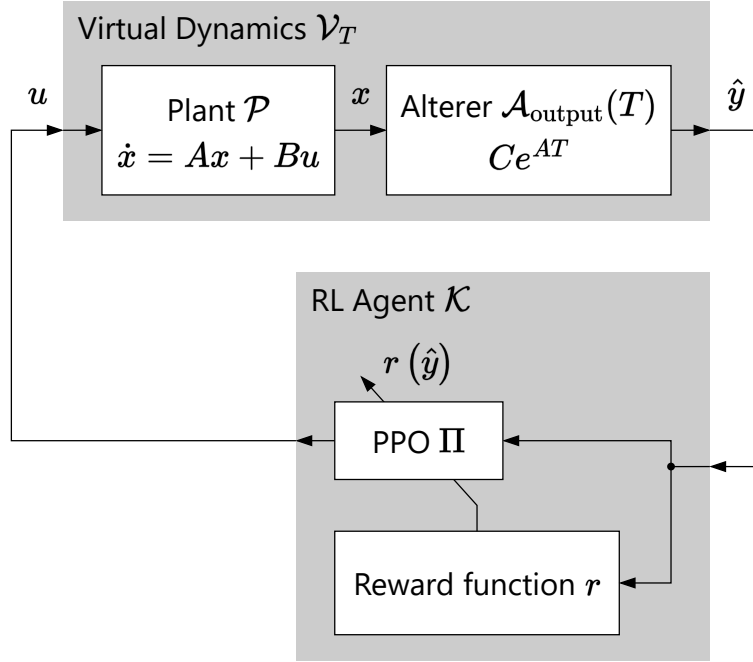


Fig. 4.5: Block diagram of the control system where the PPO agent serving as the controller receives altered information from the Alterer

learning efficiency of the RL agent $\mathcal{K}$ depends on the minimum-phase properties of the plant $\mathcal{P}$ and the Virtual Dynamics $\mathcal{A}_{\text{output}} \circ \mathcal{P}$. The details of each block are provided below.

4.3.1 Plant: Second-Order Linear System

The plant $\mathcal{P}$ is given by

$$\mathcal{P} : \dot{x}(t) = Ax(t) + Bu(t) \quad (4.17)$$

$$A := \begin{bmatrix} p_1 + p_2 & -p_1 p_2 \\ 0 & 1 \end{bmatrix}, \quad B := \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \quad (4.18)$$

Here, $t \in \mathbb{R}$ is time, $x(t) \in \mathbb{R}^2$ is the state, and $u(t) \in \mathbb{R}$ is the control input. Also, $p_1, p_2 \in \mathbb{C}$ are the system poles (a real pair or a complex-conjugate pair).

4.3.2 Alterer: Prediction

Analogously to Section 4.2, for a prediction period $T \in \mathbb{R}_{\geq 0}$, define the Alterer $\mathcal{A}_{\text{output}}(T)$ as

$$\mathcal{A}_{\text{output}}(T) : \hat{y}(t) = Ce^{AT}x(t), \quad (4.19)$$

$$C := \begin{bmatrix} 1 & -z \end{bmatrix}. \quad (4.20)$$

Here, $\hat{y}(t) \in \mathbb{R}$ is the altered output, and $z \in \mathbb{R}$ is a parameter. We also define the Virtual Dynamics $\mathcal{V}_T := \mathcal{A}_{\text{output}}(T) \circ \mathcal{P}$. When $T = 0$, the Virtual Dynamics becomes

$$\mathcal{V}_0 : \begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ \hat{y}(t) = Cx(t), \end{cases} \quad (4.21)$$

and its transfer function is

$$\mathcal{V}_0(s) = \frac{s - z}{(s - p_1)(s - p_2)}. \quad (4.22)$$

Thus, z is the parameter representing the zero in the absence of prediction.

The state equations of the Alterer $\mathcal{V}_T$ can be written as

$$\mathcal{V}_T : \begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ \hat{y}(t) = Ce^{AT}x(t). \end{cases} \quad (4.23)$$

This predicts the output of $\mathcal{V}_0$ at time $t + T$, assuming zero input. From Eq. (4.23), we can deduce two points: First, the poles of $\mathcal{V}_T$ are the eigenvalues of A , identical to those of the original Virtual Dynamics $\mathcal{V}_0$. Second, the zero and gain of $\mathcal{V}_T$ may vary depending on the prediction period T . Summarizing, using the zero $z_{\text{virtual}}(T)$ and gain $k_{\text{virtual}}(T)$ that vary with T , the transfer function of $\mathcal{V}_T$ can be expressed as

$$\mathcal{V}_T(s) = k_{\text{virtual}}(T) \frac{s - z_{\text{virtual}}(T)}{(s - p_1)(s - p_2)}. \quad (4.24)$$

Systems represented by Eq. (4.22) or Eq. (4.24) are said to be non-minimum-phase when the zeros $z, z_{\text{virtual}} > 0$, and minimum-phase when the zeros $z, z_{\text{virtual}} < 0$.

Zeros with positive values are called unstable zeros, while those with negative values are called stable zeros. It is well known that control is more challenging for non-minimum-phase plants [24]. This applies not only to model-based control design but also to controllers that are difficult to model mathematically, such as RL agents or humans [69–71]. Here, as seen from Eq. (4.24), even if the original dynamics $\mathcal{V}_0$ is non-minimum-phase, changing the prediction period T may render $\mathcal{V}_T$ minimum-phase. In this section, we investigate the impact of this effect on learning efficiency.

4.3.3 Controller: Reinforcement Learning with Proximal Policy Optimization

The agent $\mathcal{K}$ observes the altered output $\hat{y}$ as in Eq. (4.19) and produces the control input u as

$$\mathcal{K} : u(t) = \Pi(\hat{y}(t)) \quad (4.25)$$

where $\Pi : \mathbb{R} \rightarrow \mathbb{R}$ is the policy. We employ PPO (Proximal Policy Optimization) [72] for the policy Π . The agent comprises two networks with distinct roles. One is an actor that produces actions, and the other is a critic that estimates state value. Both are fully connected multilayer neural networks with three hidden layers of 512 units each and tanh activation. The input to both networks is the altered output $\hat{y}$. The actor is probabilistic: the network outputs the mean and variance of the control input u . The control input is clipped to $[-1, 1]$. The critic outputs a scalar.

The target state set $\mathbb{X}_{\text{target}}$ that the state x must satisfy is defined as

$$\mathbb{X}_{\text{target}} = \{x \in \mathbb{R}^2 \mid |x_1| \leq 1, |x_2| \leq 1\} \quad (4.26)$$

We set the sampling time to $T_s := 0.1$ s and the episode length to 500 steps (i.e., 50 s). This corresponds to 50 s. A reward of 1.0 is given at each time step, and the episode terminates when any component exceeds its boundary.

The main hyperparameters are as follows: clipping coefficient $\epsilon_{\text{clip}} = 0.2$, discount factor $\gamma = 0.99$, GAE parameter $\lambda = 0.95$, and entropy coefficient $\varepsilon = 1.0 \times 10^{-4}$. We use Adam [73] for optimization, cosine annealing for the learning rate schedule, gradient norm clipping at 1.0, 1000 environment steps per update, 50,000 total training steps, a sub-batch size of 64, and 10 epochs per update.

4.3.4 Numerical Experiments and Results

We systematically evaluate, for the second-order linear system Eq. (4.17), the impact of future-prediction Eq. (4.19) on learning efficiency. Specifically, we prepare multiple plants with different parameter settings and train PPO agents while varying the prediction period T for each plant. We then classify the plants based on two points: (i) whether prediction can replace an unstable zero with a stable zero, and (ii) whether prediction improves learning efficiency.

For the plant poles, we consider combinations (p_1, p_2) expressible as $p_1 = 0.5n_1 + 0.5n_2j$, $p_2 = 0.5n_3 + 0.5n_4j$ with integers n_1, n_2, n_3, n_4 , and select those that:

4.3. DESIGN GUIDELINES VIA MINIMUM-PHASE CONSIDERATIONS

- satisfy $|p_1| \leq 1.5$ and $|p_2| \leq 1.5$,
- are either both real or a complex-conjugate pair,
- and satisfy $\text{Re}(p_1) \leq \text{Re}(p_2)$.

We consider all 14 combinations satisfying these conditions. We set the zero $z \in \{0.05, 0.25, 1.0\}$, thus all $\mathcal{V}_0$ are non-minimum-phase systems. From the resulting 42 combinations, we exclude 3 that are uncontrollable or unobservable, leaving 39 plants for analysis. The prediction period is chosen from $T \in \{0, 0.25, 0.5, 0.75, 1.0\}$ s.

For each of the 39 systems and 5 prediction periods, we train a PPO agent and classify each system as follows.

1. Regarding minimum-phase properties,
 - “Zero Became Stable” if $z_{\text{virtual}}(T) < 0$ for some $T \in \{0.25, 0.5, 0.75, 1.0\}$,
 - “Zero Remained Unstable” otherwise.
2. Regarding learning results,
 - “Failed” if $S(T) = \infty$ for all $T \in \{0, 0.25, 0.5, 0.75, 1.0\}$,
 - “Better w/ Prediction” if there exists $T \in \{0.25, 0.5, 0.75, 1.0\}$ such that $J_{\mathcal{K}}(\mathcal{V}_T) < J_{\mathcal{K}}(\mathcal{V}_0)$,
 - “Better w/o Prediction” otherwise.

Here, if learning fails to reach the target score S_{target} within the total of 50,000 training steps, we treat it as $S(T) = \infty$. The results are summarized in Table 4.1.

Of the 39 systems, 11 were labeled “Zero Became Stable”. All 11 of these were “Better w/ Prediction”. This means that whenever prediction could replace an unstable zero with a stable one, learning efficiency consistently improved. This suggests that transforming dynamics from non-minimum-phase to minimum-phase via an Alterer is a useful design guideline for enhancing RL efficiency.

Among the 28 systems labeled “Zero Remained Unstable”, 8 were “Better w/ Prediction”, 11 were “Better w/o Prediction”, and 9 were “Failed”. Hence, in 8 systems, learning efficiency improved despite prediction not stabilizing the zero. This indicates that learning efficiency can improve even if the Virtual Dynamics remains non-minimum-phase after the insertion of an Alterer.

4.3.5 Discussion

In summary, when prediction can replace an unstable zero with a stable one, learning efficiency consistently improves; even when it cannot, improvements may still occur. This result suggests that replacing unstable zeros with stable ones via Eq. (4.19) is a useful design guide-

4.4. CONCLUSION

Table 4.1: Summary of the effect of future-prediction shaping on learning efficiency for second-order LTI systems

	Zero Became Stable	Zero Remained Unstable
Better w/o Prediction	0	11
Better w/ Prediction	11	8
Failed	0	9

line for Alterers to enhance RL efficiency.

4.4 Conclusion

In this chapter, we addressed improving learning efficiency and final performance in control systems using RL by applying Virtual Dynamics. First, we formulated the design problem of Virtual Dynamics for RL as finding an Alterer that minimizes the time for the task score to reach a target. Next, using the inverted pendulum as an example, we demonstrated that inserting a future-predictive Alterer improves learning efficiency and final performance. Finally, we examined the relationships among the minimum-phase property of a second-order linear plant, that of the plant with an Alterer, and learning efficiency. As a result, we confirmed that whenever the Alterer replaced an unstable zero with a stable one, learning efficiency consistently improved, suggesting a design guideline for Alterers.

Chapter 5

Virtual Dynamics Design for Human-Machine Systems

In this chapter, we apply Problem 2.1, the Virtual Dynamics design problem formulated in Chapter 2, to human-machine systems. Human-machine systems play a crucial role in many real-world systems, but they are also associated with numerous accidents that result in significant property damage and loss of life. Therefore, in this chapter, we aim to design Virtual Dynamics that improve the operability of human-machine systems.

Section 5.1 formulates the design problem of Virtual Dynamics that are easy for humans to operate. Section 5.2 experimentally demonstrates that altering visual information can enhance operability using a system, The Inverted Pendulum Game, developed by the author. Section 5.3 conducts numerical analysis using a linear model of a human-machine system to obtain guidelines for Virtual Dynamics design.

5.1 Design Problem

Automobiles [74–78], heavy machinery [13, 79], aircraft [8, 10, 80], drones [81, 82], and remotely operated robots [83–87] are representative human-machine systems operated based on visual information. In these systems, improving operability — including subjective ease of operation, failure avoidance, and reduced learning time — is crucial.

There are two principal approaches to improve operability:

- Input rectification: rectifying the input generated by the human operator before feeding it to the machine.
- Information presentation: modifying the information presented to the human operator.

The former input rectification has the advantage of directly managing the machine's state

5.1. DESIGN PROBLEM

and is an important approach. For example, control barrier functions that rectify inputs to avoid dangerous states [88, 89] are examples of this approach. However, rectified inputs may contradict the operator’s intentions, causing confusion and reduced trust, which can lead to accidents in the worst cases. The latter information presentation approach does not contradict the human operator’s intentions and is an important complement to input rectification. In this research, we focus on this approach to information presentation.

A common research topic in information presentation is to devise ways to present information more accurately and comprehensively. For example, position/target scaling [90] investigates the impact of information presentation methods on operability. Furthermore, rear parking cameras [91, 92] and AR for blind spots [93] improve operability by providing wider and more accurate information. Predictive delayed displays [6] improve operability by compensating for prediction in environments with delays. Moreover, devising user interfaces, such as relative targeting interfaces [94], motion cueing and multisensory feedback [95–98], haptic excavator assistance [99, 100], and natural user interfaces [81, 101], is also an important aspect. However, systems like inverted pendulums remain difficult to operate due to limitations in reaction speed and sensory accuracy, even with accurate and comprehensive information presentation.

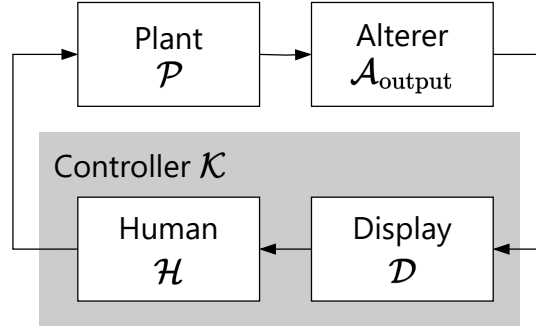


Fig. 5.1: Block diagram of a control system where a human operator serves as the controller

We therefore introduce alteration of visual information. Figure 5.1 shows the system structure considered in this chapter. In the figure, $\mathcal{P} \in \mathbb{S}$ is the plant, $\mathcal{D} \in \mathbb{S}_{\text{static}}$ is the display that converts the machine’s state into an image, and $\mathcal{H} \in \mathbb{S}$ is the human operator. Then, we regard $\mathcal{K} := \mathcal{H} \circ \mathcal{D}$ as the controller. Moreover, the Alterer $\mathcal{A}_{\text{output}} \in \mathbb{S}$ is a system that alters the machine’s state before image generation. Such alteration of visual information has been studied in various contexts, including overlay displays of parking trajectories [92], lead-computing sights that show bullet trajectory predictions [8, 9], projection transformations for remote driving [75, 76], and displays that predict opponents’ movements in martial arts [102]. However, to the best of our knowledge, the effectiveness of state alteration for stabilization tasks remains unclear. Therefore, we verify the effectiveness of the Alterer in stabilization

5.2. THE INVERTED PENDULUM GAME: A PIONEERING EXAMPLE OF OPERABILITY ENHANCEMENT VIA VIRTUAL DYNAMICS

tasks.

We define the evaluation function $J_{\mathcal{K}}$ such that better operability results in a smaller value. Specifically, let the score of a task be $S \in \mathbb{R}$. Then we define

$$J_{\mathcal{K}}(\mathcal{A}_{\text{output}} \circ \mathcal{P}) = -S. \quad (5.1)$$

Using this evaluation function, we consider the following design problem.

Problem 5.1 Given a plant $\mathcal{P} \in \mathbb{S}$, a controller $\mathcal{K} = \mathcal{H} \circ \mathcal{D}$ consisting of a human operator $\mathcal{H} \in \mathbb{S}$ and a display $\mathcal{D} \in \mathbb{S}_{\text{static}}$, design an Alterer $\mathcal{A}_{\text{output}} \in \mathbb{S}$ that minimizes the evaluation function $J_{\mathcal{K}}$ defined in Eq. (5.1).

5.2 The Inverted Pendulum Game: A Pioneering Example of Operability Enhancement via Virtual Dynamics

It is not self-evident whether an Alterer $\mathcal{A}_{\text{output}}$ exists that can truly enhance operability. Therefore, we developed a system named The Inverted Pendulum Game and experimentally demonstrated that altering visual information can enhance operability.

5.2.1 The Inverted Pendulum Game

This section describes The Inverted Pendulum Game, which I developed to verify the effectiveness of the Alterer.

Figure 5.2 shows the game screen. After the experiment described later, we published almost the same implementation of The Inverted Pendulum Game at https://knttnk.github.io/pendulum_game_web/. Interested readers can play the game from this link. Playing the game provides a clear understanding of its difficulty. Therefore, we highly recommend experiencing it firsthand.

The dynamics of the inverted pendulum are given by Eq. (4.3). The game starts with the initial state $x(0) = [0.01, 0.01, 0, 0]^T$. The player can input $u(t) \in [-3, 3]$ with a joystick.

In this game, the player aims to manually stabilize the inverted pendulum displayed in the center of the screen. The score of a performance is defined as the time it takes from the start of the performance until $|\theta|$ reaches $\pi/4$, which results in a game over. The goal of this game is to keep the pendulum upright for as long as possible, so the higher the score, the better the operation. The game screen displays the range $-p_{\max} < p < p_{\max}$. Here, $p_{\max} \in \mathbb{R}_{>0}$ is a constant. The game does not end even if the cart moves out of this range, but the player cannot grasp the system's state. Therefore, it is practically necessary to play within this range.

5.2. THE INVERTED PENDULUM GAME: A PIONEERING EXAMPLE OF OPERABILITY ENHANCEMENT VIA VIRTUAL DYNAMICS

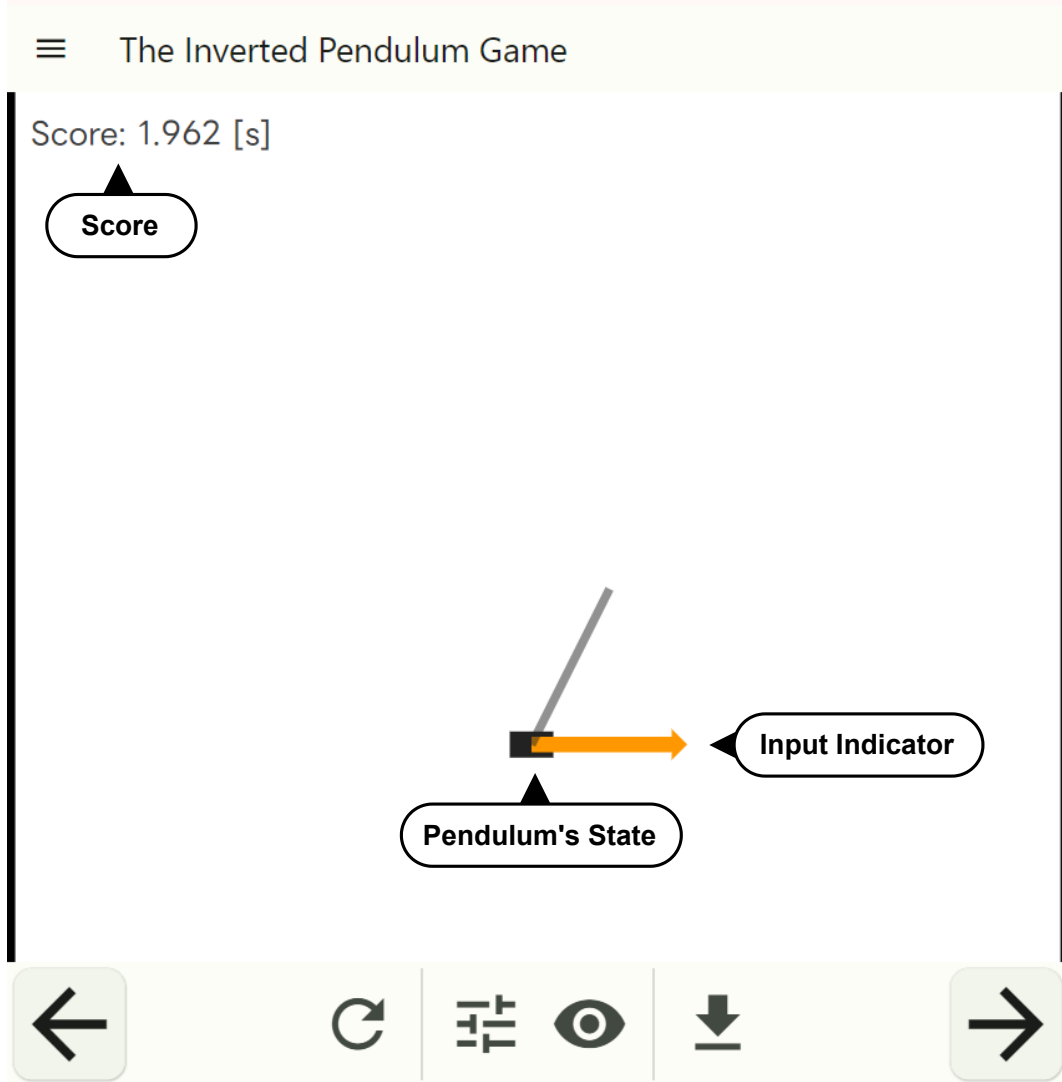


Fig. 5.2: Game screen of The Inverted Pendulum Game

5.2.2 Alteration: Prediction Based on the Current State

The Inverted Pendulum Game has two modes: one that presents the machine's state as it is and one that presents the altered state. In this study, we adopted future prediction as one of the information alterations that may enhance operability. This section describes the specific method.

We first define $\hat{x}(\tau; t) \in \mathbb{R}^4$ as the value at τ of the solution of the partial differential equation

$$\frac{\partial}{\partial \tau} \hat{x}(\tau; t) = f(\hat{x}(\tau; t)), \quad (5.2)$$

5.2. THE INVERTED PENDULUM GAME: A PIONEERING EXAMPLE OF OPERABILITY ENHANCEMENT VIA VIRTUAL DYNAMICS

with the initial value $\hat{x}(0; t) = x(t)$. In this way, $\hat{x}(T; t)$ represents the state after time T has passed from state $x(t)$ in Eq. (4.3) with $u(t) = 0$, serving as a prediction of the future state. Next, we denote the first component of $\hat{x}(\tau; t)$ as $\hat{p}_\tau$ and the second component of it as $\hat{\theta}_\tau$. Given the above, The Inverted Pendulum Game displays altered visual information in each frame as follows.

1. Calculate $\hat{x}(T; t)$ numerically.
2. Generate an image of an inverted pendulum with position $\hat{p}_T$ and angle $\hat{\theta}_T$ and display it on the game screen.

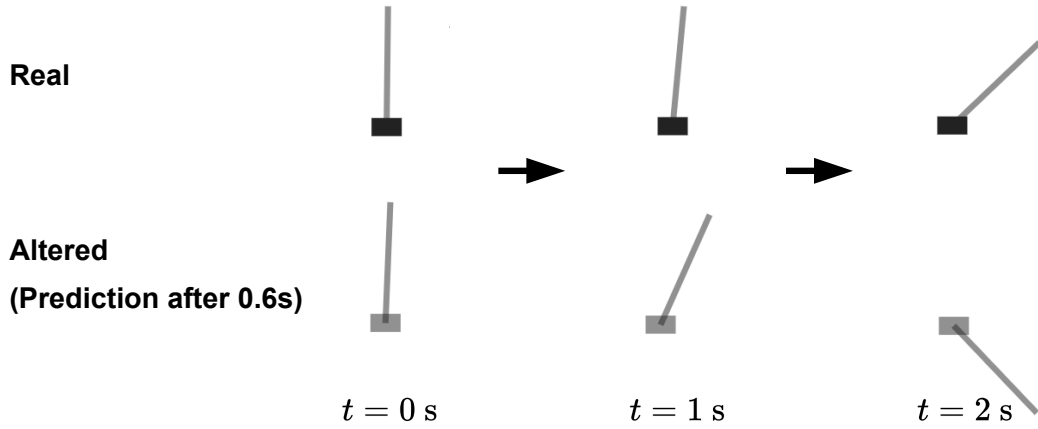


Fig. 5.3: Illustration of state alteration in The Inverted Pendulum Game

Figure 5.3 illustrates how the real and predicted states are displayed on the game screen. In the figure, the Real side shows how the inverted pendulum falls when $u(t) = 0$. The Altered side displays the predicted state; the pendulum appears to be tilted more than in the Real side at the same time.

5.2.3 Experimental Method

Figure 5.4 shows a photo of an experiment. This experiment was conducted with the approval of the Ethics Committee on Research Involving Human Subjects at the School of Engineering, Osaka University (Approval number: 6–1–1). Written informed consent was obtained from the subjects. Each experiment targeted one subject. At the beginning of the experiment, we explained the following to the subject.

- The subject will play The Inverted Pendulum Game.
- The score of The Inverted Pendulum Game is the time until the bar falls.
- The higher the score, the more reward the subject will receive after the experiment.

5.2. THE INVERTED PENDULUM GAME: A PIONEERING EXAMPLE OF OPERABILITY ENHANCEMENT VIA VIRTUAL DYNAMICS

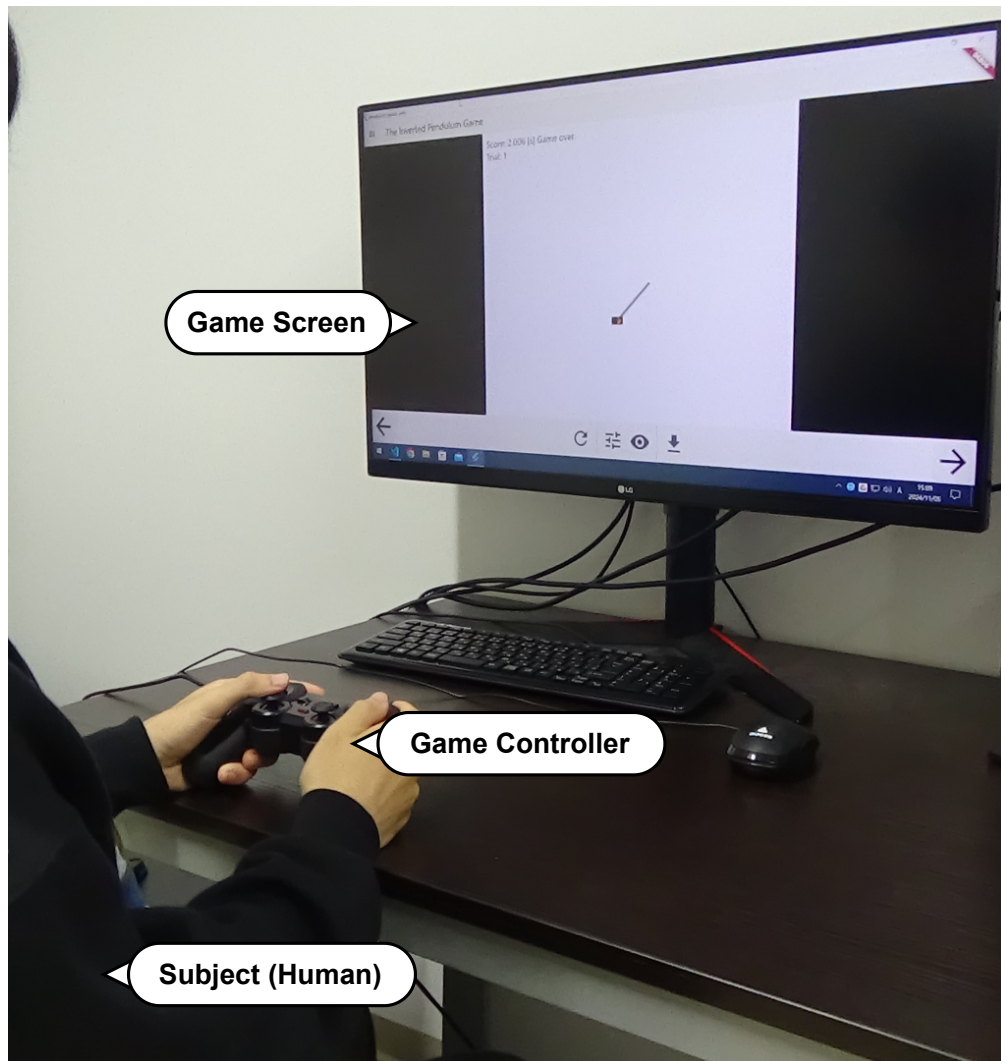


Fig. 5.4: Photo of an experiment using The Inverted Pendulum Game

(We did not disclose the specific calculation method to the subject.)

- The subject can stabilize the inverted pendulum by applying force to the cart with the joystick.

Furthermore, we explained the following strategies for The Inverted Pendulum Game.

- The behavior is similar to a bar placed on the palm.
- If the bar is likely to fall to the right, applying force to the right may prevent it from falling, and vice versa.

After that, the experimenter did not give any advice on the strategy of The Inverted Pendulum Game.

The experimental procedure is as follows.

5.2. THE INVERTED PENDULUM GAME: A PIONEERING EXAMPLE OF OPERABILITY ENHANCEMENT VIA VIRTUAL DYNAMICS

- (i) Randomly assign the subject to either the group that shows real information first (hereinafter referred to as group R) or the group that shows altered information first (hereinafter referred to as group A).
- (ii) Set the game to show real information if the subject belongs to group R and altered information if the subject belongs to group A.
- (iii) Let the subject practice The Inverted Pendulum Game for 5 min.
- (iv) Encourage the subject to take a break for an arbitrary amount of time.
- (v) Let the subject play 10 performances.
- (vi) Encourage the subject to take a break for an arbitrary amount of time.
- (vii) Switch the setting of whether to show real or altered information and repeat (iii) to (v).
- (viii) Let the subject answer a questionnaire.

Note that we did not tell the subjects what each presentation represents (e.g., real or altered), but only that the two presentations are different. Furthermore, if a subject did not experience a game over within 2 min of starting a performance, the performance was terminated, and the score was recorded as 120 s. This was done to prevent the experiment from being unnecessarily prolonged and increasing the burden on the subjects, as it is expected that skilled subjects can keep the pendulum upright for quite a long time.

In each experiment, we recorded the subject's input u , the position p , the angle θ , and the video of the subject's hands and the screen.

The experiment targeted 35 subjects. However, three subjects were deemed invalid due to procedural errors. Specifically, the value of $p_{\max}$ was incorrectly set for two subjects, and one subject completed only 19 of the intended 20 performances. The 32 valid subjects ranged in age from late teens to early thirties, comprising 22 males and 10 females. The experiment was conducted in Japanese. Thus, the subjects were required to be able to speak Japanese in daily conversation.

The values of the parameters of The Inverted Pendulum Game were set to $g = 9.81 \text{ m/s}^2$, $m = 0.5 \text{ kg}$, $m_s = 1 \text{ kg}$, $l = 2 \text{ m}$, $D_p = 0.2 \text{ N} \cdot \text{s/m}$, $D_\theta = 0.06 \text{ N} \cdot \text{m} \cdot \text{s/rad}$, $p_{\max} = 6 \text{ m}$, and $T = 0.6 \text{ s}$. The frame rate was 60 fps. The value of T was determined through trial and error in a preliminary experiment conducted by the game's developer (the first author). All length-related parameters, such as l and p , were set so that 1 m was displayed as 0.023 m on the screen. The distance between each subject's eyes and the display was adjusted by the subject at their discretion, resulting in distances ranging from 0.5 to 1.0 m.

5.2.4 Experimental Results: Example of Time Evolution

In this section, we show how the input u , position p , and angle θ evolve in two performances of one subject as an example.

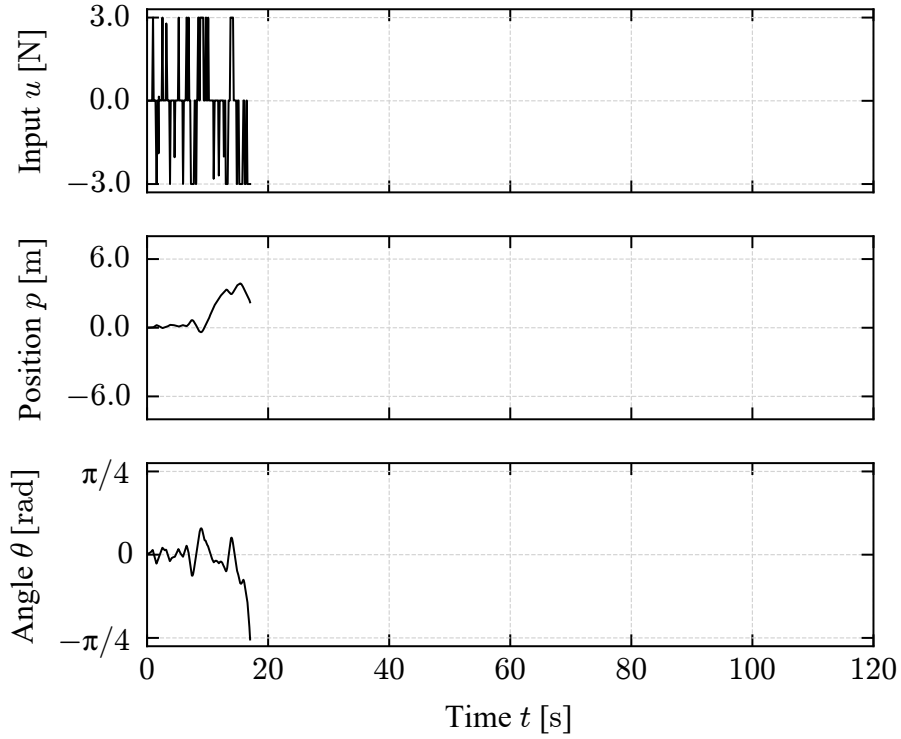


Fig. 5.5: Time evolution of input, position, and angle in a performance in which one subject is shown real visual information

Figure 5.5 shows the time evolution of input, position, and angle when one subject is shown real visual information. And, Figure 5.6 shows one when the same subject is shown altered visual information. We can see that the score was 17.0 s in the performance shown in Figure 5.5 and 82.3 s in the performance shown in Figure 5.6.

5.2.5 Experimental Results: Comparison and Analysis of Scores

In this section, we compare and analyze the scores when information was altered and when real information was presented.

First, Figure 5.7 shows the scores of 32 subjects. The vertical and horizontal axes indicate the scores when altered and real information was presented, respectively. Each marker corresponds to the median calculated from the scores of 10 performances for each subject, while

5.2. THE INVERTED PENDULUM GAME: A PIONEERING EXAMPLE OF OPERABILITY ENHANCEMENT VIA VIRTUAL DYNAMICS

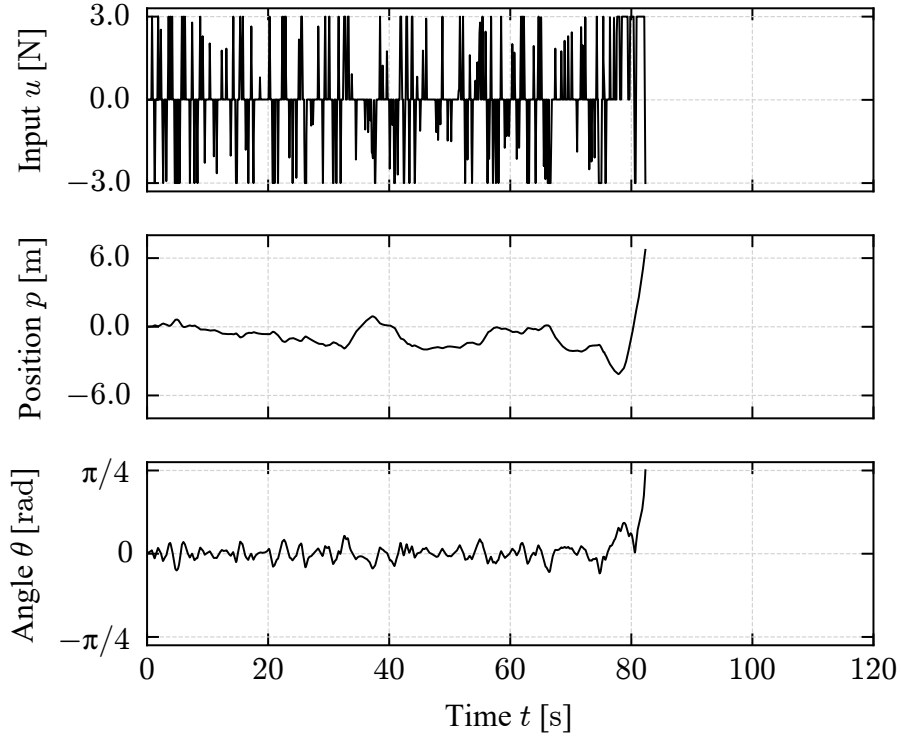


Fig. 5.6: Time evolution of input, position, and angle in a performance in which the same subject as in Fig. 5.5 is shown altered visual information

the error bars represent the interquartile range of these scores.

From this, we can see that the median is more than twice as large when altered information is presented for all subjects. We additionally performed the one-sided Wilcoxon signed-rank test and found that the median score is significantly higher when altered information is presented, with a p -value of 2.3×10^{-10} . Note that the scores are limited to 120 s, but it is worth noting that only the performances in which altered information is presented are limited by this restriction. From these facts, we can confirm that presenting altered visual information improves the score of The Inverted Pendulum Game. Furthermore, we can see that showing altered information results in a higher score in both group R and group A. The one-sided Wilcoxon signed-rank test supported this with p -values of 1.2×10^{-4} and 1.9×10^{-6} , respectively. Therefore, it is clear that showing altered information improves the score, regardless of whether it is presented first.

5.2.6 Experimental Results: Comparison of Input Cost

In this section, we define input cost to evaluate how the subject's input changes when altered information is presented compared to when real information is presented. We then calculate

5.2. THE INVERTED PENDULUM GAME: A PIONEERING EXAMPLE OF OPERABILITY ENHANCEMENT VIA VIRTUAL DYNAMICS

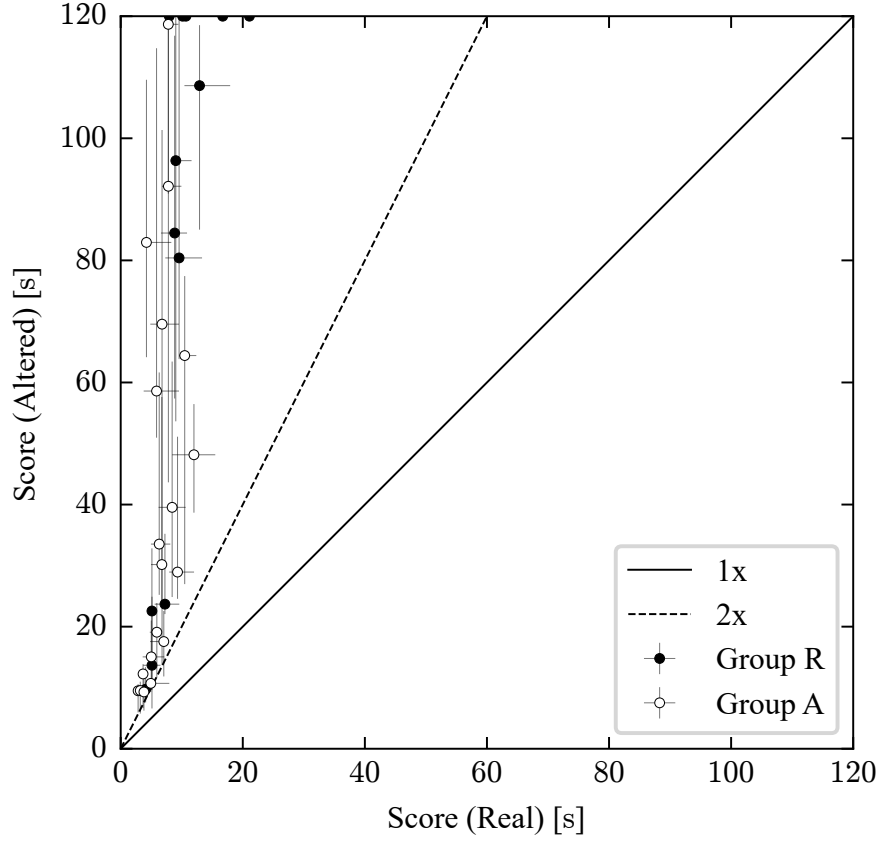


Fig. 5.7: Comparison of scores when real and altered information was presented

and compare the cost.

We define the input cost (real) C_{real} as

$$C_{\text{real}} = \frac{\sum_{i=1}^{10} \int_0^{T_{\text{real}(i)}} u_{\text{real}(i)}(t)^2 dt}{\sum_{i=1}^{10} T_{\text{real}(i)}}. \quad (5.3)$$

Here, for the i -th performance in which the subject was presented with real information, $T_{\text{real}(i)} \in \mathbb{R}$ is the performance score, and $u_{\text{real}(i)}(t) \in \mathbb{R}$ is the input u at time t after the start of the performance. The input cost (real) C_{real} is the time average of the square of the input u , so it becomes larger as the duration for which u^2 is large is longer. Similarly, we define the input cost C_{altered} for the performances in which the subject was presented with altered information.

We calculated the input costs C_{real} and C_{altered} for 32 subjects and plotted them in Figure 5.8. From this, we can see that C_{altered} is smaller than C_{real} for all subjects except two. Therefore, it is clear that presenting altered information reduces the input cost.

5.2. THE INVERTED PENDULUM GAME: A PIONEERING EXAMPLE OF OPERABILITY ENHANCEMENT VIA VIRTUAL DYNAMICS

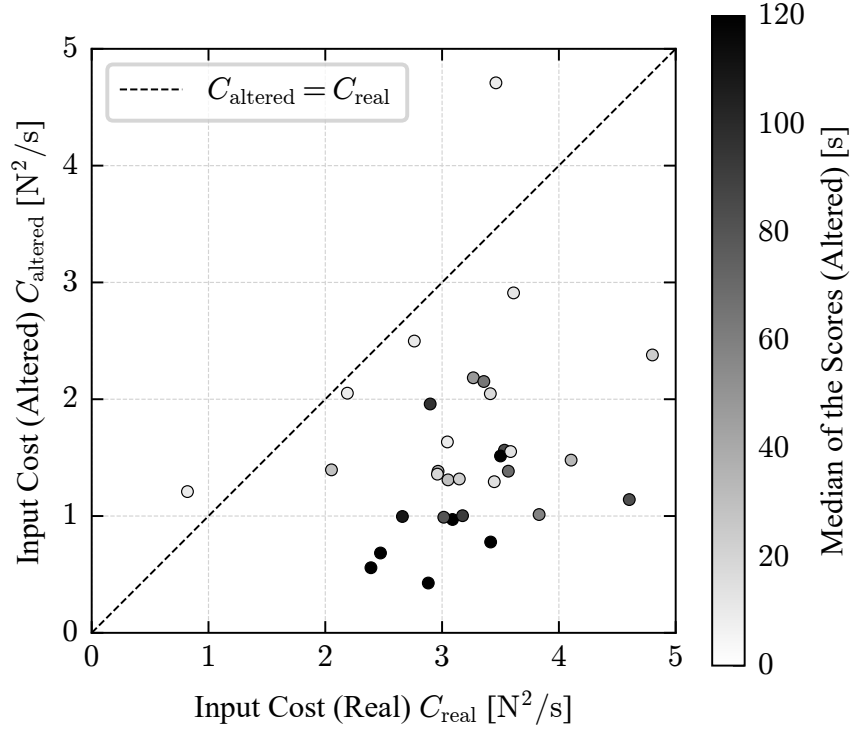


Fig. 5.8: Comparison of input costs when presented with real versus altered information

5.2.7 Experimental Results: Comparison of Angle Cost

In this section, we define angle cost to evaluate how the pendulum's angle changes when altered information is presented compared to when real information is presented. We then calculate and compare the cost.

We define the angle cost (real) Θ_{real} as

$$\Theta_{\text{real}} = \frac{\sum_{i=1}^{10} \int_0^{T_{\text{real}(i)}} \theta_{\text{real}(i)}(t)^2 dt}{\sum_{i=1}^{10} T_{\text{real}(i)}}. \quad (5.4)$$

Here, for the i -th performance, $\theta_{\text{real}(i)}(t) \in \mathbb{R}$ is the angle of the pendulum θ after time t from the start of that performance. This is obtained by replacing u in Eq. (5.3) with θ . Thus, Θ_{real} is the time average of the square of the angle θ , so it becomes larger as the duration for which θ^2 is large is longer. Similarly, we define the angle cost Θ_{altered} for the performances in which the subject was presented with altered information.

We calculated the angle costs Θ_{real} and Θ_{altered} for 32 subjects and plotted them in Fig. 5.9. From this, we can see that Θ_{altered} is smaller than Θ_{real} for all subjects. In brief, we can confirm that presenting altered information allows the angle to be small for a longer time. There-

5.2. THE INVERTED PENDULUM GAME: A PIONEERING EXAMPLE OF OPERABILITY ENHANCEMENT VIA VIRTUAL DYNAMICS

fore, it is considered that presenting altered information improves operability. Furthermore, we can see that subjects with a higher median score when altered information was presented (represented by black markers) tend to have a smaller value of $\Theta_{\text{altered}}/\Theta_{\text{real}}$. We can see similar phenomena in Figure 5.8. These suggest that altered information may be more effective for subjects accustomed to the operation.

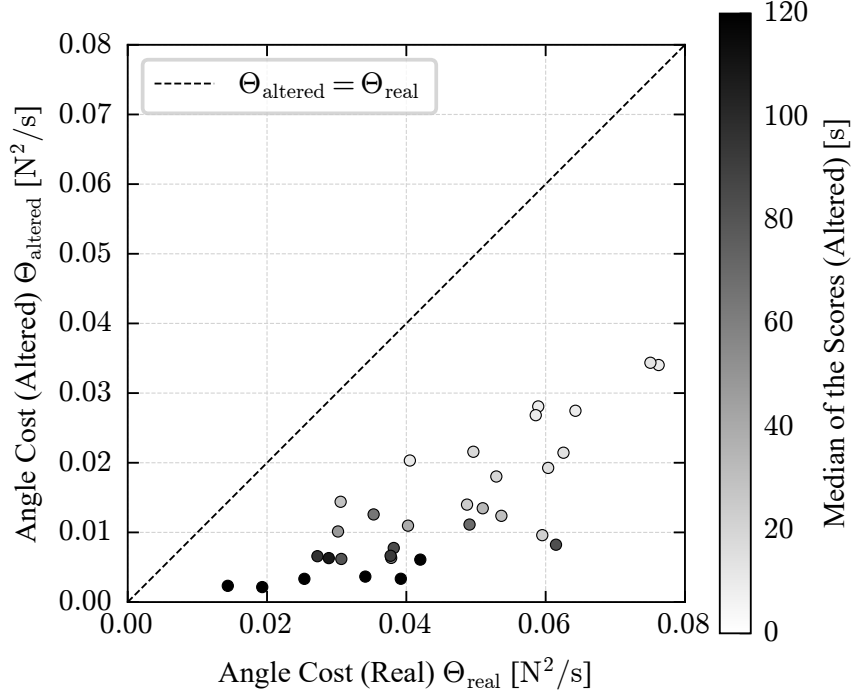


Fig. 5.9: Comparison of angle costs when presented with real versus altered information

5.2.8 Questionnaire Results

The first question was: “Which was easier to operate”? This question was answered on a 5-point scale, with “1” for “when presented with altered information” and “5” for “when presented with real information”. The answers to this question were “1” for 30 subjects, “2” for 2 subjects, and no other answers, indicating that all subjects found it easier to operate when presented with altered information. The one-sample Wilcoxon signed-rank test showed that the answers were significantly lower than “3” with a p -value of 1.8×10^{-8} . From these facts, we can see that the subjective operational feeling is better when altered information is presented.

The second question was: “If you have any other comments about the game, please write them down”. Excerpts from the answers that mention differences between presenting real and altered information are as follows. These are all translated from Japanese to English.

5.3. DESIGN GUIDELINES FOR VIRTUAL DYNAMICS TO ENHANCE HUMAN-MACHINE OPERABILITY

- When presented with real information, the angle at which the bar falls and the slipperiness of the cart seemed to be more severe.
- When presented with altered information, the allowable range of inclination was wider and easier to operate.
- When presented with altered information, both large and fine adjustments were possible, but when presented with real information, only fine adjustments were possible, making it difficult to maintain.
- When presented with real information, I felt that the angle at which the game was over was small, and the bar was more likely to fall. I felt that direction correction was often too late and that a delicate operation was required.
- When presented with real information, I found the movement heavy.

From these answers, we can see that presenting altered information is preferred concerning operational feeling. Furthermore, there are at least two reasons why the score improved. The first reason is that the allowable range of inclination felt wider due to prediction. The second reason is that an effect made the subjects feel the movement of the bar was light.

5.3 Design Guidelines for Virtual Dynamics to Enhance Human-Machine Operability

In Section 5.2, using the Inverted Pendulum Game as an example, we experimentally demonstrated that Virtual Dynamics can enhance operability in human-machine systems. However, the principles by which Virtual Dynamics improve operability and design methods for $\mathcal{A}_{\text{output}}$ required to solve Problem 2.1 remain unclear. To obtain design guidelines, we analyze linearized models approximating each system.

5.3.1 Alterer: Linear Future Prediction

We consider the human-machine system shown in Fig. 5.10, composed of a human operator $\mathcal{H}$, an inverted pendulum $\mathcal{P}$, an information Alterer $\mathcal{A}_{\text{output}}$, and a display $\mathcal{D}$. The inverted pendulum $\mathcal{P} \in \mathbb{S}$ is a dynamical system represented by the linearized cart-pendulum model in Eq. (4.7). The display $\mathcal{D} \in \mathbb{S}_{\text{static}}$ is a static system that generates and outputs an image y based on the virtual state $\bar{x}$.

5.3. DESIGN GUIDELINES FOR VIRTUAL DYNAMICS TO ENHANCE HUMAN-MACHINE OPERABILITY

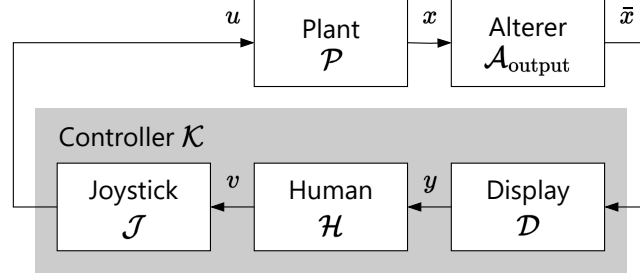


Fig. 5.10: Block diagram of the Inverted Pendulum Game model to investigate factors for operability improvement

5.3.2 Models of the Inverted Pendulum and the Information Alterer

The Alterer in Eq. (5.2) uses the nonlinear pendulum model to predict the state T seconds ahead under $u = 0$. Here, we consider an Alterer using the linearized cart-pendulum model in Eq. (4.7) as

$$\mathcal{A}_{\text{output}} : \bar{x}(t) = e^{A_{\text{lin}} T} x(t). \quad (5.5)$$

5.3.3 Controller: Static Output Feedback

In this section, we describe the models of the display $\mathcal{D}$ and the human operator $\mathcal{H}$.

For the display $\mathcal{D}$, we assume the following model. To render the image $y(t)$ at time t , only the cart position $p(t)$ and the pendulum angle $\theta(t)$ are required from the state $x(t)$; the cart velocity $\dot{p}(t)$ and the pendulum angular velocity $\dot{\theta}(t)$ are unnecessary. Thus, the display $\mathcal{D}$ can be regarded as a static map extracting part of the machine state $x(t)$, namely

$$\mathcal{D} : y(t) = Cx(t), \quad C := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \quad (5.6)$$

Therefore, not all states are observable to the human, leading to output-feedback control.

Next, we model the human operator $\mathcal{H}$ using a constant feedback gain. Since the display $\mathcal{D}$ outputs part of the machine state x , the observable information y for the human consists only of the cart position and pendulum angle. Thus, to utilize the cart velocity $\dot{p}$ and pendulum angular velocity $\dot{\theta}$ when generating the control input u , the human must estimate them from past information. If stabilization is possible with constant feedback gain, stabilization can be achieved without estimating $\dot{p}$ and $\dot{\theta}$. This may lead to improved operability. Therefore, in

5.3. DESIGN GUIDELINES FOR VIRTUAL DYNAMICS TO ENHANCE HUMAN-MACHINE OPERABILITY

this paper, we assume that the human control law is a static output feedback

$$u(t) = Ky(t) \quad (5.7)$$

using a constant feedback gain $K = [k_p \ k_\theta] \in \mathbb{R}^2$ and investigate stabilizability. Note that k_p is the proportional gain for the cart position in the generated image, and k_θ is the proportional gain for the pendulum angle.

From the above, the closed-loop model of the Inverted Pendulum Game becomes

$$\dot{x}(t) = \tilde{A}x(t) \quad (5.8)$$

$$\tilde{A} := A + BKCe^{AT} \quad (5.9)$$

Even if $\tilde{A} = A + BKC$ ($T = 0$) is unstable, $\tilde{A} = A + BKCe^{AT}$ may become stable for $T > 0$. This suggests that the stabilizability under the gain K depends on T . In the next section, we order the four eigenvalues of $\tilde{A}$ by ascending real part as $\lambda_1, \lambda_2, \lambda_3$, and λ_4 and study how they vary with the pair (K, T) .

5.3.4 Numerical Example

Parameters were set to $g = 9.81 \text{ m/s}^2$, $m = 0.5 \text{ kg}$, $m_s = 1 \text{ kg}$, $l = 2 \text{ m}$, $D_p = 0.2 \text{ N} \cdot \text{s/m}$, and $D_\theta = 0.06 \text{ N} \cdot \text{m} \cdot \text{s/rad}$.

To fix $k_p = 0.01 \text{ N/m}$, we computed the four eigenvalues of $\tilde{A}$ as we varied T and k_θ . Plotting T on the vertical axis and k_θ on the horizontal axis, with blue for negative real parts and red for non-negative real parts, yields Fig. 5.12 (a). From this, we see that λ_1 and λ_2 are stable regardless of how T and k_θ vary within the range. On the other hand, for λ_3 and λ_4 , at $T = 0$, one cannot make their real parts negative for any choice of k_θ . However, for $T > 0$, i.e., when some prediction is made, selecting k_θ appropriately can make the real parts of λ_3 and λ_4 negative.

Next, Fig. 5.12 (a) summarizes the stability of the four poles. In each figure, blue indicates the region where the real parts of all four poles are negative, and red indicates the region where at least one real part is non-negative. In other words, the closed-loop system is stable in the blue region and unstable in the red region. From these figures, for example, at $T = 0.6 \text{ s}$ used in Section 5.2, selecting $k_\theta = 15 \text{ N/rad}$ stabilizes the system. Thus, inserting a prediction-based Alterer enables stabilization via an input proportional to both the displayed cart position and the pendulum angle.

A similar plot with $k_p = 1 \text{ N/m}$ fixed is shown in Fig. 5.12 (b). It exhibits trends similar to Fig. 5.12 (a). Thus, the tendency that $\tilde{A}$ is not stabilizable by any k_θ without prediction, yet becomes stabilizable with moderate prediction, was observed for $k_p \in [0.01, 1]$.

5.3. DESIGN GUIDELINES FOR VIRTUAL DYNAMICS TO ENHANCE HUMAN-MACHINE OPERABILITY

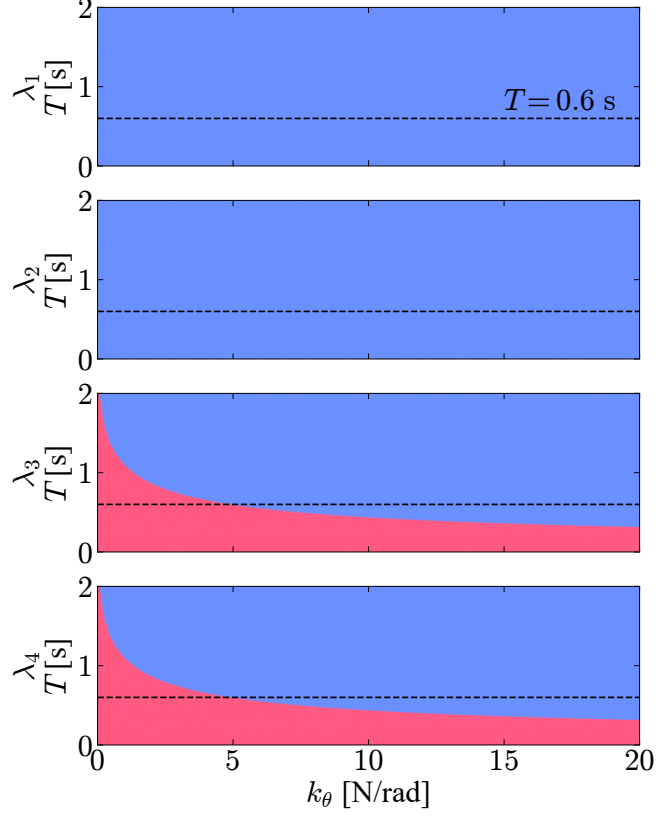


Fig. 5.11: Stability of the eigenvalues for each T and k_θ when $K = [0.01 \ k_\theta]$

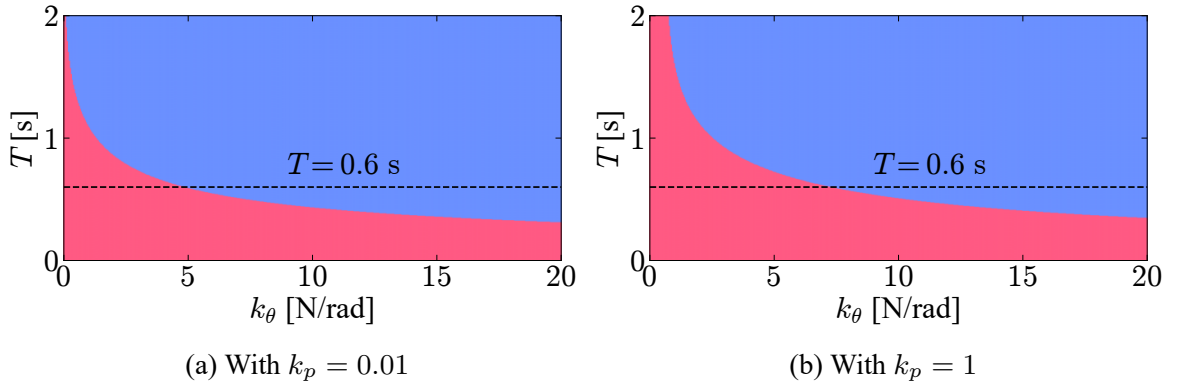


Fig. 5.12: Stability of $\tilde{A}$ for each T and k_θ

5.4 Conclusion

In this chapter, we tackled the design of Virtual Dynamics to enhance operability in human-machine systems. First, we formulated the design problem of Virtual Dynamics for human-machine systems as maximizing the median task score. Next, using the Inverted Pendulum Game as an example, we experimentally demonstrated that Virtual Dynamics can enhance operability in human-machine systems. Finally, we analyzed linearized models approximating each system and showed that inserting a prediction-based Alterer enables stabilization via an input proportional to both the displayed cart position and the pendulum angle. This suggests that Virtual Dynamics may enhance operability in human-machine systems by enabling stabilization through static output-feedback control.

Chapter 6

Conclusion

This chapter summarizes the contributions of this dissertation and discusses future work.

The central theme of this dissertation is a design methodology for improving the overall performance of a control system even when the controller is fixed and cannot be redesigned. To this end, we have introduced systems that shape signals (Alterers) before and after the plant, regarding the series interconnection Alterer-plant-Alterer as a new plant. Then, by constructing Virtual Dynamics that is easier for the controller to handle, we have aimed to improve the overall performance of the control system.

This dissertation clarified three points: i) what dynamics are easier for the controller to handle, ii) how a given Alterer induces specific Virtual Dynamics, and iii) whether Virtual Dynamics designed in this way can actually improve control performance. Building on these insights, the objective of this dissertation is to establish a design theory for Virtual Dynamics.

An important feature of this framework is that the controller is not included in the Virtual Dynamics, enabling principled discussion even when the controller is effectively a black box. Moreover, this framework is important in that it can provide insights that cannot be obtained from the perspective of adding new functionalities to the controller.

In Chapter 2, we have introduced an evaluation function for Virtual Dynamics and formulated the design problem. The key idea is that minimizing the value of the evaluation function allows us to select and construct Virtual Dynamics that is favorable for the controller.

In Chapter 3, we have positioned dynamic quantizers as input-side Alterers in discrete-valued input control and studied a design problem that minimizes discrepancy from an ideal continuous-valued input system. Under implementability constraints on stability, gain, and order, we have organized multiple design methods — analytic solutions, linear-programming relaxations, and nonlinear optimization — and provided practical pathways to implementable designs. We also developed the open-source Python library NQLib, providing an integrated environment for design, evaluation, and simulation. This has provided one tool for designing

Virtual Dynamics.

In Chapter 4, we have introduced an output-side Alterer that shapes the observed states presented to the agent in reinforcement learning control, aiming to improve learning efficiency and final performance. For an inverted-pendulum stabilization task, we have shown that prediction-type Alterers based on local linear models accelerate learning and can also improve final performance. Moreover, systematic evaluations on second-order linear plants have confirmed that when prediction replaces an unstable zero with a stable zero, learning efficiency consistently improves, suggesting that minimum-phase characteristics serve as a design guideline.

In Chapter 5, we have introduced an output-side Alterer that shapes and presents visual information in human-machine systems, aiming to improve operability. We have constructed The Inverted Pendulum Game and experimentally demonstrated that a prediction-based Alterer presenting a 0.6-second-ahead state prediction can substantially increase scores. In comparative experiments with 32 valid subjects, the median score exceeded that of the real-presentation mode by more than a factor of two for all subjects, while both input and angle costs decreased. Furthermore, analysis assuming a linearized model and static output feedback suggested that enabling stabilization via static output feedback control can serve as a design guideline for Alterers.

Overall, this dissertation has proposed signal-shaping-based Virtual Dynamics design as a cross-domain framework for performance improvement without altering a fixed controller, and demonstrated its effectiveness in control systems involving quantization, reinforcement learning, and human operation.

A first direction for future work is to build a theoretical foundation for applying Virtual Dynamics to a wider variety of control targets. In this dissertation, we have focused on specific cases targeting systems such as linear systems with discrete-valued input constraints, inverted pendulums, and second-order linear systems. Future work aims to construct a design theory for Virtual Dynamics while extending it to more general systems, such as those including friction and saturation nonlinearities, as well as higher-order linear systems.

Another direction for future work is to apply Virtual Dynamics to systems controlled in the real world by a human or reinforcement learning. By applying Virtual Dynamics to the operation and autopilot of airplanes, drones, and robots, it may be possible to reduce the risk of accidents and failures, as well as lower implementation costs. Furthermore, Virtual Dynamics design may be applicable to fields such as image and video recognition and natural language processing in machine learning, as well as physical activities like rehabilitation and sports for humans.

A

Appendix A

This appendix provides the documentation of NQLib described in Chapter 3.

A.1 Documentation of NQLib

NQLib is a Python library to design noise shaping quantizer for discrete-valued input control.

A.1.1 What Can I Do with NQLib?

In the real world, a dynamic system may have to be controlled by discrete-valued signals due to the inclusion of actuators that are driven by ON/OFF or network with capacity limitations. In such a case, a good output may be obtained by converting continuous-valued input to discrete-valued input with a quantizer designed by NQLib.

A.1.2 Documentation

All the documentation is available at <https://knttnk.github.io/NQLib/>.

Examples of usage are available at <https://colab.research.google.com/drive/1Ui-XqaTZCjwqRXC3ZeMeMCbPqGxK9YX0>.

A.1.3 References

NQLib is a Python library version of ODQ Toolbox, which were developed in MATLAB.

- <https://github.com/rmorita-jp/odqtoolbox>
- <https://www.mathworks.com/products/matlab.html>

The algorithms used in NQLib are based on the following paper.

A.2. CLASSES

1. S. Azuma and T. Sugie: Synthesis of optimal dynamic quantizers for discrete-valued input control; IEEE Transactions on Automatic Control, Vol. 53, pp. 2064 – 2075 (2008)
2. S. Azuma, Y. Minami and T. Sugie: Optimal dynamic quantizers for feedback control with discrete-level actuators; Journal of Dynamic Systems, Measurement, and Control, Vol. 133, No. 2, 021005 (2011)
3. Y. Minami and K. Kashima: Synthesis of dynamic quantizers based on series decomposition of systems; Trans. of SICE, Vol. 52, pp. 46 – 51(2016) (in Japanese)
4. R. Morita, S. Azuma, Y. Minami and T. Sugie: Graphical design software for dynamic quantizers in control systems; SICE Journal of Control, Measurement, and System Integration, Vol. 4, No. 5, pp. 372-379 (2011)
5. Y. Minami and T. Muromaki: Differential evolution-based synthesis of dynamic quantizers with fixed-structures; International Journal of Computational Intelligence and Applications, Vol. 15, No. 2, 1650008 (2016)

A.1.4 License

This software is released under the MIT License, see LICENSE.txt.

A.2 Classes

A.2.1 nqlib.Controller

State-space model of a controller K . The controller K is given by:

$$K: \begin{cases} x(t+1) &= Ax(t) + B_1r(t) + B_2y(t) \\ u(t) &= Cx(t) + D_1r(t) + D_2y(t) \end{cases} \quad (\text{A.1})$$

Example

```
import nqlib
K = nqlib.Controller(A=0,
                    B1=0,
                    B2=[0, 0],
                    C=0,
                    D1=1,
                    D2=[-20, -3])
K.A
```

A.2. CLASSES

```
# array([[0]])
```

Methods

■ `__init__(A, B1, B2, C, D1, D2)` Initialize a Controller instance.

Parameters:

- **A** (*array_like*) – State matrix ($n \times n$), real.
- **B1** (*array_like*) – Input matrix for r ($n \times l$), real.
- **B2** (*array_like*) – Input matrix for y ($n \times p$), real.
- **C** (*array_like*) – Output matrix ($m \times n$), real.
- **D1** (*array_like*) – Feedthrough matrix for r ($m \times l$), real.
- **D2** (*array_like*) – Feedthrough matrix for y ($m \times p$), real.

A.2.2 nqlib.DynamicQuantizer

Dynamic quantizer.

Example

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(1.0)
Q = nqlib.DynamicQuantizer(0.6, 1, 1, q)
Q.quantize([0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4])
# array([[ 0.,  0.,  0.,  0.,  0., -1., -2., -3.]])
```

Properties

- **A**: State matrix ($N \times N$, real). $N \geq 1$.
- **B**: Input matrix ($N \times m$, real). $m \geq 1$.
- **C**: Output matrix ($m \times N$, real).
- **N**: Order of this dynamic quantizer ($N \geq 1$). Equals the number of rows (or columns) in A .
- **delta**: The maximum allowed quantization error of q .

A.2. CLASSES

Methods

■ **__init__(A, B, C, q)** Initialize a DynamicQuantizer instance. The dynamic quantizer is defined by the following equations:

$$Q: \begin{cases} \xi(t+1) &= A\xi(t) + Bu(t) \\ v(t) &= q(C\xi(t) + u(t)) \end{cases} \quad (\text{A.2})$$

Parameters:

- **A** (*array_like*) – State matrix ($N \times N$, real). $N \geq 1$.
- **B** (*array_like*) – Input matrix ($N \times m$, real). $m \geq 1$.
- **C** (*array_like*) – Output matrix ($m \times N$, real).
- **q** (*StaticQuantizer*) – Static quantizer instance.

Methods

■ **cost(system, steptime=Inflnt.Value, _check_stability=True)** Compute the cost ($E(Q)$) for this quantizer and system.

■ **design(system, *, q, steptime=Inflnt.Value, max_gain_wv=inf, max_N=Inflnt.Value, verbose=False, use_analytical_method=True, use_LP_method=True, use_design_GB_method=True, use_DE_method=False, solver=None)** Design a stable and optimal dynamic quantizer for a system.

Example:

```
import nqlib
P = nqlib.Plant(
    A=[[ 1.8, 0.8],
        [-1.0, 0. ]],
    B=[[1.],
        [0.]],
    C1=[0.01, -0.09],
    C2=[0, 0],
)
G = nqlib.System.from_FF(P)
q = nqlib.StaticQuantizer.mid_tread(d=2)
Q, E = nqlib.DynamicQuantizer.design(
    system=G,
    q=q,
    steptime=100,
```

A.2. CLASSES

```
)  
Q.is_stable  
# True  
E < 0.5  
# np.True_
```

■ **design_AG(system, *, q, allow_unstable=False, verbose=False)** Algebraically design a stable and optimal dynamic quantizer for a system.

Example:

```
import nqlib  
P = nqlib.Plant(  
    A=[[ 1.8, 0.8],  
        [-1.0, 0. ]],  
    B=[[1.],  
        [0.]],  
    C1=[0.01, -0.09],  
    C2=[0, 0],  
)  
G = nqlib.System.from_FF(P)  
q = nqlib.StaticQuantizer.mid_tread(d=2)  
Q, E = nqlib.DynamicQuantizer.design_AG(system=G, q=q)  
Q.is_stable  
# True  
E < 0.5  
# np.True_
```

■ **design_DE(system, *, q, N, steptime=Inflnt.Value, max_gain_wv=inf, verbose=False)** Design a stable and optimal dynamic quantizer using differential evolution.

Example:

```
import nqlib  
G = nqlib.System(  
    A=[[1.15, 0.05],  
        [0.00, 0.99]],  
    B1=[[1],  
        [1]],  
    B2=[[0.004],  
        [0.099]],  
    C1=[1., 0.], C2=[-15., -3.],  
    D1=0, D2=0,
```

A.2. CLASSES

```
)
q = nqlib.StaticQuantizer.mid_tread(d=2)
Q, E = nqlib.DynamicQuantizer.design_DE(
    system=G,
    q=q,
    N=2,
)
Q.is_stable
# True
E < 0.01
# np.True_
```

■ `design_GD(system, *, q, N, steptime=Inflnt.Value, max_gain_wv=inf, verbose=False, method='SLSQP', obj_type='exp')` Design a stable and optimal dynamic quantizer using gradient-based optimization.

Example:

```
import nqlib
G = nqlib.System(
    A=[[1.15, 0.05],
        [0.00, 0.99]],
    B1=[[1],
        [1]],
    B2=[[0.004],
        [0.099]],
    C1=[1., 0.], C2=[-15., -3.],
    D1=0, D2=0,
)
q = nqlib.StaticQuantizer.mid_tread(d=2)
Q, E = nqlib.DynamicQuantizer.design_GD(
    system=G,
    q=q,
    N=2,
)
Q.is_stable
# True
E < 0.01
# np.True_
```

A.2. CLASSES

■ **design_LP**(system, *, q, dim=Inflnt.Value, T=Inflnt.Value, max_gain_wv=inf, solver=None, verbose=False) Design a stable and optimal dynamic quantizer using the linear programming method.

Example:

```
import nqlib
P = nqlib.Plant(
    A=[[ 1.8, 0.8],
        [-1.0, 0. ]],
    B=[[1.],
        [0.]],
    C1=[0.01, -0.09],
    C2=[0, 0],
)
G = nqlib.System.from_FF(P)
q = nqlib.StaticQuantizer.mid_tread(d=2)
Q, E = nqlib.DynamicQuantizer.design_LP(
    system=G,
    q=q,
    T=100,
    dim=2,
    max_gain_wv=2.0,
)
Q.is_stable
# True
E < 0.5
# np.True_
```

■ **from_SISO_parameters**(parameters, *, q) Create a SISO dynamic quantizer from parameters.

Example:

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(0.1)
Q = nqlib.DynamicQuantizer.from_SISO_parameters([1, 0.33], q=q)
```

■ **gain_wv**(steptime=Inflnt.Value, verbose=False) Compute the gain from w to v for this quantizer.

Example:

A.2. CLASSES

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(0.1)
dq = nqlib.DynamicQuantizer(0.5, 0.1, 0.1, q)
dq.gain_wv() < 1.021
# np.True_
```

■ **objective_function(system, stepsize=Inflnt.Value)** Calculate the objective function $E(Q)$ for a given system.

Example:

```
import nqlib
G = nqlib.System(
    A=[[1.15, 0.05],
        [0.00, 0.99]],
    B1=[[1],
        [1]],
    B2=[[0.004],
        [0.099]],
    C1=[1., 0.], C2=[-15., -3.],
    D1=0, D2=0,
)
q = nqlib.StaticQuantizer.mid_tread(0.1)
Q, E = nqlib.DynamicQuantizer.design_AG(system=G, q=q)
e = Q.objective_function(G)
abs(e - E) < 1e-4
# True
```

■ **order_reduced(new_N)** Returns a reduced-order quantizer.

Example:

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(0.1)
import numpy as np
Q = nqlib.DynamicQuantizer(
    np.eye(2)*0.5,
    np.eye(2, 1),
    np.eye(1, 2),
    q
)
Q.N # Original order
# 2
```

A.2. CLASSES

```
Q2 = Q.order_reduced(1) # Reduce the order to 1
Q2.N
# 1
```

Parameters:

- **new_N** (*int*) – Desired order ($1 \leq \text{new_N} < \text{self.N}$).

■ **quantize(u)** Quantize input u .

Example:

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(0.1)
Q = nqlib.DynamicQuantizer.from_SISO_parameters([0.9], q=q)
u = [0.1, 0.2, 0.3]
v = Q.quantize(u)
v.shape
# (1, 3)
```

■ **spec(steptime=InfInt.Value, show=True)** Specification of the quantizer (range and intervals).

Example:

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(0.1)
Q = nqlib.DynamicQuantizer.from_SISO_parameters([0.9], q=q)
Q.spec()
# (inf, 0.1)
```

■ **to_parameters()** Convert to parameter representation.

Example:

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(0.1)
Q = nqlib.DynamicQuantizer.from_SISO_parameters([0.9], q=q)
Q.to_parameters()
# [0.9]
```

Properties

■ **gain_uv** L_2 gain from u to v .

Example:

A.2. CLASSES

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(0.1)
Q = nqlib.DynamicQuantizer.from_SISO_parameters([0.9], q=q)
Q.gain_uv
# 1.0
```

■ **is_stable** Check stability of the dynamic quantizer.

Example:

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(0.1)
Q = nqlib.DynamicQuantizer.from_SISO_parameters([0.9], q=q)
Q.is_stable
# True
```

■ **m** Input/output dimension.

Example:

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(0.1)
Q = nqlib.DynamicQuantizer.from_SISO_parameters([0.9], q=q)
Q.m
# 1
```

■ **minreal** Minimal realization of the internal system H .

Example:

```
import nqlib
import numpy as np
q = nqlib.StaticQuantizer.mid_tread(0.1)
Q = nqlib.DynamicQuantizer(
    np.array([[0.1, 0], [0, 0.2]]),
    np.array([[1], [1]]),
    np.array([[1, 1]]),
    q
)
Q.N
# 2
Q_min = Q.minreal
Q_min.N
```

A.2. CLASSES

```
# 2
```

A.2.3 nqlib.Plant

State-space model of a plant P . The plant P is given by:

$$P: \begin{cases} x(t+1) &= Ax(t) + Bu(t) \\ z(t) &= C_1x(t) \\ y(t) &= C_2x(t) \end{cases} \quad (\text{A.3})$$

Methods

■ `__init__(A, B, C1, C2)` Initialize a Plant instance.

■ `from_TF(tf)` Create a Plant instance from a transfer function.

Example:

```
import nqlib
import control
tf = control.TransferFunction([1], [1, 2, 1], 1) # 1 / (s^2 + 2s + 1)
P = nqlib.Plant.from_TF(tf)
P.A.shape[0]
# 2
```

A.2.4 nqlib.StaticQuantizer

Static quantizer.

Properties

- **delta:** The maximum allowed quantization error. Declares that for any real vector u , $\max(|q(u) - u|) \leq \text{delta}$. $\text{delta} > 0$.

Methods

■ `__init__(function, delta, *, error_on_excess=True)` Initialize a StaticQuantizer q .

Example:

```
import nqlib
import numpy as np
q = nqlib.StaticQuantizer(lambda u: np.round(u), 1.0)
```

A.2. CLASSES

```
q([1.2, 2.3])  
# array([1., 2.]
```

■ `cost(system, steptime=InfInt.Value, _check_stability=True)` Compute the cost ($E(Q)$) for this quantizer and system.

Example:

```
import nqlib  
import numpy as np  
G = nqlib.System(  
    A=[[1.15, 0.05],  
        [0.00, 0.99]],  
    B1=[[1],  
        [1]],  
    B2=[[0.004],  
        [0.099]],  
    C1=[1., 0.], C2=[-15., -3.],  
    D1=0, D2=0,  
)  
q = nqlib.StaticQuantizer.mid_tread(d=2)  
Q, E = nqlib.DynamicQuantizer.design_AG(  
    system=G,  
    q=q,  
)  
Q.cost(G) < q.cost(G)  
# np.True_
```

■ `mid_riser(d, bit=InfInt.Value, *, error_on_excess=True)` Create a mid-riser uniform StaticQuantizer.

Example:

```
import nqlib  
q = nqlib.StaticQuantizer.mid_riser(0.5)  
q([0.2, 0.7, 1.1, 0, -1])  
# array([ 0.25,  0.75,  1.25,  0.25, -0.75])
```

■ `mid_tread(d, bit=InfInt.Value, *, error_on_excess=True)` Create a mid-tread uniform StaticQuantizer.

Example:

```
import nqlib
```

A.2. CLASSES

```
q = nqlib.StaticQuantizer.mid_tread(0.5)
q([0.2, 0.7, 1.1, 0, -1])
# array([ 0. ,  0.5,  1. ,  0. , -1. ])
```

■ `quantize(u)` Quantize the input signal.

Example:

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(0.5)
u = [0, 0.32, 0.44]
q.quantize(u)
# array([0. , 0.5, 0.5])
```

A.2.5 nqlib.System

State-space model of an ideal system. The system is given by:

$$G: \begin{cases} x(t+1) &= Ax(t) + B_1r(t) + B_2v(t) \\ z(t) &= C_1x(t) + D_1r(t) \\ u(t) &= C_2x(t) + D_2r(t) \end{cases} \quad (\text{A.4})$$

Methods

■ `__init__(A, B1, B2, C1, C2, D1, D2)` Initialize a System instance.

Example:

```
import nqlib
G = nqlib.System(
    A=[[1.15, 0.05],
        [0.00, 0.99]],
    B1=[[0.],
        [0.]],
    B2=[[0.004],
        [0.099]],
    C1=[1., 0.],
    C2=[-15., -3.],
    D1=0,
    D2=1,
)
G.A
# array([[1.15, 0.05],
```

A.2. CLASSES

```
# [0. , 0.99]])
```

■ `E(Q, steptime=Inflnt.Value, _check_stability=True, verbose=False)` Estimate $E(Q)$ for the system and quantizer.

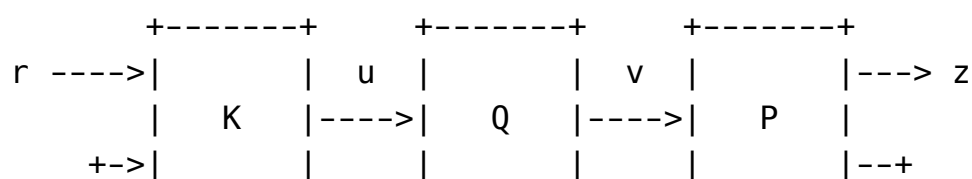
Example:

```
import nqlib
import numpy as np
G = nqlib.System(
    A=[[1.15, 0.05],
        [0.00, 0.99]],
    B1=[[1],
        [1]],
    B2=[[0.004],
        [0.099]],
    C1=[1., 0.], C2=[-15., -3.],
    D1=0, D2=0,
)
q = nqlib.StaticQuantizer.mid_tread(d=2)
Q, E = nqlib.DynamicQuantizer.design_AG(
    system=G,
    q=q,
)
np.isclose(E, G.E(Q))
# np.True_
G.E(Q) < G.E(q)
# np.True_
```

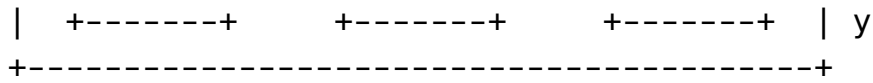
■ `from_FBIQ(P, K)` Alias for `from_FB_connection_with_input_quantizer`.

■ `from_FBOQ(P, K)` Alias for `from_FB_connection_with_output_quantizer`.

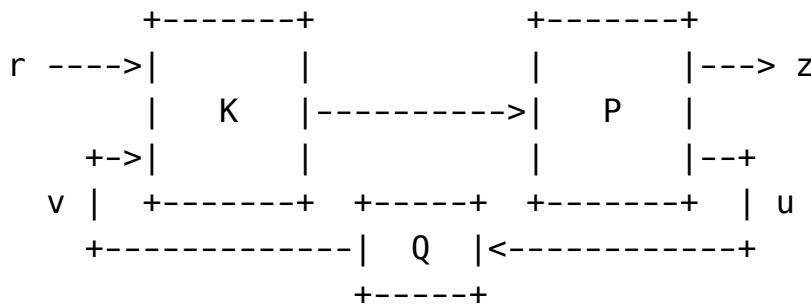
■ `from_FB_connection_with_input_quantizer(P, K)` Create a System instance from Plant and Controller with input quantizer (feedback connection). 'from_FB_connection_with_input_quantizer' means that a quantizer is inserted as shown in the following figure.



A.2. CLASSES



■ `from_FB_connection_with_output_quantizer(P, K)` Create a System instance from Plant and Controller with output quantizer (feedback connection). 'from_FB_connection_with_output_quantizer' means that a quantizer is inserted as shown in the following figure.



■ `from_FF(P)` Create a System instance from a Plant (feedforward connection).

■ `is_stable` Check if the closed-loop system is stable.

Example:

```
import nqlib
sys = nqlib.System(0.3, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)
sys.is_stable
# True
```

■ `is_stable_with(Q)` Alias for `is_stable_with_quantizer`.

■ `is_stable_with_quantizer(Q)` Check if the closed-loop system with the quantizer is stable.

Example:

```
import nqlib
G = nqlib.System(
    A=[[1.15, 0.05],
        [0.00, 0.99]],
    B1=[[1],
        [1]],
    B2=[[0.004],
        [0.099]],
    C1=[1., 0.], C2=[-15., -3.],
```

A.3. FUNCTIONS

```
D1=0, D2=0,
)
q = nqlib.StaticQuantizer.mid_tread(d=2)
Q, E = nqlib.DynamicQuantizer.design_AG(
    system=G,
    q=q,
)
G.is_stable_with_quantizer(Q)
# True
```

■ `response(input, x_0)` Simulate the system and return results.

Example:

```
import nqlib
import numpy as np
sys = nqlib.System(1, 1, 1, 1, 1, 1, 1)
t, u, z = sys.response(np.ones((1, 5)), np.zeros((1, 1)))
t.shape
# (1, 5)
```

■ `response_with_quantizer(quantizer, input, x_0)` Simulate the system with a quantizer and return results.

Example:

```
import nqlib
import numpy as np
sys = nqlib.System(1, 1, 1, 1, 1, 1, 1)
q = nqlib.StaticQuantizer.mid_tread(0.1)
t, u, v, z = sys.response_with_quantizer(q, np.ones((1, 5)), np.
    zeros((1, 1)))
```

A.3 Functions

A.3.1 `nqlib.installed_solvers`

■ `installed_solvers()` List the installed solvers.

A.3.2 `nqlib.order_reduced`

■ `order_reduced(Q, new_N)` Returns the quantizer with its order reduced.

A.3. FUNCTIONS

Note that the quantizer with the reduced order will generally have larger $E(Q)$ and a larger gain_wv than those of the original quantizer. You should check the performance and gain yourself.

This function requires `slycot`. Please install it.

Example:

```
import nqlib
q = nqlib.StaticQuantizer.mid_tread(0.1)
import numpy as np
Q = nqlib.DynamicQuantizer(
    np.eye(2)*0.4,
    np.eye(2, 1),
    np.eye(1, 2),
    q
)
Q.N # Original order
# 2
Q2 = nqlib.order_reduced(Q, 1) # Reduce the order to 1
Q2.N
# 1
```

Parameters:

- **Q** (*DynamicQuantizer*) – The quantizer to be reduced. Must be an instance of *DynamicQuantizer*.
- **new_N** (*int*) – Desired order ($1 \leq \text{new_N} < Q.N$).

Bibliography

- [1] Y. LIU and Y. MINAMI: Data-driven command shaping for improved torque control accuracy of torsional dampers; *Transactions of the JSME (in Japanese)*, Vol. 91, No. 946, pp. 24–00273–24–00273 (2025)
- [2] J. Vaughan, E. Maleki and W. Singhose: Advantages of using command shaping over feedback for crane control; in *Proceedings of the 2010 American Control Conference*, pp. 2308–2313 (2010)
- [3] W. Singhose: Command shaping for flexible systems: A review of the first 50 years; *International journal of precision engineering and manufacturing*, Vol. 10, No. 4, pp. 153–168 (2009)
- [4] E. Garone, S. Di Cairano and I. Kolmanovsky: Reference and command governors for systems with constraints: A survey on theory and applications; *Automatica*, Vol. 75, pp. 306–328 (2017)
- [5] W. Xue et al.: Add-on module of active disturbance rejection for set-point tracking of motion control systems; *IEEE Transactions on Industry Applications*, Vol. 53, No. 4, pp. 4028–4040 (2017)
- [6] F. Richter et al.: Augmented Reality Predictive Displays to Help Mitigate the Effects of Delayed Telesurgery; in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 444–450 (2019)
- [7] Q. Fu, L. Zhang, Y. Xu and F. You: The Review of Human–Machine Collaborative Intelligent Interaction With Driver Cognition in the Loop; *Systems Research and Behavioral Science* (2025)
- [8] E. R. Loskill, A. COMMAND and S. C. M. A. AL: The All-Aspect Gunsight: Returning Short Range Combat Efficiency to the F-15.; (1988)
- [9] G. Thomson: Advances in electro-optic systems for targeting; *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 212, No. 1, pp. 1–19 (1998)
- [10] E. F. Papavasileiou, A. Papatheodorou and J. Edmunds: Human operators in air transport: A decade of systematic reviews; *Journal of the Air Transport Research Society*,

-
- Vol. 4, p. 100059 (2025)
- [11] D. Kim and P. Y. Oh: Human-embodied drone interface for aerial manipulation: advantages and challenges; *Intelligent Service Robotics*, Vol. 17, No. 3, pp. 375–388 (2024)
 - [12] R. K. Saini, M. Raju and A. Chail: Cry in the sky: Psychological impact on drone operators; *Industrial psychiatry journal*, Vol. 30, No. Suppl 1, pp. S15–S19 (2021)
 - [13] J. Duarte, A. T. Marques and J. Santos Baptista: Occupational Accidents Related to Heavy Machinery: A Systematic Review; *Safety*, Vol. 7, No. 1 (2021)
 - [14] K. Catchpole et al.: Human factors in robotic assisted surgery: Lessons from studies ‘in the Wild’ ; *Applied Ergonomics*, Vol. 78, pp. 270–276 (2019)
 - [15] A. D. Ames, X. Xu, J. W. Grizzle and P. Tabuada: Control barrier function based quadratic programs for safety critical systems; *IEEE Transactions on Automatic Control*, Vol. 62, No. 8, pp. 3861–3876 (2016)
 - [16] A. D. Ames, J. W. Grizzle and P. Tabuada: Control barrier function based quadratic programs with application to adaptive cruise control; in *53rd IEEE Conference on Decision and Control*, pp. 6271–6278 (2014)
 - [17] R. Morita, S. Azuma and T. Sugie: Performance Analysis of Random Dither Quantizers in Feedback Control Systems; *IFAC Proceedings Volumes*, Vol. 44, No. 1, pp. 11296–11301 (2011), 18th IFAC World Congress
 - [18] N. C. Singer and W. P. Seering: Preshaping command inputs to reduce system vibration; *Journal of dynamic systems, measurement, and control*, Vol. 112, No. 1, pp. 76–82 (1990)
 - [19] W.-H. Chen, J. Yang, L. Guo and S. Li: Disturbance-observer-based control and related methods—An overview ; *IEEE Transactions on industrial electronics*, Vol. 63, No. 2, pp. 1083–1095 (2015)
 - [20] W.-H. Chen: Disturbance observer based control for nonlinear systems; *IEEE/ASME transactions on mechatronics*, Vol. 9, No. 4, pp. 706–710 (2004)
 - [21] A. J. Krener: On the equivalence of control systems and the linearization of nonlinear systems; *SIAM Journal on Control*, Vol. 11, No. 4, pp. 670–676 (1973)
 - [22] B. Jaw Jakubczyk and W. Respondek: On linearization of control systems; *Bull. Acad. Polon. Sci. Ser. Sci. Math*, Vol. 28, No. 9-10, pp. 517–522 (1980)
 - [23] N. A. S. Alwan and Z. M. Hussain: Deep Learning for Robust Adaptive Inverse Control of Nonlinear Dynamic Systems: Improved Settling Time with an Autoencoder; *Sensors*, Vol. 22, No. 16 (2022)
 - [24] J. B. Hoagg and D. S. Bernstein: Nonminimum-phase zeros - much to do about nothing - classical control - revisited part II; *IEEE Control Systems Magazine*, Vol. 27, No. 3, pp. 45–57 (2007)
 - [25] M. Charfeddine, K. Jouili and M. Ben Moussa: Adaptive control of nonlinear non-

-
- minimum phase systems using actor–critic reinforcement learning; *Symmetry*, Vol. 17, No. 7, p. 1083 (2025)
- [26] Y. Minami, S. Azuma and T. Sugie: An optimal dynamic quantizer for feedback control with discrete-valued signal constraints; in *2007 46th IEEE Conference on Decision and Control*, pp. 2259–2264 (2007)
- [27] S. Azuma and T. Sugie: Optimal dynamic quantizers for discrete-valued input control; *Automatica*, Vol. 44, No. 2, pp. 396–406 (2008)
- [28] S. Azuma and T. Sugie: Stability analysis of quantized feedback systems including optimal dynamic quantizers; in *2008 47th IEEE Conference on Decision and Control*, pp. 3392–3397 (2008)
- [29] S. Azuma and T. Sugie: Synthesis of optimal dynamic quantizers for discrete-valued input control; *IEEE Transactions on Automatic Control*, Vol. 53, No. 9, pp. 2064–2075 (2008)
- [30] S. Azuma, Y. Minami and T. Sugie: Optimal dynamic quantizers for feedback control with discrete-level actuators: unified solution and experimental evaluation; *Journal of Dynamic Systems, Measurement, and Control*, Vol. 133, No. 2 (2011)
- [31] Y. Minami and T. Muromaki: Differential Evolution-Based Synthesis of Dynamic Quantizers with Fixed-Structures; *International Journal of Computational Intelligence and Applications*, Vol. 15, No. 02, p. 1650008 (2016)
- [32] Y. Minami: Design of model following control systems with discrete-valued signal constraints; *International Journal of Control, Automation and Systems*, Vol. 14, No. 1, pp. 331–339 (2016)
- [33] Y. Minami and K. Kashima: Dynamic Quantizer Design based on Serial System Decomposition (in Japanese); in *22nd International Symposium on Mathematical Theory of Networks and Systems*, pp. 577–579 (2016)
- [34] R. Morita, Y. Minami, S. Azuma and T. Sugie: ODQLab: A graphical software tool to design dynamic quantizers for discrete-valued input control; in *2010 IEEE International Conference on Control Applications*, pp. 2356–2361 (2010)
- [35] D. Silver et al.: Mastering the game of Go with deep neural networks and tree search; *nature*, Vol. 529, No. 7587, pp. 484–489 (2016)
- [36] A. Kuhnle et al.: Designing an adaptive production control system using reinforcement learning; *Journal of Intelligent Manufacturing*, Vol. 32, No. 3, pp. 855–876 (2021)
- [37] J. Huang et al.: Model-based Adversarial Imitation Learning from Demonstrations and Human Reward; in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1683–1690 (2023)
- [38] J. Liu et al.: Prediction of human–machine interface (HMI) operational errors for mar-

-
- itime autonomous surface ships (MASS); *Journal of Marine Science and Technology*, Vol. 27, No. 1, pp. 293–306 (2022)
- [39] M. CAVA, G. Paletta and C. Picardi: Stability analysis of PWM control systems with PID regulators; *International Journal of Control - INT J CONTR*, Vol. 39, pp. 987–1005 (1984)
- [40] H. Inose, Y. Yasuda and J. Murakami: A Telemetering System by Code Modulation - Δ - Σ Modulation; *IRE Transactions on Space Electronics and Telemetry*, Vol. SET-8, pp. 204–209 (1962)
- [41] M. Hagiwara, T. Kitayabu, H. Ishikawa and H. Shirai: Delta-sigma modulator with non-uniform quantization; in *2011 IEEE Radio and Wireless Symposium*, pp. 351–354 (2011)
- [42] Y. Chida, N. Ishihara and M. Tanemura: Multirate and Model Predictive Control of a Pneumatic Isolation Table with a Discrete Actuator; *IFAC-PapersOnLine*, Vol. 52, No. 15, pp. 442–447 (2019), 8th IFAC Symposium on Mechatronic Systems MECHATRONICS 2019
- [43] H. Qi, G. M. Bone and Y. Zhang: Position control of pneumatic actuators using three-mode discrete-valued model predictive control; in *Actuators*, Vol. 8, p. 56 (2019)
- [44] Y. Tange, S. Kiryu and T. Matsui: Model Predictive Control Based on Deep Reinforcement Learning Method with Discrete-Valued Input; in *2019 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 308–313 (2019)
- [45] S. Azuma and T. Sugie: Synthesis of Optimal Dynamic Quantizers for Symbolic Input Control; *Transactions of the Institute of Systems, Control and Information Engineers*, Vol. 20, No. 3, pp. 122–129 (2007)
- [46] J. A. Snyman: Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms, Springer (2005)
- [47] S. Das and P. N. Suganthan: Differential evolution: A survey of the state-of-the-art; *IEEE transactions on evolutionary computation*, Vol. 15, No. 1, pp. 4–31 (2010)
- [48] R. Morita, S. Azuma, Y. Minami and T. Sugie: Graphical design software for dynamic quantizers in control systems; *SICE Journal of Control, Measurement, and System Integration*, Vol. 4, No. 5, pp. 372–379 (2011)
- [49] Welcome to Python.org; <https://www.python.org/>
- [50] S. Fuller et al.: The Python Control Systems Library (python-control); in *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 4875–4881 (2021)
- [51] Welcome To Colaboratory - Colaboratory; <https://colab.research.google.com/notebooks/welcome.ipynb>
- [52] PyPI · The Python Package Index; <https://pypi.org/>
- [53] NumPy; <https://numpy.org/>

-
- [54] SciPy.org — SciPy.org ; , <https://www.scipy.org/>
- [55] python-control/Slycot: Python wrapper for the Subroutine Library in Systems and Control Theory (SLICOT); <https://github.com/python-control/Slycot>
- [56] Control Systems Library for Python — GitHub; <https://github.com/python-control>
- [57] S. Diamond and S. Boyd: CVXPY: A Python-embedded modeling language for convex optimization; *Journal of Machine Learning Research*, Vol. 17, No. 83, pp. 1–5 (2016)
- [58] A. Agrawal, R. Verschueren, S. Diamond and S. Boyd: A rewriting system for convex optimization problems; *Journal of Control and Decision*, Vol. 5, No. 1, pp. 42–60 (2018)
- [59] R. S. Sutton, A. G. Barto et al.: Reinforcement learning: An introduction, Vol. 1 MIT press Cambridge (1998)
- [60] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath: Deep Reinforcement Learning: A Brief Survey; *IEEE Signal Processing Magazine*, Vol. 34, No. 6, pp. 26–38 (2017)
- [61] D. Silver et al.: Mastering the game of Go without human knowledge; *Nature*, Vol. 550, No. 7676, pp. 354–359 (2017)
- [62] E. F. Morales, R. Murrieta-Cid, I. Becerra and M. A. Esquivel-Basaldúa: A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning; *Intelligent Service Robotics*, Vol. 14, No. 5, pp. 773–805 (2021)
- [63] D. Han, B. Mulyana, V. Stankovic and S. Cheng: A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation; *Sensors*, Vol. 23, No. 7 (2023)
- [64] S. Aradi: Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles; *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 2, pp. 740–759 (2022)
- [65] A. Krishna G.S., T. Zhang, O. Ardakanian and M. E. Taylor: Mitigating an adoption barrier of reinforcement learning-based control strategies in buildings; *Energy and Buildings*, Vol. 285, p. 112878 (2023)
- [66] Z. Du et al.: Green Deep Reinforcement Learning for Radio Resource Management: Architecture, Algorithm Compression, and Challenges; *IEEE Vehicular Technology Magazine*, Vol. 16, No. 1, pp. 29–39 (2021)
- [67] A. Y. Ng, D. Harada and S. Russell: Policy invariance under reward transformations: Theory and application to reward shaping; in *ICML*, Vol. 99, pp. 278–287 (1999)
- [68] T. P. Lillicrap et al.: Continuous control with deep reinforcement learning; *arXiv preprint arXiv:1509.02971* (2015)
- [69] X. Zhang, T. M. Seigler and J. B. Hoagg: The Impact of Nonminimum-Phase Zeros on Human-in-the-Loop Control Systems; *IEEE Transactions on Cybernetics*, Vol. 52, No. 6,

-
- pp. 5098–5112 (2022), Conference Name: IEEE Transactions on Cybernetics
- [70] F. Tavakkoli, P. Sarhadi, B. Clement and W. Naeem: Model Free Deep Deterministic Policy Gradient Controller for Setpoint Tracking of Non-Minimum Phase Systems; in *2024 UKACC 14th International Conference on Control (CONTROL)*, pp. 1–6 (2024)
 - [71] M. Charfeddine, K. Jouili and M. Ben Moussa: Adaptive Control of Nonlinear Non-Minimum Phase Systems Using Actor-Critic Reinforcement Learning; *Symmetry*, Vol. 17, No. 7 (2025)
 - [72] J. Schulman et al.: Proximal policy optimization algorithms; *arXiv preprint arXiv:1707.06347* (2017)
 - [73] D. P. Kingma: Adam: A method for stochastic optimization; *arXiv preprint arXiv:1412.6980* (2014)
 - [74] S. M. Petermeijer, D. A. Abbink, M. Mulder and J. C. De Winter: The effect of haptic support systems on driver performance: A literature survey; *IEEE transactions on haptics*, Vol. 8, No. 4, pp. 467–479 (2015)
 - [75] K. Matsubara and M. Omae: Compensation of Camera Image Latency in Remotely Operated Vehicle using Projection Transformation (in Japanese); in *19th ITS Symposium* (2021)
 - [76] M. Omae: Remote Control of Vehicles (Connecting Machines: Fusion of Machines and Communication) (in Japanese); *Journal of the Society of Mechanical Engineers*, Vol. 125, No. 1246, pp. 28–31 (2022)
 - [77] R. E. Fenton: An Improved Man-Machine Interface for the Driver-Vehicle System; *IEEE Transactions on Human Factors in Electronics*, Vol. HFE-7, No. 4, pp. 150–157 (1966)
 - [78] P. Bazilinskyy et al.: Take-over requests in highly automated driving: A crowdsourcing survey on auditory, vibrotactile, and visual displays; *Transportation research part F: traffic psychology and behaviour*, Vol. 56, pp. 82–98 (2018)
 - [79] T. A. Sitompul and M. Wallmyr: Using augmented reality to improve productivity and safety for heavy machinery operators: State of the art; in *Proceedings of the 17th International Conference on Virtual-Reality Continuum and Its Applications in Industry*, pp. 1–9 (2019)
 - [80] A. Zolghadri: A review of fault management issues in aircraft systems: Current status and future directions; *Progress in Aerospace Sciences*, Vol. 147, p. 101008 (2024)
 - [81] R. A. Suárez Fernández et al.: Natural user interfaces for human-drone multi-modal interaction; in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1013–1022 (2016)
 - [82] D. Tezza and M. Andujar: *ieee access*, Vol. 7, pp. 167438–167454 (2019)
 - [83] V. Villani, F. Pini, F. Leali and C. Secchi: Survey on human-robot collaboration in

-
- industrial settings: Safety, intuitive interfaces and applications; *Mechatronics*, Vol. 55, pp. 248–266 (2018)
- [84] A. Simorov, R. S. Otte, C. M. Kopietz and D. Oleynikov: Review of surgical robotics user interface: what is the best way to control robotic surgery?; *Surgical endoscopy*, Vol. 26, pp. 2117–2125 (2012)
- [85] A. Dünser, M. Lochner, U. Engelke and D. R. Fernández: Visual and manual control for human-robot teleoperation; *IEEE computer graphics and applications*, Vol. 35, No. 3, pp. 22–32 (2015)
- [86] K. MÃ³ga, D. B. Boesl and T. Haidegger: Augmented/Mixed Reality Technologies Supporting Digital Surgery; in *2021 IEEE 19th International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 183–189 (2021)
- [87] S. Livatino et al.: Intuitive Robot Teleoperation Through Multi-Sensor Informed Mixed Reality Visual Aids; *IEEE Access*, Vol. 9, pp. 25795–25808 (2021)
- [88] I. Tezuka et al.: Input-constrained Human Assist Control via Control Barrier Function for Viability Kernel; in *2023 American Control Conference (ACC)*, pp. 3921–3926 (2023)
- [89] B. Xu and K. Sreenath: Safe Teleoperation of Dynamic UAVs Through Control Barrier Functions; in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7848–7855 (2018)
- [90] S. Hansen and D. Elliott: Three-dimensional manual responses to unexpected target perturbations during rapid aiming; *Journal of motor behavior*, Vol. 41, No. 1, pp. 16–29 (2009)
- [91] C. Hughes, M. Glavin, E. Jones and P. Denny: Wide-angle camera technology for automotive applications: a review; *IET Intelligent Transport Systems*, Vol. 3, pp. 19–31 (2009)
- [92] K. Sakiyama, T. Shimizu and K. Sako: Parking Assist System; *Fujitsu Tech. Ma*, No. 15 (2000)
- [93] K. Maruta et al.: Blind-Spot Visualization via AR Glasses using Millimeter-Wave V2X for Safe Driving; in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, pp. 1–5 (2021)
- [94] M. Regodić et al.: Visual display for surgical targeting: concepts and usability study; *International Journal of Computer Assisted Radiology and Surgery*, Vol. 16, pp. 1565–1576 (2021)
- [95] L. Zhao et al.: The Influence of Motion-Cueing, Sound and Vibration Feedback on Driving Behavior and Experience—A Virtual Teleoperation Experiment ; *IEEE Transactions on Intelligent Transportation Systems*, Vol. 25, No. 8, pp. 9797–9809 (2024)
- [96] M. Chu et al.: Multisensory Fusion, Haptic, and Visual Feedback Teleoperation System

-
- Under IoT Framework; *IEEE Internet of Things Journal*, Vol. 9, No. 20, pp. 19717–19727 (2022)
- [97] R. E. Fenton and W. B. Montano: An Intervehicular Spacing Display for Improved Car-Following Performance; *IEEE Transactions on Man-Machine Systems*, Vol. 9, No. 2, pp. 29–35 (1968)
- [98] K. O. Koerten, D. A. Abbink and A. Zgonnikov: Haptic Shared Control for Dissipating Phantom Traffic Jams; *IEEE Transactions on Human-Machine Systems*, Vol. 54, No. 1, pp. 11–20 (2024)
- [99] M. C S et al.: Design and Analysis of New Haptic Joysticks for Enhancing Operational Skills in Excavator Control; *Journal of Mechanical Design*, Vol. 142, No. 12, p. 121406 (2020)
- [100] R. Sekizuka et al.: Force Feedback Design of Operation Levers Considering the Characteristics of Human Force Perception to Improve Hydraulic Excavator Operability; *IEEE Access*, Vol. 10, pp. 926–938 (2022)
- [101] M. Korayem, M. Madihi and V. Vahidifar: Controlling surgical robot arm using leap motion controller with Kalman filter; *Measurement*, Vol. 178, p. 109372 (2021)
- [102] E. Wu and H. Koike: Futurepose-mixed reality martial arts training using real-time 3d human pose forecasting with a rgb camera; in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1384–1392 (2019)

Related papers

The results of this dissertation are published in the following articles.

Chapter 3

- [1] Kenta Tanaka, Yuki Minami, Masato Ishikawa: NQLib: A Python Library for Noise-Shaping Quantizer Synthesis; *SoftwareX*, Vol. 27, 101792 (2024)
- [2] Kenta Tanaka, Yuki Minami, Masato Ishikawa: NQLib: Python Library for the Practical Design of Noise Shaping Quantizers; in 2021 SICE Annual Conference, pp. 814–816 (2021)
- [3] 田中 健太, 南 裕樹, 石川 将人: ノイズシェーピング量子化器設計のための Python ライブラリ NQLib の開発; 自動制御連合講演会講演論文集 第 63 回 自動制御連合講演会, pp. 474–477 (2020)

Chapter 4

- [1] 田中 健太, 南 裕樹, 石川 将人: 情報改変による強化学習の効率化: 倒立振子の安定化制御を例として; システム制御情報学会論文誌, Vol. 37, No. 8, pp. 234–236 (2024)
- [2] Kenta Tanaka, Yuki Minami, Masato Ishikawa: Reinforcement Learning-Based Analysis of the Effect of Presenting Predicted State on Operability of 2nd-Order LTI Systems in Human-Machine Systems; in 2025 SICE Festival with Annual Conference, ThAT11.18 (2025) (not refereed)
- [3] 田中 健太, 南 裕樹, 石川 将人: 人間機械系の制御性能向上のための視覚情報の改変; 自動制御連合講演会講演論文集 第 65 回 自動制御連合講演会, pp. 824–827 (2022)

Chapter 5

- [1] Kenta Tanaka, Yuki Minami, Masato Ishikawa: Operability improvement of Human-machine Systems by Altering Visual Information: A Case-study of the Inverted Pendulum; IEEE Access, Vol. 13, pp. 93566–93573 (2025)
- [2] 田中 健太, 南 裕樹, 石川 将人: 人間機械系における視覚情報改変による操作性向上効果の線形モデル解析; 計測自動制御学会論文集, (2025) 投稿中
- [3] Kenta Tanaka, Yuki Minami, Masato Ishikawa: Improving Controllability of Manual Control Systems by Altering Visual Information; in 29th International Symposium on Artificial Life and Robotics, pp. 356–359 (2024)
- [4] Kenta Tanaka, Yuki Minami, Masato Ishikawa: Reinforcement Learning-Based Analysis of the Effect of Presenting Predicted State on Operability of 2nd-Order LTI Systems in Human-Machine Systems; in 2025 SICE Festival with Annual Conference, ThAT11.18 (2025) (not refereed)
- [5] 田中 健太, 南 裕樹, 石川 将人: 人間機械系の制御性能向上のための視覚情報の改変; 自動制御連合講演会講演論文集 第 65 回 自動制御連合講演会, pp. 824–827 (2022)
- [6] 田中 健太, 南 裕樹, 石川 将人: 不安定零点を有する動的システムの手動制御のための視覚情報改変; 第 12 回計測自動制御学会制御部門マルチシンポジウム, 1PS2–17 (2025)

Appendix A

- [1] Kenta Tanaka, Yuki Minami, Masato Ishikawa: NQLib: A Python Library for Noise-Shaping Quantizer Synthesis; SoftwareX, Vol. 27, 101792 (2024)
- [2] Kenta Tanaka, Yuki Minami, Masato Ishikawa: NQLib: Python Library for the Practical Design of Noise Shaping Quantizers; in 2021 SICE Annual Conference, pp. 814–816 (2021)
- [3] 田中 健太, 南 裕樹, 石川 将人: ノイズシェーピング量子化器設計のための Python ライブラリ NQLib の開発; 自動制御連合講演会講演論文集 第 63 回自動制御連合講演会, pp. 474–477 (2020)

Publication list

Journal papers

- [1] 田中 健太, 南 裕樹, 石川 将人: 情報改変による強化学習の効率化: 倒立振子の安定化制御を例として; システム制御情報学会論文誌, Vol. 37, No. 8, pp. 234–236 (2024)
- [2] Kenta Tanaka, Yuki Minami, Masato Ishikawa: NQLib: A Python Library for Noise-Shaping Quantizer Synthesis; SoftwareX, Vol. 27, 101792 (2024)
- [3] Kenta Tanaka, Yuki Minami, Masato Ishikawa: Operability improvement of Human-machine Systems by Altering Visual Information: A Case-study of the Inverted Pendulum; IEEE Access, Vol. 13, pp. 93566–93573 (2025)
- [4] 田中 健太, 南 裕樹, 石川 将人: 人間機械系における視覚情報改変による操作性向上効果の線形モデル解析; 計測自動制御学会論文集, (2025) 投稿中

International conference papers

- [1] Kenta Tanaka, Yuki Minami, Masato Ishikawa: NQLib: Python Library for the Practical Design of Noise Shaping Quantizers; in 2021 SICE Annual Conference, pp. 814–816 (2021)
- [2] Kenta Tanaka, Yuki Minami, Masato Ishikawa: Improving Controllability of Manual Control Systems by Altering Visual Information; in 29th International Symposium on Artificial Life and Robotics, pp. 356–359 (2024)
- [3] Kenta Tanaka, Yuki Minami, Masato Ishikawa: Reinforcement Learning-Based Analysis of the Effect of Presenting Predicted State on Operability of 2nd-Order LTI Systems in Human-Machine Systems; in 2025 SICE Festival with Annual Conference, ThAT11.18 (2025) (not refereed)

Domestic conference papers

- [1] 田中 健太, 南 裕樹, 石川 将人: ノイズシェーピング量子化器設計のための Python ライブラリ NQLib の開発; 自動制御連合講演会講演論文集 第 63 回自動制御連合講演会, pp. 474–477 (2020)
- [2] 田中 健太, 南 裕樹, 石川 将人: 人間機械系の制御性能向上のための視覚情報の改変; 自動制御連合講演会講演論文集 第 65 回 自動制御連合講演会, pp. 824–827 (2022)
- [3] 南 裕樹, 田中 健太, 荻尾 優吾: Python を用いた制御系設計: えっ! そんなことできるの? ; 第 11 回計測自動制御学会制御部門マルチシンポジウム, 3A7–2 (2024)
- [4] 田中 健太, 南 裕樹, 石川 将人: 不安定零点を有する動的システムの手動制御のための視覚情報改変; 第 12 回計測自動制御学会制御部門マルチシンポジウム, 1PS2–17 (2025)

Honors and awards

- [1] 空気調和・衛生工学会振興賞学生賞 (2020 年度)
- [2] 計測自動制御学会優秀学生賞 (2022 年度)
- [3] 計測自動制御学会 マルチシンポジウム優秀ポスター発表賞 (2025)

Research grants

- [1] 次世代挑戦的研究者育成プロジェクトに採用 (2023/4–2026/3)
- [2] 立石科学技術振興財団研究助成 (C) に採用 (2024/4–2026/3)