# Efficient Algorithms for High-Dimensional Hamilton-Jacobi Partial Differential Equations and Optimal Control Problems

by

Paula X. Chen

B.A., Dartmouth College, Hanover, NH, 2017

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Division of Applied Mathematics at Brown University

PROVIDENCE, RHODE ISLAND

May 2023

This dissertation by Paula X. Chen is accepted in its present form
by the Division of Applied Mathematics as satisfying the
dissertation requirement for the degree of Doctor of Philosophy.

Date _____ _____

Jérôme Darbon, Ph.D., Advisor

Recommended to the Graduate Council

Date _____ _____

Chi-Wang Shu, Ph.D., Reader

Date _____ _____

Gary A. Hewer, Ph.D., Reader

Approved by the Graduate Council

Date _____ _____

Thomas A. Lewis, Dean of the Graduate School

# Vita

Paula Chen graduated from Dartmouth College in June 2017 with a Bachelor of Arts in mathematics. At Dartmouth, she researched mathematical models of cancer tumor growth in response to various chemotherapy treatments as a Women in Science Project intern supervised by Dr. Dorothy Wallace. She also completed a senior honors thesis advised by Dr. Alexander Barnett, titled "Neural spike sorting algorithms for overlapping spikes."

Paula earned a Doctor of Philosophy in applied mathematics from the Division of Applied Mathematics at Brown University in May 2023. Her Ph.D. advisor was Dr. Jérôme Darbon, and she was awarded a SMART Scholarship in 2021 for her doctoral studies. The primary topics of her doctoral research were Hamilton-Jacobi partial differential equations, applications in optimal control, efficient algorithms for high-dimensional problems, and FPGA implementations of scientific computing methods. Her preprints and publications include [27, 25, 24, 26], and she has a patent application [14].

# Acknowledgments

First and foremost, I thank my advisor Jérôme Darbon for his support and guidance over the years. I came to Brown wanting to work with Jérôme, and I could not be more happy that the gamble paid off. I am extremely grateful for his expertise, honesty, and life lessons (one day, I will watch The Wire), and I look forward to continuing to collaborate after graduation. Next, I would like to thank my dissertation readers, Professor Chi-Wang Shu and Dr. Gary Hewer, for their support and time. It has been a pleasure learning from and working with the both of you on various projects (that, unfortunately, did not make it into this dissertation). Additionally, I thank Phil Peters, Katia Estabridis, and Gary for sponsoring my SMART Scholarship; I look forward to working with you after graduation. I would also like to thank my (former) fellow research group members, Tingwei Meng, Taewoo Kim, and Gabriel Provencher Langlois. While transitioning from a research group of four to a group of one all at once was intimidating, I learned a lot from all of you and enjoyed working together when we could. Furthermore, occasionally having to coordinate between five different time zones kept working through a pandemic interesting. I also thank my other collaborators: George Em Karniadakis, Zongren Zou, Olivier Bokanowski, and Hasnaa Zidani. Finally, I express my gratitude for my family and friends, who have supported me throughout my life.

# Contents

**Vita** iv

**Acknowledgments** v

**1 Introduction** 1

**I Efficient algorithms based on representation formulas for certain high-dimensional problems** 4

**2 Hopf-type representation formulas** 11
2.1 Analytical solutions . . . 14
2.1.1 One-dimensional case . . . 15
2.1.2 Separable high-dimensional case . . . 23
2.1.3 The general high-dimensional case . . . 27
2.1.4 An extension to certain non-convex initial costs . . . 30
2.2 Efficient numerical algorithms . . . 32
2.2.1 Quadratic initial costs . . . 34
2.2.2 Convex initial costs . . . 36
2.2.3 A class of non-convex initial costs . . . 46
2.2.4 An FPGA implementation . . . 51
2.3 Some proofs and technical lemmas . . . 54
2.3.1 Technical lemmas for Section 2.1 . . . 54
2.3.2 Proof of Proposition 2.1.3 . . . 89
2.3.3 Proof of Proposition 2.1.4 . . . 92
2.3.4 Proof of Proposition 2.1.5 . . . 93
2.3.5 Proof of Proposition 2.1.6 . . . 96
2.3.6 Proof of Proposition 2.1.7 . . . 98
2.3.7 Proof of Proposition 2.2.1 . . . 101
2.4 Some computations for the numerical implementation . . . 103
2.4.1 Proximal point of $p \mapsto -\frac{S(x,t;p,a,b)}{\lambda}$ . . . 103
2.4.2 An equivalent expression for $S$ . . . 107
2.5 Summary . . . 109

**3 Lax-Oleinik-type representation formulas** **111**
3.1 Analytical solutions . . . 114
3.1.1 One-dimensional case . . . 115
3.1.2 Separable high-dimensional case . . . 123
3.1.3 General high-dimensional case . . . 125
3.2 Efficient algorithms and FPGA implementations . . . 128
3.2.1 Quadratic initial costs . . . 129
3.2.2 Convex initial costs . . . 136
3.2.3 Certain nonconvex initial costs . . . 147
3.3 Some proofs and technical lemmas for the analytical solutions . . . 158
3.3.1 Technical lemmas for Section 3.1 . . . 158
3.3.2 Proof of Lemma 3.1.1 . . . 162
3.3.3 Proof of Proposition 3.1.4 . . . 164
3.3.4 Proof of Proposition 3.1.6 . . . 166
3.3.5 Proof of Proposition 3.1.7 . . . 167
3.4 Some computations for the numerical implementation . . . 171
3.4.1 A numerical method for computing the building block . . . 171
3.4.2 An equivalent expression for $S$ and $\gamma$ . . . 175
3.5 Proofs of convergence results in Section 3.2 . . . 176
3.5.1 Proof of Proposition 3.2.1 . . . 176
3.5.2 Proof of Proposition 3.2.2 . . . 179
3.6 Summary . . . 180

**II Connection between the multi-time Hopf formula and learning problems** **182**

**4 Leveraging Multi-time Hamilton-Jacobi PDEs for Certain Scientific Machine Learning Problems** **184**
4.1 Introduction . . . 185
4.2 Generalized Hopf formula . . . 188
4.2.1 Introduction to the Hopf formula . . . 189
4.2.2 Connection between the Hopf formula and learning problems . . . 190
4.3 Linear Quadratic Regulator . . . 192
4.3.1 Introduction to the Linear Quadratic Regulator and Riccati equation . . . 193
4.3.2 Connection to single-point regularized linear regression problems 194
4.3.3 Connection to multi-point regularized linear regression problems 196
4.4 Methodology . . . 199
4.4.1 Solving the regularized linear regression problem using Riccati ODEs . . . 199
4.4.2 Adding or removing data . . . 202
4.4.3 Hyper-parameter tuning . . . 204
4.4.4 General convex regularization functions . . . 207
4.5 Numerical examples . . . 209

4.5.1 Function approximation in continual learning . . . . . 209
4.5.2 1D steady-state reaction-diffusion equation and post-training calibration . . . . . 213
4.5.3 Poisson equation using PINNs and transfer learning . . . . . 216
4.5.4 Identifying the dynamics of the Kraichnan-Orszag system from data . . . . . 220
4.6 Summary . . . . . 223

**5 Conclusion** **225**

# List of Tables


2.1 Time results for the HJ PDE (2.21) with quadratic initial condition. . 38

2.2 Time results for the HJ PDE (2.21) with initial condition $J(\boldsymbol{x}) = \sqrt{\langle \boldsymbol{x}, M\boldsymbol{x} \rangle}$. . . . . 43

2.3 Time results for the HJ PDE (2.21) with initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$. . . . . 44

2.4 Time results for the HJ PDE (2.21) with initial condition $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} \{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j\}$. . . . . 51

2.5 FPGA resources and latencies for evaluating the solution of the HJ PDE (2.21) with quadratic initial condition (2.3). . . . . 53

2.6 Comparison of the CPU and FPGA timing results for evaluating the solution of the HJ PDE (2.21) with quadratic initial condition (2.3). . 53

3.1 Comparison of the CPU and FPGA timing results for evaluating the solution of the HJ PDE (3.18) with quadratic initial condition (3.3). . 132

3.2 FPGA resources and latencies for evaluating the solution of the HJ PDE (3.18) with quadratic initial condition (3.3). . . . . 132

3.3 Results for a high throughput FPGA implementation of ADMM for evaluating the solution of the HJ PDE (3.18) with initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$. . . . . 141

3.4 Results for a low latency FPGA implementation of ADMM for evaluating the solution of the HJ PDE (3.18) with initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$. . . . . 142

3.5 Comparison of the CPU and FPGA timing results for evaluating the solution of the HJ PDE (3.18) with initial condition $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} \{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j\}$. . . . . 156

3.6 FPGA resources and latencies for evaluating the solution of the HJ PDE (3.18) with initial condition $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} \{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j\}$. . . . . 156

4.1 Summary of the learning problems and corresponding numerical examples presented in Chapter 4. . . . . 200

4.2 Errors in computing the minimizer of the function approximation loss (4.2) using our Riccati-based approach. . . . . 213

4.3 Errors of the minimizer of (4.6) and the solution to the 2D Poisson equation (4.5) using transfer learning and our Riccati-based approach. 219
4.4 Results of sparse identification of the K-O system (4.7) using PDHG to minimize the $\ell_1$-regularized losses (4.9). . . . . 222
4.5 Errors of the solution of the system identified using PDHG compared to the true solution of the K-O system. . . . . 222

# List of Figures


2.1 An illustration of a two-dimensional slice of the sets $\Omega_1, \Omega_2, \Omega_3, \Omega_4, \Omega_5$ on the $tx$-plane. . . . . 16

2.2 Evaluation of the solution of the HJ PDE (2.21) with initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|^2$. . . . . 36

2.3 Evaluation of the solution of the HJ PDE (2.21) with initial condition $J(\boldsymbol{x}) = \sqrt{\langle \boldsymbol{x}, M\boldsymbol{x}\rangle}$. . . . . 41

2.4 Evaluation of the optimal trajectory of the optimal control problem (2.20) with initial cost $J(\boldsymbol{x}) = \sqrt{\langle \boldsymbol{x}, M\boldsymbol{x}\rangle}$. . . . . 42

2.5 Evaluation of the solution of the HJ PDE (2.21) with initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$. . . . . 44

2.6 Evaluation of the optimal trajectory of the optimal control problem (2.20) with initial cost $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$. . . . . 45

2.7 Evaluation of the solution of the HJ PDE (2.21) with initial condition $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}}\{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j\}$. . . . . 49

2.8 Evaluation of an optimal trajectory of the optimal control problem (2.20) with initial cost $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}}\{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j\}$. . . . . 50

3.1 An illustration of a two-dimensional slice of the three sets $\Omega_1, \Omega_2, \Omega_3$ on the $xt$-plane. . . . . 118

3.2 Evaluation of the solution of the HJ PDE (3.18) with initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|^2$. . . . . 130

3.3 Evaluation of the solution of the HJ PDE (3.18) with initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$. . . . . 139

3.4 Evaluation of the optimal trajectory of the optimal control problem (3.17) with initial cost $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$. . . . . 140

3.5 Evaluation of the solution of the HJ PDE (3.18) with initial condition $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}}\{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j\}$. . . . . 154

3.6 Evaluation of an optimal trajectory of the optimal control problem (3.17) with initial cost $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}}\{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j\}$. . . . . 155

4.1 Illustration of a connection between a regularized learning problem, the multi-time Hopf formula, and an optimal control problem. . . . . 185
4.2 Overview of the overall structure of Chapter 4, the learning problems considered, our new theoretical connection, and our new Riccati-based algorithm. . . . . 188
4.3 Mathematical formulation describing the connection between a regularized learning problem, the multi-time Hopf formula, and an optimal control problem. . . . . 191
4.4 Mathematical formulation describing the connection between a regularized linear regression problem, the multi-time Hopf formula, and a piecewise LQR problem with $A = \mathcal{Q} = 0$ and $\mathcal{N} = 0$ on each piece. . . 197
4.5 Mathematical formulation describing the connection between our model linear regression problem with quadratic regularization, the multi-time Hopf formula, and a piecewise LQR problem. . . . . 199
4.6 Evolution and continual learning of the function approximation learned using our Riccati-based approach as more data becomes available. . . . . 212
4.7 Results of adding data points and enforcing boundary conditions when solving the 1D steady-state reaction-diffusion equation (4.3). . . . . 216
4.8 Results of removing outliers when solving the 1D steady-state reaction-diffusion equation (4.3). . . . . 217
4.9 Results of changing the regularization weight when solving the 2D Poisson equation using PINNs and transfer learning. . . . . 220

# Chapter One

# Introduction

Hamilton-Jacobi (HJ) equations are a class of first-order nonlinear partial differential equations (PDEs) that was originally developed as an alternative formulation of classical mechanics [10]. Since then, HJ PDEs have been shown to have deep connections to many scientific disciplines beyond physics, including optimal control [8], differential games [51], imaging sciences [31, 37], and machine learning [34, 36, 32], among many other fields. In this dissertation, we focus on the connections between HJ PDEs, optimal control, and machine learning.

Optimal control problems are an important class of optimization problems with many applications, including robot manipulator control [91, 73, 82, 54, 23], humanoid robot control [50, 53, 85, 56, 52, 44], and trajectory planning [30, 121, 64, 42, 112, 90]. It is well-known that, under some assumptions, the viscosity solution to certain HJ PDEs is equivalent to the value function of the corresponding optimal control problems [8]. Furthermore, the optimal control in the optimal control problem can be recovered from the spatial gradient of the viscosity solution to the HJ PDE.

In Part I, we utilize this connection between optimal control problems and HJ PDEs to derive representation formulas for two classes of optimal control problems with non-quadratic, state-dependent running costs or non-smooth constraints on the control and their corresponding HJ PDEs. Representation formulas convert the solution to HJ PDEs (and their corresponding optimal control problems) to the solution of an optimization problem over $\mathbb{R}^n$. We leverage our derived representation formulas to develop efficient numerical algorithms based on optimization techniques to solve these problems in high dimensions. In general, solving optimal control problems and HJ PDEs is complicated computationally when the spatial dimension is high. For example, when the spatial dimension is greater than about five, standard grid-based numerical algorithms such as ENO [108], WENO [72], and DG [67] become infeasible since their complexity scales exponentially with the dimension, a phenomenon

known as the "curse of dimensionality" [11]. In each chapter of Part I, we show that our algorithms overcome the curse of dimensionality, and we provide numerical results when the spatial dimension is as high as 16. We also present implementations of our algorithms on both central processing units (CPUs) and field-programmable gate arrays (FPGAs) to demonstrate their potential for real-time applications.

Now consider the case where the time variable of the HJ PDE is a higher dimensional quantity; in other words, consider the extension of HJ PDEs to the multi-time case. Multi-time HJ PDEs were originally introduced in economics [118]. The solution to certain multi-time HJ PDEs was then shown to be able to be represented by a multi-time Hopf formula [95], which is a generalization of the Hopf formula from the single-time case and has been shown to have connections with imaging sciences [35].

In Part II, we establish a novel theoretical connection between specific optimization problems arising in machine learning and the multi-time Hopf formula. Through this connection, we increase the interpretability of the training process of certain machine learning applications by showing that when we solve these learning problems, we also solve a multi-time HJ PDE and, by extension, its corresponding optimal control problem. We then leverage our theoretical connection to adapt existing efficient numerical algorithms from optimal control to design new training approaches for machine learning. In the machine learning applications considered, the limiting dimension is the size of the training dataset. Our proposed training approaches allow the learned models to be continually updated without having to retrain the entire model or having access to all of the previous data. In particular, the complexity of these updates using our methods is independent of the size of the original training dataset. Thus, our proposed approaches provide potential computational and memory advantages over conventional learning approaches.

# Part I

# Efficient algorithms based on representation formulas for certain high-dimensional problems

In this part, we consider HJ PDEs of the following form:

$$\begin{cases} \dfrac{\partial S}{\partial t}(\boldsymbol{x},t) + H(\boldsymbol{x},t,\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)) = 0 & \boldsymbol{x}\in\mathbb{R}^n, t\in(0,+\infty), \\ S(\boldsymbol{x},0) = J(\boldsymbol{x}) & \boldsymbol{x}\in\mathbb{R}^n, \end{cases} \tag{1.1}$$

where the $H\colon \mathbb{R}^n \times [0,\infty) \times \mathbb{R}^n \to \mathbb{R}$ is the Hamiltonian and $J : \mathbb{R}^n \to \mathbb{R}$ is the initial condition. Under some assumptions, the viscosity solution to the above HJ PDE (1.1) is equivalent to the value function of the following continuous optimal control problem with finite time horizon $t \in (0,+\infty)$:

$$S(\boldsymbol{x},t) = \min\left\{ \int_0^t \ell(\boldsymbol{x}(s),s,\boldsymbol{\alpha}(s))ds + J(\boldsymbol{x}(0)) \right\}, \tag{1.2}$$

subject to the constraint that a trajectory $\boldsymbol{x}(\cdot)\colon [0,t] \to \mathbb{R}^n$ satisfies the following backward ordinary differential equation (ODE):

$$\begin{cases} \dot{\boldsymbol{x}}(s) = f(\boldsymbol{x}(s),s,\boldsymbol{\alpha}(s)) & s\in(0,t), \\ \boldsymbol{x}(t) = \boldsymbol{x}. \end{cases}$$

In optimal control, we refer to $\ell\colon \mathbb{R}^n \times [0,t] \times A \to \mathbb{R}$ as the running cost (where the control space $A$ is a subset of a Euclidean space), $J\colon \mathbb{R}^n \to \mathbb{R}$ as the initial cost, and the objective function of the minimization problem in (1.2) as the cost of a control $\boldsymbol{\alpha}\colon [0,t] \to A$ and the corresponding trajectory $\boldsymbol{x}(\cdot)$.

The relationships between these optimal control problems and HJ PDEs are well-known (see [8], for instance). In particular, the Hamiltonian $H$ in the HJ PDE (1.1) is related to the functions $f$ and $\ell$ in the optimal control problem (1.2) via

$$H(\boldsymbol{x},t,\boldsymbol{p}) = \sup_{\boldsymbol{\alpha}\in A}\{\langle f(\boldsymbol{x},t,\boldsymbol{\alpha}),\boldsymbol{p}\rangle - \ell(\boldsymbol{x},t,\boldsymbol{\alpha})\},$$

and the initial cost in the optimal control problem (1.2) is given by the initial data of the HJ PDE (1.1). Moreover, the optimal control in the optimal control problem (1.2) can be recovered from the spatial gradient $\nabla_{\boldsymbol{x}} S(\boldsymbol{x}, t)$ of the viscosity solution $S$ to the HJ PDE (1.1).

Note that we could instead relate the HJ PDE (1.1) to an optimal control problem with a terminal cost as follows. Consider the following continuous optimal control problem with finite terminal time $T$:

$$\tilde{S}(\boldsymbol{x}, \tau) = \min \left\{ \int_{\tau}^{T} \ell(\boldsymbol{x}(s), s, \boldsymbol{\alpha}(s)) ds + J(\boldsymbol{x}(T)) \right\}, \tag{1.3}$$

subject to the constraint that a trajectory $\boldsymbol{x}(\cdot)$ satisfies the following (forward) ODE:

$$\begin{cases} \dot{\boldsymbol{x}}(s) = f(\boldsymbol{x}(s), s, \boldsymbol{\alpha}(s)) \quad s \in (\tau, T), \\ \boldsymbol{x}(\tau) = \boldsymbol{x}. \end{cases}$$

As before, $\ell$ is the running cost and $\boldsymbol{\alpha}$ is a control, which takes values in the control space $A$. Unlike before, we now consider $J$ to the be terminal cost, instead of the initial cost. Consider the function $S(\boldsymbol{x}, \tau) := \tilde{S}(\boldsymbol{x}, T - \tau)$. Then, $S$ also solves the initial value HJ PDE (1.1) in the viscosity sense, where the Hamiltonian $H$, the running cost $\ell$, and the dynamics $f$ are now related by

$$H(\boldsymbol{x}, t, \boldsymbol{p}) = \sup_{\boldsymbol{\alpha} \in A} \{ \langle -f(\boldsymbol{x}, T - t, \boldsymbol{\alpha}), \boldsymbol{p} \rangle - \ell(\boldsymbol{x}, T - t, \boldsymbol{\alpha}) \},$$

the PDE (1.1) now holds for $t \in (0, T - \tau)$ instead of $t \in (0, +\infty)$, and the initial data of the HJ PDE (1.1) is now given by the terminal cost of the optimal control problem (1.3).

In general, handling state-dependent running costs (or equivalently, state-

dependent Hamiltonians) or constraints on the control are open problems in optimal control and HJ PDE theory. In the literature, such problems are often solved using approximations in a grid-free manner as follows. Instead of approximating the solution space by a finite-dimensional space (as grid-based methods do), grid-free methods approximate the original optimal control problem by some simpler, more easily computable optimal control problems. In doing so, the solution to the original problem is approximated using the solutions to the simpler ones. Thus, an important research direction is to enlarge the class of optimal control problems with easily computable solutions; such problems and their corresponding exact solvers can then serve as building blocks for solving more complicated optimal control problems. Some well-known techniques for solving optimal control problems that often serve as these building blocks include: the linear-quadratic regulator (LQR) [94, 125, 98, 30], which corresponds to optimal control problems with linear dynamics and certain quadratic running and initial costs; the Hopf and Lax-Oleinik representation formulas [31, 35, 38, 132], which correspond to optimal control problems whose running costs do not depend on the state variable; and the max-plus (or min-plus) technique [1, 2, 49, 55, 58, 98, 97, 99, 100], which corresponds to optimal control problems whose running costs or initial costs are the maximum (or minimum) of several simpler functions. However, there are still many more classes of optimal control problems that cannot be solved (exactly) using these techniques.

In Chapter 2, we consider a class of optimal control problems (and their corresponding HJ PDEs) with non-quadratic, state-dependent running costs. In Chapter 3, we consider a class of optimal control problems (and their corresponding HJ PDEs) with non-smooth constraints on the control. In each chapter, we derive analytical solutions to the optimal control problem and representation formulas for the corresponding HJ PDEs, which solve these problems exactly and provide theoretical

guarantees for the properties of the solutions. To our knowledge, no numerically-computable representation formulas for these classes of problems existed previously. As such, our results enlarge the class of easily computable optimal control problems, and thus, our proposed methods have the potential to serve as building blocks for solving more complicated optimal control problems.

Another key challenge in HJ PDE theory and optimal control is designing efficient numerical algorithms for solving these problems in high dimensions. Many practical engineering applications involve high dimensions. For example, robot manipulator control problems involve multiple joints and end effectors, each of which yields several degrees of freedom, including velocities, angles, and positions. In turn, each of these degrees of freedom results in a state variable. As a result, robot manipulator control problems generally have high-dimensional state spaces (e.g., with dimension greater than five). However, as discussed previously, standard grid-based numerical algorithms such as ENO [108], WENO [72], and DG [67] suffer from the curse of dimensionality and, thus, are computationally intractable when the dimension is high. Previously, several methods have been proposed to overcome the curse of dimensionality when solving high-dimensional optimal control problems and their associated HJ PDEs, which include, but are not limited to, optimization methods [31, 35, 38, 132, 37, 89], max-plus methods [1, 2, 49, 55, 58, 98, 97, 99, 100], tensor decomposition techniques [47, 66, 127], model order reduction [4, 86], polynomial approximation [78, 79], sparse grids [13, 57, 80], dynamic programming and reinforcement learning [3, 12, 133], and neural networks [6, 7, 46, 71, 62, 68, 69, 87, 105, 117, 120, 126, 92, 34, 36, 103, 104, 74, 75, 32, 107]. In each chapter of this part, we propose efficient numerical algorithms based on our analytical representation formulas, convex optimization techniques, and min-plus methods. We then show that our resulting algorithms overcome the curse of

dimensionality.

While, in scientific computing, efficiency is usually measured by computational time alone, many real-life optimal control applications (e.g., the control of autonomous vehicles) often require strict constraints on power/energy consumption and computational resource availability in addition to computational speed (which generally is required to be on the order of nanoseconds to microseconds). Thus, although CPU implementations are standard in scientific computing for their fast computations and high parallelizability, CPUs are generally unable to meet all of these constraints. In contrast, FPGAs offer more flexibility than CPUs in designing implementations that meet these constraints.

FPGAs are arrays of programmable logic blocks and memory elements connected by reconfigurable interconnects (we refer the reader to [81] for a brief overview of FPGAs). One of the key advantages of FPGAs is the ability to specify which and how many computational resources are used. These resources include general purpose logics such as flip flops (FFs) and lookup tables (LUTs), specialized arithmetic units such as digital signal processing units (DSPs), and memory such as Block Random Access Memory (BRAMs). When applicable, FPGAs can achieve comparable or faster computational speeds (typically 1-2 orders of magnitude greater) than CPUs, while also consuming less energy and having more flexibility in specifying resource usage. In each chapter of this part, we provide numerical results using both a CPU and an FPGA implementation. Although our CPU results already illustrate the efficiency of our numerical solvers in terms of computational time, our FPGA results demonstrate the potential performance boosts (i.e., at least an order of magnitude speedup in most cases) that FPGAs are able to achieve over CPUs, while also tracking the specific amounts of logic and memory resources consumed. As such, our numerical solvers show promise in their ability to solve certain high-dimensional optimal con-

trol applications involving state-dependent running costs and control constraints in real time.

While FPGAs have existed since the 1980s, there has been limited progress in utilizing FPGAs for scientific computing. For instance, much of the scientific computing literature involving FPGAs centers on implementations for signal processing [111, Chap. 28], [40, 113], neural network architectures [106], or computational linear algebra [41, 124, 134]. One of the main barriers to the progress of FPGAs in scientific computing is the difficulty in FPGA programming, which originally would require coding in hardware description languages (HDLs) like Verilog. Recently, some high-level synthesis (HLS) technology has been developed, which allows users to create FPGA implementations using C/C++ instead of working with the HDL directly [29]. In both chapters of this part, we design our FPGA implementations using Xilinx Vitis HLS 2022.2. Although HLS tools like Vitis HLS greatly improve the workflow and ease of FPGA design, developing FPGA implementations using these HLS tools still requires keeping the underlying hardware in mind (e.g., via coding style or software-specific pragmas) and the behavior of the implementations produced by these tools is still not fully understood. As such, when we present our FPGA implementations, we also discuss some of the challenges that arose in the design process and some possible solutions for overcoming those challenges.

# Chapter Two

---

# Hopf-type representation formulas

In this chapter, we consider a class of optimal control problems whose running costs consist of a quadratic on the control variable and a convex, non-negative, piecewise affine function on the state variable. In general, when the running cost and corresponding Hamiltonian depend on the state variable $\boldsymbol{x}$ (which occurs in many practical applications, including [82, 23, 52, 64, 54, 30]), there are no known representation formulas that are computable in high dimensions. Typically, Hopf and Lax-Oleinik formulas are used to represent the solution of HJ PDEs. While Hopf and Lax-Oleinik representation formulas are computationally tractable for solving high-dimensional optimal control problems (see [31, 35, 38, 132]), they only apply to state-independent Hamiltonians. Other approaches based on the Hopf and Lax-Oleinik formulas have been introduced (see, for example, [28]) to handle cases where there is a state dependence in the Hamiltonian, but these approaches require discretization in time and hence, do not solve the problems exactly.

In this chapter, we provide the analytical solution to a particular class of optimal control problems with state-dependent running costs as well as a Hopf-type representation formula for the corresponding HJ PDEs with state-dependent Hamiltonians. Moreover, we show that the analytical solutions to these problems with convex initial costs and certain non-convex initial costs are easily and efficiently computable in high dimensions using convex optimization algorithms and min-plus techniques. As a result, our proposed methods have the potential to serve as a building block for solving more complicated optimal control problems. More specifically, since the running cost is non-smooth with respect to the state variable, our proposed methods could be helpful in solving some non-smooth optimal control problems. The content of this chapter is joint work with Jérôme Darbon and Tingwei Meng [25]. My main contributions include the development of a prototype implementation of our numerical solvers in MATLAB (the results of which are not shown, but would look identical

to the figures in Section 2.2), the conceptualization of the CPU implementation, and the creation of the FPGA implementation.

The organization of this chapter is as follows. In Section 2.1, we present the class of optimal control problems and HJ PDEs considered in this chapter as well as the analytical solutions of these problems. More specifically, we analyze the one-dimensional problems in Section 2.1.1, we consider a class of separable high-dimensional problems in Section 2.1.2, and we provide the Hopf-type representation formula for the general high-dimensional case in Section 2.1.3. In Section 2.1.4, we use min-plus techniques to extend the Hopf-type formula from Section 2.1.3 to solve the general high-dimensional problem with a certain class of non-convex initial costs. In Section 2.2, we propose efficient numerical solvers for these problems and present some high-dimensional numerical results. Quadratic initial costs are considered in Section 2.2.1, and the corresponding numerical solver serves as a building block for the algorithm in Section 2.2.2, which handles more general convex initial costs. Then, in Section 2.2.3, we generalize our proposed algorithms from the previous sections to handle the class of non-convex initial costs discussed in Section 2.1.4. Several high-dimensional numerical results as well as the computational runtime for each example are provided in each of these subsections to illustrate the performance of our proposed algorithms. In Section 2.2.4, we present an implementation of the numerical solver from Section 2.2.1 on an FPGA and some corresponding numerical results, which demonstrate the promising performance boosts FPGAs are able to achieve over CPUs. Some proofs and technical lemmas are provided in Section 2.3, and some computations for the numerical implementation are provided in Section 2.4. Finally, in Section 2.5, we make some concluding remarks and list some possible future directions.

## 2.1 Analytical solutions

In this section, we provide the analytical solution to the following optimal control problem:

$$S(\boldsymbol{x},t)=\inf\left\{\int_0^t\left(\frac{1}{2}\|\dot{\boldsymbol{x}}(s)\|_{M^{-1}}^2-K(\boldsymbol{x}(s))\right)ds+J(\boldsymbol{x}(0))\colon \boldsymbol{x}(t)=\boldsymbol{x}\right\},\qquad(2.1)$$

where $t>0$ is the time horizon, $\boldsymbol{x}\in\mathbb{R}^n$ is the terminal position, $\boldsymbol{x}(\cdot)\colon[0,t]\to\mathbb{R}^n$ is a locally Lipschitz function, and $\dot{\boldsymbol{x}}(s)$ denotes its the derivative at time $s$, which exists at $s\in(0,t)$ almost everywhere. Here, $M$ is a positive definite matrix with $n$ rows and $n$ columns. The matrix $M$ and its inverse respectively define the norms $\|\cdot\|_M\colon\mathbb{R}^n\to\mathbb{R}$ and $\|\cdot\|_{M^{-1}}\colon\mathbb{R}^n\to\mathbb{R}$ by

$$\|\boldsymbol{v}\|_M:=\sqrt{\langle\boldsymbol{v},M\boldsymbol{v}\rangle},\quad \|\boldsymbol{v}\|_{M^{-1}}:=\sqrt{\langle\boldsymbol{v},M^{-1}\boldsymbol{v}\rangle}\quad\forall\,\boldsymbol{v}\in\mathbb{R}^n,$$

where $\langle\cdot,\cdot\rangle$ denotes the standard Euclidean inner product in $\mathbb{R}^n$. The potential energy is given by $K\colon\mathbb{R}^n\to(-\infty,0]$, which is a piecewise affine concave function satisfying some assumptions. The initial cost is given by the continuous function $J\colon\mathbb{R}^n\to\mathbb{R}$. In the remainder of this chapter, if not mentioned specifically, we use bold characters to denote high-dimensional vectors in $\mathbb{R}^n$, and we use $x_i$ to denote the $i$-th component of a high-dimensional vector $\boldsymbol{x}\in\mathbb{R}^n$. To avoid the ambiguity of a trajectory and a vector, we use $\boldsymbol{x}(\cdot)$ to denote the trajectory, which is a function of the time variable, and we use $\boldsymbol{x}$ to denote the vector.

The corresponding HJ PDE reads:

$$\begin{cases} \dfrac{\partial S}{\partial t}(\boldsymbol{x},t) + \dfrac{1}{2}\|\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)\|_M^2 + K(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \mathbb{R}^n, t \in (0,+\infty), \\ S(\boldsymbol{x},0) = J(\boldsymbol{x}) & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{2.2}$$

where the potential energy $K$, the matrix $M$, and the initial data $J$ are the corresponding quantities in (2.1).

In what follows, we provide analytical solutions to the optimal control problem (2.1) and the corresponding HJ PDE (2.2) under various assumptions. In Section 2.1.1, we solve the one-dimensional problems with convex initial cost $J$. Then, in Section 2.1.2, we use the functions defined in Section 2.1.1 to solve the high-dimensional problems, where $M$ is the identity matrix, $J$ is convex, and $K$ has a specific form. In Section 2.1.3, we solve the high-dimensional problems, where $J$ is convex and $M$ and $K$ satisfy more general assumptions. Finally, in Section 2.1.4, we consider a certain class of non-convex initial costs $J$, and we generalize the representation formulas provided in Sections 2.1.1, 2.1.2, and 2.1.3 to handle this case using min-plus techniques.

### 2.1.1 One-dimensional case

In this section, we solve the one-dimensional versions of the problems (2.1) and (2.2). More specifically, we consider the following one-dimensional optimal control problem:

$$S(x,t) = \inf\left\{ \int_0^t \left( \frac{(\dot{x}(s))^2}{2} - K(x(s)) \right) ds + J(x(0)) \colon x(t) = x \right\}, \tag{2.3}$$

where $K\colon \mathbb{R} \to (-\infty, 0]$ is the positively 1-homogeneous concave function defined by

$$K(x) = \begin{cases} -ax & x \geq 0, \\ bx & x < 0, \end{cases} \tag{2.4}$$

for some positive constants $a$ and $b$. The corresponding HJ PDE reads:

$$\begin{cases} \dfrac{\partial S}{\partial t}(x,t) + \dfrac{1}{2}(\nabla_x S(x,t))^2 + K(x) = 0 & x \in \mathbb{R}, t \in (0, +\infty), \\ S(x,0) = J(x) & x \in \mathbb{R}. \end{cases} \tag{2.5}$$

In this section, we provide the analytical solution to the one-dimensional optimal control problem (2.3). We also present a Hopf-type representation formula for the viscosity solution to the one-dimensional HJ PDE (2.5).

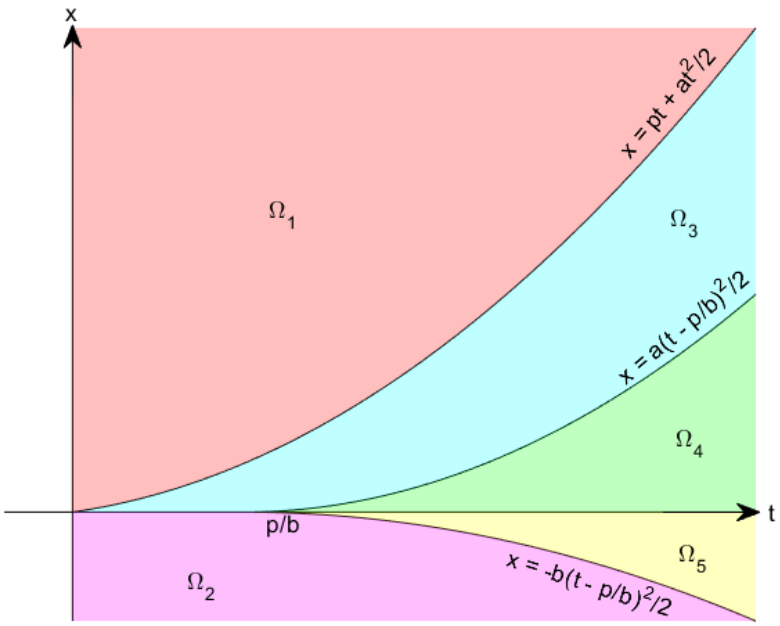



Figure 2.1: An illustration of a two-dimensional slice of the sets $\Omega_1, \Omega_2, \Omega_3, \Omega_4, \Omega_5$ on the $tx$-plane.

First, we consider the case when the initial cost $J$ is linear, i.e., when $J(x) = px$ holds for some $p \in \mathbb{R}$. In this case, we denote the solution to the HJ PDE (2.5) by

$\mathbb{R} \times [0, +\infty) \ni (x, t) \mapsto S(x, t; p, a, b)$, and we denote the optimal trajectory in (2.3) by $[0, t] \ni s \mapsto \gamma(s; x, t, p, a, b) \in \mathbb{R}$. The function $S$ is a function of $x$ and $t$, which are, respectively, the terminal position and the time horizon in (2.3). The function $S$ has three parameters: $p \in \mathbb{R}$, which is the slope of the initial cost $J$, and $a, b > 0$, which are the two positive parameters in the potential function $K$. The function $\gamma$ is a function of the current (running) time $s$ with five parameters: $x \in \mathbb{R}$, $t > 0$, $p \in \mathbb{R}$, and $a, b > 0$, which have the same meaning as the corresponding variables and parameters in the function $S$.

If $p \geq 0$, we define the function $\mathbb{R} \times [0, +\infty) \ni (x, t) \mapsto S(x, t; p, a, b) \in \mathbb{R}$ as follows:

$$S(x, t; p, a, b) := \begin{cases} f_1(x, t; p, a, b) & (x, t, p) \in \Omega_1, \\ f_2(x, t; p, a, b) & (x, t, p) \in \Omega_2, \\ f_3(x, t; p, a, b) & (x, t, p) \in \Omega_3, \\ f_4(x, t; p, a, b) & (x, t, p) \in \Omega_4, \\ f_5(x, t; p, a, b) & (x, t, p) \in \Omega_5, \end{cases} \tag{2.6}$$

where the five regions $\{\Omega_i\}_{i=1}^5 \subset \mathbb{R} \times [0, +\infty) \times [0, +\infty)$ are defined by

$$\begin{aligned} \Omega_1 &:= \left\{ (x, t, p) \in \mathbb{R} \times [0, +\infty) \times [0, +\infty) \colon x \geq pt + \frac{a}{2} t^2 \right\}, \\ \Omega_2 &:= \left\{ (x, t, p) \colon x < 0,\ t < \frac{p}{b} \right\} \bigcup \left\{ (x, t, p) \colon x < -\frac{b}{2} \left( t - \frac{p}{b} \right)^2,\ t \geq \frac{p}{b} \right\}, \\ \Omega_3 &:= \left\{ (x, t, p) \colon 0 \leq x < pt + \frac{a}{2} t^2,\ t < \frac{p}{b} \right\} \\ &\qquad \bigcup \left\{ (x, t, p) \colon \frac{a}{2} \left( t - \frac{p}{b} \right)^2 \leq x < pt + \frac{a}{2} t^2,\ t \geq \frac{p}{b} \right\}, \\ \Omega_4 &:= \left\{ (x, t, p) \in \mathbb{R} \times [0, +\infty) \times [0, +\infty) \colon 0 \leq x < \frac{a}{2} \left( t - \frac{p}{b} \right)^2,\ t \geq \frac{p}{b} \right\}, \\ \Omega_5 &:= \left\{ (x, t, p) \in \mathbb{R} \times [0, +\infty) \times [0, +\infty) \colon -\frac{b}{2} \left( t - \frac{p}{b} \right)^2 \leq x < 0,\ t \geq \frac{p}{b} \right\}, \end{aligned} \tag{2.7}$$

and the five functions $f_1, f_2, f_3, f_4, f_5$ are defined by

$$\begin{aligned}
f_1(x,t;p,a,b) &:= -\frac{a^2}{6}t^3 - \frac{a}{2}pt^2 + atx - \frac{1}{2}p^2t + px, \\
f_2(x,t;p,a,b) &:= -\frac{b^2}{6}t^3 + \frac{b}{2}pt^2 - \frac{1}{2}p^2t + px - btx, \\
f_3(x,t;p,a,b) &:= \frac{a+b}{3(a+2b)^2}\left((bt-p)^3 + ((bt-p)^2 + 2x(a+2b))^{3/2}\right) \\
&\quad - \frac{b}{a+2b}(bt-p)x - \frac{b^2}{6}t^3 + \frac{b}{2}pt^2 - \frac{1}{2}p^2t, \\
f_4(x,t;p,a,b) &:= \frac{a^2}{3}\left(\frac{2x}{a}\right)^{3/2} - \frac{p^3}{6b}, \\
f_5(x,t;p,a,b) &:= \frac{b^2}{3}\left(\frac{-2x}{b}\right)^{3/2} - \frac{p^3}{6b}.
\end{aligned} \tag{2.8}$$

An illustration of a two-dimensional slice of the sets $\Omega_i, i = 1, \ldots, 5$ for a fixed $p \geq 0$ is shown in Figure 2.1.

Now, we define the function $[0,t] \ni s \mapsto \gamma(s;x,t,p,a,b) \in \mathbb{R}$ for $p \geq 0$. We define the function $\gamma$ in five cases, which correspond to the five lines in (2.6), as follows:

1. When $(x,t,p) \in \Omega_1$ holds, which corresponds to the first line in (2.6), we define $\gamma(s;x,t,p,a,b)$ by

$$\gamma(s;x,t,p,a,b) := x - p(t-s) - \frac{a}{2}(t^2 - s^2) \qquad \forall s \in [0,t]. \tag{2.9}$$

   In this case, we have $\gamma(s;x,t,p,a,b) \geq 0$ for all $s \in [0,t]$.

2. When $(x,t,p) \in \Omega_2$ holds, which corresponds to the second line in (2.6), we define $\gamma(s;x,t,p,a,b)$ by

$$\gamma(s;x,t,p,a,b) := x - p(t-s) + \frac{b}{2}(t^2 - s^2) \qquad \forall s \in [0,t]. \tag{2.10}$$

   In this case, we have $\gamma(s;x,t,p,a,b) \leq 0$ for all $s \in [0,t]$.

3. When $(x,t,p) \in \Omega_3$ holds, which corresponds to the third line in (2.6), we define $\gamma(s;x,t,p,a,b)$ by

$$\gamma(s;x,t,p,a,b) := \begin{cases} -p(\tau - s) + \dfrac{b}{2}(\tau^2 - s^2) & s \in [0,\tau), \\ (p - b\tau)(s-\tau) + \dfrac{a}{2}(s-\tau)^2 & s \in [\tau,t], \end{cases} \tag{2.11}$$

where $\tau \in \mathbb{R}$ is defined by

$$\tau := \frac{(a+b)t + p - \sqrt{(bt-p)^2 + 2(2b+a)x}}{2b+a}. \tag{2.12}$$

By straightforward calculation using $0 \le x \le pt + \frac{a}{2}t^2$, we have $\tau \in [0,t]$, and the function $s \mapsto \gamma(s;x,t,p,a,b)$ is continuous in this case. Here, the trajectory $\gamma$ is divided into two parts: $\gamma(s;x,t,p,a,b) \le 0$ for $s \in [0,\tau)$ and $\gamma(s;x,t,p,a,b) \ge 0$ for $s \in [\tau,t]$.

4. When $(x,t,p) \in \Omega_4$ holds, which corresponds to the fourth line in (2.6), we define $\gamma(s;x,t,p,a,b)$ by

$$\gamma(s;x,t,p,a,b) := \begin{cases} -\dfrac{1}{2b}(p-bs)^2 & s \in \left[0, \dfrac{p}{b}\right), \\ 0 & s \in \left[\dfrac{p}{b}, t - \sqrt{\dfrac{2x}{a}}\right), \\ \dfrac{a}{2}\left(s - t + \sqrt{\dfrac{2x}{a}}\right)^2 & s \in \left[t - \sqrt{\dfrac{2x}{a}}, t\right]. \end{cases} \tag{2.13}$$

By straightforward calculation using $(x,t,p) \in \Omega_4$, we have $0 \le \frac{p}{b} \le t - \sqrt{\frac{2x}{a}} \le t$ and the function $s \mapsto \gamma(s;x,t,p,a,b)$ is continuous. Therefore, in this case, the trajectory $\gamma$ is divided into three parts: $\gamma(s;x,t,p,a,b) \le 0$ in the first time period $s \in [0, \frac{p}{b})$, it remains zero in the second time period $\left[\frac{p}{b}, t - \sqrt{\frac{2x}{a}}\right)$, and it becomes non-negative in the third time period $\left[t - \sqrt{\frac{2x}{a}}, t\right]$.

5. When $(x,t,p) \in \Omega_5$ holds, which corresponds to the fifth line in (2.6), we define $\gamma(s;x,t,p,a,b)$ by

$$\gamma(s;x,t,p,a,b) := \begin{cases} -\dfrac{1}{2b}(p-bs)^2 & s \in \left[0, \dfrac{p}{b}\right), \\ 0 & s \in \left[\dfrac{p}{b}, t - \sqrt{\dfrac{2|x|}{b}}\right), \\ -\dfrac{b}{2}\left(s - t + \sqrt{\dfrac{2|x|}{b}}\right)^2 & s \in \left[t - \sqrt{\dfrac{2|x|}{b}}, t\right]. \end{cases} \tag{2.14}$$

   By straightforward calculation using $(x,t,p) \in \Omega_5$, we have $0 \le \frac{p}{b} \le t - \sqrt{\frac{2|x|}{b}} \le t$ and the function $s \mapsto \gamma(s;x,t,p,a,b)$ is continuous. Therefore, in this case, the trajectory is divided into three parts: the trajectory $\gamma$ is negative in the first time period $\left[0, \frac{p}{b}\right)$, it remains zero in the second time period $\left[\frac{p}{b}, t - \sqrt{\frac{2|x|}{b}}\right)$, and it becomes non-positive again in the third time period $\left[t - \sqrt{\frac{2|x|}{b}}, t\right]$.

So far, we have provided the analytical solution of the one-dimensional optimal control problem (2.3) and the corresponding HJ PDE (2.5), with initial cost $J(x) = px$ for some $p \ge 0$. When the initial cost is $J(x) = px$ for some $p < 0$, we define the function $\mathbb{R} \times [0, +\infty) \ni (x,t) \mapsto S(x,t;p,a,b) \in \mathbb{R}$ by

$$S(x,t;p,a,b) := S(-x,t;-p,b,a) \quad \forall x \in \mathbb{R}, t \ge 0, \tag{2.15}$$

where $S(-x,t;-p,b,a)$ on the right-hand side is defined by (2.6) since $-p$ is positive. Similarly, the optimal trajectory $[0,t] \ni s \mapsto \gamma(s;x,t,p,a,b) \in \mathbb{R}$ for $p < 0$ is defined by

$$\gamma(s;x,t,p,a,b) := -\gamma(s;-x,t,-p,b,a) \quad \forall\, s \in [0,t], \tag{2.16}$$

where the right-hand side is well-defined using (2.9), (2.10), (2.11), (2.13), and (2.14) for different cases.

Now, we consider a general convex initial cost $J\colon \mathbb{R} \to \mathbb{R}$. The corresponding HJ PDE is solved using the following Hopf-type formula:

$$S(x,t) = \sup_{p\in\mathbb{R}}\{S(x,t;p,a,b) - J^*(p)\} \quad \forall\, x \in \mathbb{R},\, t \geq 0, \tag{2.17}$$

where the function $S(x,t;p,a,b)$ on the right-hand side is defined by (2.6) and (2.15), and the function $J^*$ on the right-hand side is the Legendre-Fenchel transform of the initial cost $J$. By a one-dimensional corollary of Lemma 2.3.9, the function value $S(x,t)$ defined in (2.17) is finite and the maximizer in (2.17) exists. Moreover, for any positive time horizon $t > 0$, the maximizer is unique and we denote the unique maximizer by $p^*(x,t) \in \mathbb{R}$. Then, the optimal trajectory $[0,t] \ni s \mapsto \gamma(s;x,t) \in \mathbb{R}$ is defined by

$$\gamma(s;x,t) := \gamma(s;x,t,p^*(x,t),a,b) \quad \forall s \in [0,t], \tag{2.18}$$

where the function $\gamma(s;x,t,p,a,b)$ on the right-hand side is defined by (2.9), (2.10), (2.11), (2.13), (2.14), and (2.16) for different cases of $x, t$ and $p$.

Next, we provide some theoretical properties for the functions $S$ and $\gamma$ defined above. In Proposition 2.1.1, we prove that, under some assumptions, the function $S$ defined above is indeed the unique viscosity solution to the HJ PDE (2.5). In Proposition 2.1.2, we show that the function $\gamma$ is the unique optimal trajectory in (2.3), whose optimal value equals $S(x,t)$.

*Proposition* 2.1.1. Let $J\colon \mathbb{R} \to \mathbb{R}$ be a convex function. Let $a, b > 0$ be positive constants and $K\colon \mathbb{R} \to \mathbb{R}$ be defined by (2.4) with parameters $a$ and $b$. Let $S\colon \mathbb{R}^n \times [0,+\infty) \to \mathbb{R}$ be the function defined in (2.17). Then, the following statements hold:

(a) The function $S$ is a continuously differentiable solution to the HJ PDE (2.5).

(b) If $J$ satisfies

$$|J(x) - J(y)| \leq C|x - y|(1 + |x|^{\delta} + |y|^{\delta}) \qquad \forall x, y \in \mathbb{R}, \tag{2.19}$$

for some constants $C \geq 0$ and $\delta \geq 0$, then the function $S$ is the unique viscosity solution to the HJ PDE (2.5) in the solution set $\mathcal{G}$ defined by

$$\mathcal{G} := \{W \in C(\mathbb{R} \times [0, +\infty)) \colon \|W\|_R < \infty \ \forall R > 0, \exists \alpha \in \mathbb{R} s.t. \ \forall T > 0,$$
$$\exists C_T \in \mathbb{R} \ s.t. \ W(x,t) - \alpha x \geq C_T \, \forall x \in \mathbb{R}, \forall t \in [0,T]\},$$

where $\|W\|_R$ is defined by $\|W\|_R := \sup\{|W(x,t)| + |q| \colon (x,t) \in B_R(\mathbb{R}) \times [0,R], q \in D_x^- W(x,t)\}$, the set $B_R(\mathbb{R})$ denotes the closed ball in $\mathbb{R}$ with center 0 and radius $R$, and $D_x^- W(x,t)$ denotes the subdifferential of $W$ with respect to $x$ at $(x,t)$.

*Proof.* This is a corollary of Proposition 2.1.3. □

*Proposition* 2.1.2. Let $J \colon \mathbb{R} \to \mathbb{R}$ be a convex function satisfying (2.19) and $K$ be the function defined in (2.4) with parameters $a, b > 0$. Let $x \in \mathbb{R}$ and $t > 0$. Then, the unique optimal trajectory for the optimal control problem (2.3) is given by the function $[0,t] \ni s \mapsto \gamma(s; x, t) \in \mathbb{R}$ defined in (2.18). Moreover, the optimal value of the optimal control problem (2.3) equals $S(x,t)$, as defined in (2.17).

*Proof.* This is a corollary of Proposition 2.1.4. □

*Remark* 2.1.1. If $J$ is a linear function, i.e., there exists a scalar $p \in \mathbb{R}$ such that $J(x) = px$ holds for all $x \in \mathbb{R}$, then $J$ satisfies the assumption (2.19). In this case, Proposition 2.1.1 shows that the function $(x,t) \mapsto S(x,t;p,a,b)$ defined in (2.6)

and (2.15) (where $p$ is the slope of $J$) is the unique continuously differentiable solution in the solution set $\mathcal{G}$ to the HJ PDE (2.5) with this linear initial data $J$. Moreover, Proposition 2.1.2 shows that the trajectory $s \mapsto \gamma(s; x, t, p, a, b)$ defined by (2.9), (2.10), (2.11), (2.13), (2.14), and (2.16) for different cases is the unique optimal trajectory of the optimal control problem (2.3), whose optimal value equals $S(x, t; p, a, b)$.

### 2.1.2 Separable high-dimensional case

In this section, we consider a special case of the high-dimensional problems (2.1) and (2.2). To be specific, we consider the following high-dimensional optimal control problem:

$$S(\boldsymbol{x}, t) = \inf \left\{ \int_0^t \left( \frac{1}{2} \|\dot{\boldsymbol{x}}(s)\|^2 - K(\boldsymbol{x}(s)) \right) ds + J(\boldsymbol{x}(0)) \colon \boldsymbol{x}(t) = \boldsymbol{x} \right\}. \tag{2.20}$$

In the optimal control problem (2.20), the initial cost $J \colon \mathbb{R}^n \to \mathbb{R}$ is a convex function, and the function $K \colon \mathbb{R}^n \to (-\infty, 0]$ satisfies

$$K(\boldsymbol{x}) = \sum_{i=1}^{n} K_i(x_i) \quad \forall\, \boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n,$$

where each function $K_i \colon \mathbb{R} \to (-\infty, 0]$ is a positively 1-homogeneous concave function given by (2.4) with positive constants $a_i$ and $b_i$. The corresponding HJ PDE reads:

$$\begin{cases} \dfrac{\partial S}{\partial t}(\boldsymbol{x}, t) + \dfrac{1}{2} \|\nabla_{\boldsymbol{x}} S(\boldsymbol{x}, t)\|^2 + K(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \mathbb{R}^n, t \in (0, +\infty), \\ S(\boldsymbol{x}, 0) = J(\boldsymbol{x}) & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{2.21}$$

where the initial condition $J$ and the potential energy $K$ are the corresponding functions in (2.20).

We will see later that each component of the optimal trajectory is independent from each other as long as the initial momentum $\boldsymbol{p}^*$ is determined. In other words, the computation of the optimal trajectory can be done in parallel, and hence, we call this problem separable.

The solution $S\colon \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is defined by the following Hopf-type formula:

$$S(\boldsymbol{x}, t) := \sup_{\boldsymbol{p}\in\mathbb{R}^n} \left\{ \sum_{i=1}^{n} S(x_i, t; p_i, a_i, b_i) - J^*(\boldsymbol{p}) \right\} \quad \forall\, \boldsymbol{x} \in \mathbb{R}^n,\ t \geq 0, \tag{2.22}$$

where the function $(x_i, t) \mapsto S(x_i, t; p_i, a_i, b_i)$ on the right-hand side is defined in (2.6) and (2.15). By Lemma 2.3.9, the function value $S(\boldsymbol{x}, t)$ defined in (2.22) is finite and the maximizer in (2.22) exists. Moreover, for a positive time horizon $t > 0$, the maximizer is unique and we denote the unique maximizer by $\boldsymbol{p}^* = (p_1^*, \ldots, p_n^*) \in \mathbb{R}^n$. Define the trajectory $[0, t] \ni s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t) \in \mathbb{R}^n$ by

$$\boldsymbol{\gamma}(s; \boldsymbol{x}, t) := (\gamma(s; x_1, t, p_1^*, a_1, b_1), \ldots, \gamma(s; x_n, t, p_n^*, a_n, b_n)) \quad \forall\, s \in [0, t], \tag{2.23}$$

where the $i$-th element $\gamma(s; x_i, t, p_i^*, a_i, b_i)$ on the right-hand side is the one-dimensional trajectory defined in (2.9), (2.10), (2.11), (2.13), (2.14), and (2.16) for different cases of $x_i, t$ and $p_i^*$. Note that the components of $\boldsymbol{\gamma}(s; \boldsymbol{x}, t)$ are independent from each other, and hence they can be computed in parallel as long as $\boldsymbol{p}^*$ is known. Thus, we call this problem separable.

Now, we provide some theoretical guarantees for the functions $S$ and $\boldsymbol{\gamma}$ defined above. Proposition 2.1.3 shows that the function $S$ is indeed the unique viscosity

solution to the HJ PDE (2.21) under some assumptions. Moreover, Proposition 2.1.4 proves that the function $\boldsymbol{\gamma}$ is the unique optimal trajectory in (2.20) under some assumptions and that the corresponding optimal value equals $S(\boldsymbol{x}, t)$.

*Proposition* 2.1.3. Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function. Let $\{a_i, b_i\}_{i=1}^n$ be positive constants and $K_i\colon \mathbb{R} \to \mathbb{R}$ be defined by (2.4) with constants $a_i$ and $b_i$ for each $i \in \{1, \dots, n\}$. Let $S\colon \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$ be the function defined in (2.22). Then, the following results hold:

(a) The function $S$ is a continuously differentiable solution to the HJ PDE (2.21).

(b) Furthermore, assume $J$ satisfies

$$|J(\boldsymbol{x}) - J(\boldsymbol{y})| \leq C\|\boldsymbol{x} - \boldsymbol{y}\|(1 + \|\boldsymbol{x}\|^\delta + \|\boldsymbol{y}\|^\delta) \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \tag{2.24}$$

for some constants $C \geq 0$ and $\delta \geq 0$. Then, the function $S$ is the unique viscosity solution to the HJ PDE (2.21) in the solution set $\mathcal{G}$ defined by

$$\begin{aligned}\mathcal{G} := \{W \in C(\mathbb{R}^n \times [0, +\infty))\colon \|W\|_R < \infty \ \forall R > 0, \exists \boldsymbol{\alpha} \in \mathbb{R}^n \ s.t. \ \forall T > 0,\\ \exists C_T \in \mathbb{R} \ s.t. \ W(\boldsymbol{x}, t) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle \geq C_T \ \forall \boldsymbol{x} \in \mathbb{R}^n, \forall t \in [0, T]\},\end{aligned} \tag{2.25}$$

where $\|W\|_R$ is defined by

$$\|W\|_R := \sup\{|W(\boldsymbol{x}, t)| + \|\boldsymbol{q}\|\colon (\boldsymbol{x}, t) \in B_R(\mathbb{R}^n) \times [0, R], \boldsymbol{q} \in D_{\boldsymbol{x}}^- W(\boldsymbol{x}, t)\}, \tag{2.26}$$

where the set $B_R(\mathbb{R}^n)$ denotes the closed ball in $\mathbb{R}^n$ with center $\mathbf{0}$ and radius $R$ and $D_{\boldsymbol{x}}^- W(\boldsymbol{x}, t)$ denotes the subdifferential of $W$ with respect to $\boldsymbol{x}$ at $(\boldsymbol{x}, t)$.

*Proof.* The proof is provided in Section 2.3.2. □

*Proposition* 2.1.4. Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function satisfying (2.24). Let $a_1, \ldots, a_n, b_1, \ldots, b_n$ be positive constants and $K_i\colon \mathbb{R} \to \mathbb{R}$ be defined by (2.4) with constants $a = a_i$ and $b = b_i$ for each $i \in \{1, \ldots, n\}$. Then, for any $\boldsymbol{x} \in \mathbb{R}^n$ and $t > 0$, the unique optimal trajectory for the optimal control problem (2.20) is given by the function $[0, t] \ni s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t) \in \mathbb{R}^n$ defined in (2.23). Moreover, the optimal value of the optimal control problem (2.20) equals $S(\boldsymbol{x}, t)$ defined in (2.22).

*Proof.* The proof is provided in Section 2.3.3. □

*Remark* 2.1.2. For the special case when $J$ is a linear function, i.e., $J(\boldsymbol{x}) = \langle \boldsymbol{p}, \boldsymbol{x} \rangle$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and for some constant vector $\boldsymbol{p} = (p_1, \ldots, p_n) \in \mathbb{R}^n$, the function $J^*$ equals the indicator function of $\{\boldsymbol{p}\}$, and hence the solution to the corresponding HJ PDE (2.21) reads:

$$S(\boldsymbol{x}, t) = \sum_{i=1}^{n} S(x_i, t; p_i, a_i, b_i) \quad \forall \boldsymbol{x} \in \mathbb{R}^n, t \geq 0,$$

where the function $S$ in the summation on the right-hand side is defined by (2.6) and (2.15). In this case, the optimal trajectory of the corresponding optimal control problem (2.20) equals

$$\boldsymbol{\gamma}(s; \boldsymbol{x}, t) = (\gamma(s; x_1, t, p_1, a_1, b_1), \ldots, \gamma(s; x_n, t, p_n, a_n, b_n)) \quad \forall\, s \in [0, t],$$

where each component $\gamma(s; x_i, t, p_i, a_i, b_i)$ on the right-hand side is defined by (2.9), (2.10), (2.11), (2.13), (2.14), and (2.16) for different cases.

### 2.1.3 The general high-dimensional case

In this section, we provide the analytical solution to the high-dimensional problems (2.1) and (2.2) under more general assumptions. Assume there exists a vector $\boldsymbol{u}_0 \in \mathbb{R}^n$ such that the function $\boldsymbol{x} \mapsto K(\boldsymbol{x} + \boldsymbol{u}_0)$ is positively 1-homogeneous and there exists an invertible matrix $P$ with $n$ rows and $n$ columns whose column vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in \mathbb{R}^n$ satisfy

$$\{\boldsymbol{x} \in \mathbb{R}^n \colon K(\boldsymbol{x} + \boldsymbol{u}_0) \geq -1\} = \operatorname{co}\left(\bigcup_{j=1}^{n} \left\{\frac{1}{a_j}\boldsymbol{u}_j, -\frac{1}{b_j}\boldsymbol{u}_j\right\}\right), \tag{2.27}$$

for some positive scalars $a_i, b_i > 0, i \in \{1, \ldots, n\}$, where co $E$ denotes the convex hull of a set $E$.

We define the function $S \colon \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$ by the following Hopf-type formula:

$$S(\boldsymbol{x}, t) := \sup_{\boldsymbol{p} \in \mathbb{R}^n} \left\{\sum_{i=1}^{n} S\left((P^{-1}\boldsymbol{x} - P^{-1}\boldsymbol{u}_0)_i, t; p_i, a_i, b_i\right) - \tilde{J}^*(\boldsymbol{p})\right\}, \tag{2.28}$$

for any $\boldsymbol{x} \in \mathbb{R}^n, t \geq 0$, where the function $S$ in the summation on the right-hand side is defined by (2.6) and (2.15) and the function $\tilde{J}^*$ is the Legendre-Fenchel transform of the function $\tilde{J} \colon \mathbb{R}^n \to \mathbb{R}$, which is defined by

$$\tilde{J}(\boldsymbol{y}) := J(P\boldsymbol{y} + \boldsymbol{u}_0) \quad \forall\, \boldsymbol{y} \in \mathbb{R}^n. \tag{2.29}$$

By straightforward calculation, the function $\tilde{J}^*$ equals

$$\tilde{J}^*(\boldsymbol{p}) = J^*\left((P^{-1})^T\boldsymbol{p}\right) - \langle \boldsymbol{p}, P^{-1}\boldsymbol{u}_0 \rangle \quad \forall \boldsymbol{p} \in \mathbb{R}^n. \tag{2.30}$$

Hence, for any $\boldsymbol{x} \in \mathbb{R}^n$ and $t \geq 0$, the Hopf-type formula (2.28) can equivalently be

formulated as:

$$\begin{aligned} & S(\boldsymbol{x}, t) \\ &= \sup_{\boldsymbol{p}\in\mathbb{R}^n} \left\{ \sum_{i=1}^{n} S\left((P^{-1}\boldsymbol{x} - P^{-1}\boldsymbol{u}_0)_i, t; p_i, a_i, b_i\right) - J^*\left((P^{-1})^T\boldsymbol{p}\right) + \langle \boldsymbol{p}, P^{-1}\boldsymbol{u}_0\rangle \right\} \\ &= \sup_{\boldsymbol{q}\in\mathbb{R}^n} \left\{ \sum_{i=1}^{n} S\left((P^{-1}\boldsymbol{x} - P^{-1}\boldsymbol{u}_0)_i, t; (P^T\boldsymbol{q})_i, a_i, b_i\right) - J^*(\boldsymbol{q}) + \langle \boldsymbol{q}, \boldsymbol{u}_0\rangle \right\}, \end{aligned} \tag{2.31}$$

where the second equality holds by the change of variable $\boldsymbol{q} = (P^{-1})^T\boldsymbol{p}$. Define the trajectory $[0, t] \ni s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t) \in \mathbb{R}^n$ by

$$\boldsymbol{\gamma}(s; \boldsymbol{x}, t) := P\tilde{\boldsymbol{\gamma}}(s; P^{-1}(\boldsymbol{x} - \boldsymbol{u}_0), t) + \boldsymbol{u}_0, \tag{2.32}$$

where the function $s \mapsto \tilde{\boldsymbol{\gamma}}(s; \boldsymbol{y}, t)$ for any $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$ and $t > 0$ is defined by

$$\tilde{\boldsymbol{\gamma}}(s; \boldsymbol{y}, t) := (\gamma(s; y_1, t, p_1^*, a_1, b_1), \ldots, \gamma(s; y_n, t, p_n^*, a_n, b_n)) \quad \forall s \in [0, t], \tag{2.33}$$

where $\boldsymbol{p}^* = (p_1^*, \ldots, p_n^*)$ is the maximizer in (2.28) and the $i$-th component $\gamma(s; y_i, t, p_i^*, a_i, b_i)$ on the right-hand side is the one-dimensional trajectory defined by (2.9), (2.10), (2.11), (2.13), (2.14), and (2.16) for different cases of $y_i, t$, and $p_i^*$. Note that for $t > 0$, by Lemma 2.3.1, the negative of the objective function in (2.28) is 1-coercive and strictly convex. Therefore, the optimal value $S(\boldsymbol{x}, t)$ in (2.28) is finite, and the maximizer $\boldsymbol{p}^*$ exists and is unique. Hence, the functions $S$ and $\boldsymbol{\gamma}$ are well-defined.

The following propositions show that the functions $\boldsymbol{\gamma}$ and $S$ defined above do indeed solve the problems (2.1) and (2.2). Proposition 2.1.5 proves that the function $S$ is the unique viscosity solution to the HJ PDE (2.2) under some assumptions.

Proposition 2.1.6 shows that the function $\boldsymbol{\gamma}$ is the unique optimal trajectory of the optimal control problem (2.1) under some assumptions and that the corresponding optimal value equals $S(\boldsymbol{x}, t)$.

*Proposition* 2.1.5. Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function. Let $K\colon \mathbb{R}^n \to (-\infty, 0]$ be a piecewise affine concave function. Assume there exists a vector $\boldsymbol{u}_0 \in \mathbb{R}^n$ such that $\boldsymbol{x} \mapsto K(\boldsymbol{x} + \boldsymbol{u}_0)$ is 1-homogeneous and there exists an invertible matrix $P$ whose column vectors $\boldsymbol{u}_1, \dots, \boldsymbol{u}_n \in \mathbb{R}^n$ satisfy (2.27) with the vector $\boldsymbol{u}_0$ and some positive scalars $a_1, \dots, a_n, b_1, \dots, b_n > 0$. Let $M$ be a matrix with $n$ rows and $n$ columns satisfying $M = PP^T$. Let $S\colon \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$ be the function defined in (2.28). Then, the function $S$ is a continuously differentiable solution to the HJ PDE (2.2). Moreover, if $J$ satisfies (2.24), the function $S$ is the unique viscosity solution to the HJ PDE (2.2) in the solution set $\mathcal{G}$ defined in (2.25).

*Proof.* The proof is provided in Section 2.3.4. □

*Remark* 2.1.3. Under the assumptions of Proposition 2.1.5, the set $\{\boldsymbol{x} \in \mathbb{R}^n\colon K(\boldsymbol{x} + \boldsymbol{u}_0) \geq -1\}$ (which is the shifted superlevel set of the function $K$) is a convex polyhedral with $2n$ vertices. Moreover, the matrix $M$ in the kinetic energy term $\frac{1}{2}\|\nabla_{\boldsymbol{x}} S(\boldsymbol{x}, t)\|_M^2$ is related to the extreme points (and hence the shape) of the set $\{\boldsymbol{x} \in \mathbb{R}^n\colon K(\boldsymbol{x} + \boldsymbol{u}_0) \geq -1\}$ in (2.27). For instance, if $M$ is a diagonal matrix, then the vectors $\boldsymbol{u}_1, \dots, \boldsymbol{u}_n$ are orthogonal to each other. In this case, the $2n$ vertices of $\{\boldsymbol{x} \in \mathbb{R}^n\colon K(\boldsymbol{x} + \boldsymbol{u}_0) \geq -1\}$ can be grouped pairwise into $n$ groups, where the $i$-th group contains the points $\frac{\boldsymbol{u}_i}{a_i}$ and $-\frac{\boldsymbol{u}_i}{b_i}$. Then, by connecting the two vertices in each group, we obtain $n$ line segments that are pairwise orthogonal to each other.

*Proposition* 2.1.6. Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function satisfying (2.24). Let the function $K\colon \mathbb{R}^n \to (-\infty, 0]$, the matrices $P$ and $M$, the vector $\boldsymbol{u}_0 \in \mathbb{R}^n$, and the scalars $a_1, \dots, a_n, b_1, \dots, b_n > 0$ satisfy the assumptions in Proposition 2.1.5. Then,

for any $\boldsymbol{x} \in \mathbb{R}^n$ and $t > 0$, the unique optimal trajectory of the optimal control problem (2.1) is given by the function $[0,t] \ni s \mapsto \boldsymbol{\gamma}(s;\boldsymbol{x},t) \in \mathbb{R}^n$ defined in (2.32). Moreover, the optimal value of the optimal control problem (2.1) equals $S(\boldsymbol{x},t)$ as defined in (2.28).

*Proof.* The proof is provided in Section 2.3.5. □

*Remark* 2.1.4. If the initial cost $J$ is a linear function given by $J(\boldsymbol{x}) = \langle \boldsymbol{p}, \boldsymbol{x} \rangle$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and for some constant vector $\boldsymbol{p} = (p_1, \ldots, p_n) \in \mathbb{R}^n$, then the solution $S$ to the corresponding HJ PDE (2.2) becomes

$$S(\boldsymbol{x},t) = \sum_{i=1}^{n} S\left((P^{-1}\boldsymbol{x} - P^{-1}\boldsymbol{u}_0)_i, t; p_i, a_i, b_i\right) \quad \forall\, \boldsymbol{x} \in \mathbb{R}^n, t \geq 0,$$

where the function $S$ in the summation on the right-hand side is defined by (2.6) and (2.15). The optimal trajectory of the optimal control problem (2.1) is defined by (2.32) and (2.33), where the point $\boldsymbol{p}^*$ equals $\boldsymbol{p}$, which is the slope of the linear initial cost $J$.

### 2.1.4 An extension to certain non-convex initial costs

In this section, we solve the high-dimensional problems (2.1) and (2.2) for certain non-convex initial data $J$. To be specific, we assume the function $J\colon \mathbb{R}^n \to \mathbb{R}$ satisfies

$$J(\boldsymbol{x}) := \min_{j \in \{1,\ldots,m\}} J_j(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \tag{2.34}$$

where $J_1, \ldots, J_m\colon \mathbb{R}^n \to \mathbb{R}$ are convex functions satisfying (2.24).

The solution $S\colon \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$ is defined by

$$S(\boldsymbol{x}, t) = \min_{j \in \{1,\dots,m\}} S_{J_j}(\boldsymbol{x}, t) \quad \forall\, \boldsymbol{x} \in \mathbb{R}^n,\, t \geq 0, \tag{2.35}$$

where for each $j \in \{1, \dots, m\}$, the function $S_{J_j}$ on the right-hand side is the solution defined by (2.28) (or (2.17) and (2.22) for special cases) with the initial data $J_j$. Similarly, the optimal trajectory $[0, t] \ni s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t) \in \mathbb{R}^n$ is defined by

$$\boldsymbol{\gamma}(s; \boldsymbol{x}, t) := \boldsymbol{\gamma}_{J_r}(s; \boldsymbol{x}, t) \quad \forall s \in [0, t], \qquad \text{where } r \in \operatorname*{arg\,min}_{j \in \{1,\dots,m\}} S_{J_j}(\boldsymbol{x}, t), \tag{2.36}$$

and the function $s \mapsto \boldsymbol{\gamma}_{J_r}(s; \boldsymbol{x}, t)$ is the trajectory defined by (2.32) (or (2.18) and (2.23) for special cases) with the initial cost $J_r$.

In the following proposition, we prove that the function $S$ defined above is the viscosity solution to the HJ PDE (2.2) and the value function of the optimal control problem (2.1) with initial data $J$. Moreover, we show that the trajectory $s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t)$ defined above is an optimal trajectory of the optimal control problem (2.1) with initial cost $J$.

*Proposition* 2.1.7. Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a continuous function of the form of (2.34) for some convex functions $J_1, \dots, J_m\colon \mathbb{R}^n \to \mathbb{R}$ satisfying (2.24). Assume the function $J$ is bounded below by an affine function. Let the function $K\colon \mathbb{R}^n \to (-\infty, 0]$, the matrices $P$ and $M$, the vector $\boldsymbol{u}_0 \in \mathbb{R}^n$, and the scalars $a_1, \dots, a_n, b_1, \dots, b_n > 0$ satisfy the assumptions in Proposition 2.1.5. Let $S$ be the function defined in (2.35) and $\boldsymbol{\gamma}$ be the trajectory defined in (2.36). Then, the following statements hold:

(a) The function $S$ is the unique viscosity solution in the solution set $\mathcal{G}$ in (2.25) to the HJ PDE (2.2) with initial data $J$.

(b) Let $\boldsymbol{x}$ be an arbitrary vector in $\mathbb{R}^n$ and $t > 0$ be an arbitrary positive number.

The trajectory $s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t)$ is an optimal trajectory of the optimal control problem (2.1) with initial cost $J$. Moreover, the optimal value of the optimal control problem (2.1) equals $S(\boldsymbol{x}, t)$.

*Proof.* The proof is provided in Section 2.3.6. □

*Remark* 2.1.5. The technique used in (2.35) is called the min-plus technique (or max-plus if the optimal control problem is formulated as a maximization problem). For more details, see [98, 84], for instance. The min-plus technique can also be applied to solve the problems in Sections 2.1.1 or 2.1.2. The unique viscosity solution and the optimal value are given by (2.35), where each $S_{J_j}$ is the solution to the corresponding HJ PDE in Sections 2.1.1 or 2.1.2 with initial data $J_j$. An optimal trajectory is given by (2.36), where the trajectory $s \mapsto \boldsymbol{\gamma}_{J_r}(s; \boldsymbol{x}, t)$ is the optimal trajectory of the corresponding optimal control problem in Sections 2.1.1 or 2.1.2 with initial cost $J_r$.

*Remark* 2.1.6. Note that the minimizer $r$ in (2.36) may be not unique. Whenever there are multiple minimizers, each minimizer $r$ gives an optimal trajectory $s \mapsto \boldsymbol{\gamma}_{J_r}(s; \boldsymbol{x}, t)$. The optimal trajectory of the optimal control problem (2.1) may be non-unique since (2.1) is not a convex optimization problem in this case (the initial cost $J$ is non-convex).

## 2.2 Efficient numerical algorithms

In this section, we present efficient numerical solvers based on some optimization methods that evaluate the optimal trajectory of the high-dimensional optimal control problem (2.20) as well as the solution of the corresponding high-dimensional HJ PDE (2.21). We note that the algorithms we present in this section can be extended to

solve (2.1) and (2.2), instead. To solve the more general problems in (2.1) and (2.2), we apply our algorithms to compute the optimizer $\boldsymbol{p}^*$ with the terminal position $\boldsymbol{x}$ replaced by $P^{-1}\boldsymbol{x} - P^{-1}\boldsymbol{u}_0$ and the Legendre transform of the initial cost $J^*$ replaced by $\tilde{J}^*$ as defined by (2.30). Then, we compute the optimal values and optimal trajectories using (2.28) and (2.32), respectively.

Recall that in Section 2.1, we provided representation formulas for the problems (2.20) and (2.21). Thus, we can numerically solve these problems using these representation formulas if the optimization problem in (2.22) is numerically solvable. In this section, we provide different methods to solve (2.22) for different classes of initial costs $J$. More specifically, in Section 2.2.1, we present efficient numerical solvers for quadratic initial costs based on explicit formulas for solving (2.22) exactly. In Section 2.2.2, we solve (2.22) with more general convex initial costs using optimization methods that utilize the numerical solver presented in Section 2.2.1 as a building block. For illustrative purposes, the alternating direction method of multipliers (ADMM) algorithm (see [59, 15]) is applied. However, we note that ADMM can be replaced by any other appropriate convex optimization algorithm. In Section 2.2.3, we extend our numerical methods to address the class of non-convex initial costs that are of the form of (2.34). In each of these three sections, we also present high-dimensional numerical examples and timing results, which demonstrate the efficiency of our proposed methods in each of these cases. All of the numerical examples in Sections 2.2.1- 2.2.3 are run using a C++ implementation on an 8th Gen Intel Laptop Core i5-8250U with a 1.60GHz processor. Finally, in Section 2.2.4, we describe a high throughput FPGA implementation of our building block from Section 2.2.1 and present some numerical results demonstrating the performance boost that can be obtained using FPGAs.

To avoid confusion, we use $S$ and $\boldsymbol{\gamma}$ to denote the analytical solutions to the HJ

PDEs and optimal control problems, while we use $\hat{S}$ and $\hat{\boldsymbol{\gamma}}$ to denote their numerical approximations as obtained by our proposed methods.

### 2.2.1 Quadratic initial costs

In this section, we solve the optimal control problem (2.20) and the HJ PDE (2.21) with quadratic initial cost defined by

$$J(\boldsymbol{x}) = \frac{1}{2\lambda}\|\boldsymbol{x} - \boldsymbol{y}\|^2 + \alpha \quad \forall \boldsymbol{x} \in \mathbb{R}^n,$$

where $\boldsymbol{y} \in \mathbb{R}^n$, $\lambda > 0$, and $\alpha \in \mathbb{R}$ are some parameters. Recall that $\|\cdot\|$ denotes the $\ell^2$-norm in $\mathbb{R}^n$. To solve these problems, we need to solve the optimization problem in (2.22). Then, the solution is given by the explicit formulas (2.22) and (2.23). By straightforward computation, the Legendre-Fenchel transform of $J$ is

$$J^*(\boldsymbol{p}) = \frac{\lambda}{2}\left\|\boldsymbol{p} + \frac{\boldsymbol{y}}{\lambda}\right\|^2 - \frac{\|\boldsymbol{y}\|^2}{2\lambda} - \alpha \quad \forall \boldsymbol{p} \in \mathbb{R}^n.$$

Therefore, for quadratic initial costs, the optimization problem in (2.22) is equivalent to

$$\min_{\boldsymbol{p}\in\mathbb{R}^n} \left\{ \sum_{i=1}^{n} \left( -S(x_i, t; p_i, a_i, b_i) + \frac{\lambda}{2}\left(p_i + \frac{y_i}{\lambda}\right)^2 \right) \right\}. \tag{2.1}$$

Note that the optimization problem (2.1) can be divided into $n$ one-dimensional subproblems since each term in the summation, which corresponds to each state dimension, is independent from each other. The $i$-th subproblem amounts to computing

$$p_i^* = \arg\min_{p\in\mathbb{R}} \left\{ -S(x_i, t; p, a_i, b_i) + \frac{\lambda}{2}\left(p + \frac{y_i}{\lambda}\right)^2 \right\}, \tag{2.2}$$

which can be calculated using the numerical solver described in Section 2.4.1 with parameters $x = x_i$, $a = a_i$, $b = b_i$, and $c = -\frac{y_i}{\lambda}$. Thus, solving (2.1) is embarrassingly parallel. We also note that the minimizer in (2.2) is the proximal point of $p \mapsto -\frac{1}{\lambda}S(x_i, t; p, a_i, b_i)$ at $-\frac{y_i}{\lambda}$. This relation suggests that our proposed numerical solver for quadratic initial costs may be a useful building block in proximal point-based methods for solving the problems with more general initial costs.

Now, we apply our proposed numerical solver to solve the HJ PDE (2.21) with the following quadratic initial cost:

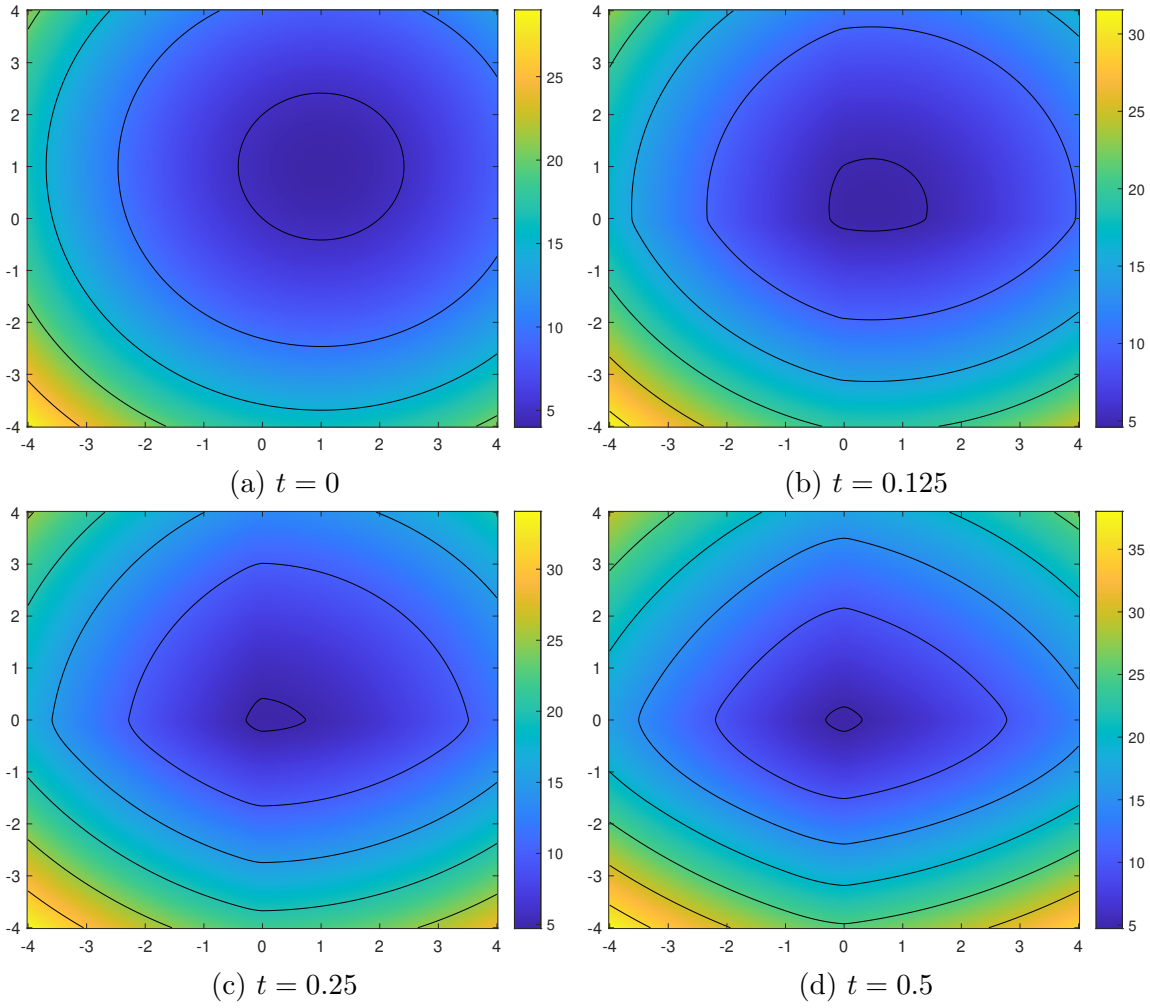
$$J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|^2 \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \tag{2.3}$$

i.e., we set $\lambda = 1$, $\alpha = 0$, and $\boldsymbol{y} = \mathbf{1}$, where $\mathbf{1}$ denotes the vector in $\mathbb{R}^n$ whose elements are all one. We also define the parameters $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{R}^n$ and $\boldsymbol{b} = (b_1, \ldots, b_n) \in \mathbb{R}^n$ by

$$a_i = \begin{cases} 4 & \text{if } i = 1, \\ 6 & \text{if } i = 2, \\ 5 & \text{if } i > 2, \end{cases} \qquad \text{and} \qquad b_i = \begin{cases} 3 & \text{if } i = 1, \\ 9 & \text{if } i = 2, \\ 6 & \text{if } i > 2. \end{cases} \tag{2.4}$$

In Figure 2.2, we show the numerical solution to the HJ PDE (2.21) in dimension $n = 10$. More specifically, Figure 2.2 depicts two-dimensional contour plots of the solution $\hat{S}(\boldsymbol{x}, t)$ for $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$ and different times $t$. The running time for this example in different dimensions is shown in Table 2.1. To compute the running time, we first compute the overall running time for computing the solution at $102,400$ random points $(\boldsymbol{x}, t) \in [-4, 4]^n \times [0, 0.5]$ and then report the average running time for computing the solution $\hat{S}(\boldsymbol{x}, t)$ at one point $(\boldsymbol{x}, t)$ over these $102,400$ trials. From Table 2.1, we see that on average, it takes less than $2 \times 10^{-6}$ seconds to compute the

solution at one point in a 16-dimensional problem, which demonstrates the efficiency of our proposed solver even in high dimensions.



(a) $t = 0$ (b) $t = 0.125$

(c) $t = 0.25$ (d) $t = 0.5$

Figure 2.2: Evaluation of the solution $\hat{S}(\boldsymbol{x}, t)$ of the HJ PDE (2.21) with $\boldsymbol{a}$ and $\boldsymbol{b}$ defined in (2.4) and initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\mathbf{1}\|^2$ for $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0) \in \mathbb{R}^{10}$ and different times $t$. Plots for $t = 0$, 0.125, 0.25, and 0.5 are depicted in (a)-(d), respectively. Level lines are superimposed on the plots.

### 2.2.2 Convex initial costs

In this section, we solve the optimal control problem (2.20) and the corresponding HJ PDE (2.21) with convex initial costs. To solve these problems, we need to solve the optimization problem in the representation formula (2.22), which can be rewritten

**Algorithm 1:** An ADMM algorithm for solving the optimal control problem (2.20) and the corresponding HJ PDE (2.21) with convex initial cost.

**Inputs :** Parameters in the problem: $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, terminal position $\boldsymbol{x} \in \mathbb{R}^n$, time horizon $t > 0$, running time $s > 0$ of the trajectory, and Legendre transform $J^*$ of the convex initial cost $J$. Parameters for ADMM: $\lambda > 0$, initialization $\boldsymbol{d}^0 \in \mathbb{R}^n$, $\boldsymbol{w}^0 \in \mathbb{R}^n$, error tolerance $\epsilon > 0$.

**Outputs:** The optimal trajectory $\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t)$ in the optimal control problem (2.20) and the solution value $\hat{S}(\boldsymbol{x}, t)$ to the corresponding HJ PDE (2.21).

**for** $k = 1, 2, \ldots$ **do**

  Update $\boldsymbol{v}^{k+1} \in \mathbb{R}^n$ by

$$\boldsymbol{v}^{k+1} = \arg\min_{\boldsymbol{v} \in \mathbb{R}^n} \left\{ J^*(\boldsymbol{v}) + \frac{\lambda}{2} \left\| \boldsymbol{v} - \boldsymbol{d}^k + \boldsymbol{w}^k \right\|^2 \right\}. \tag{2.5}$$

  Update $\boldsymbol{d}^{k+1} \in \mathbb{R}^n$, where the $i$-th element $d_i^{k+1}$ is updated by

$$d_i^{k+1} = \arg\min_{d_i \in \mathbb{R}} \left\{ -S(x_i, t; d_i, a_i, b_i) + \frac{\lambda}{2} (v_i^{k+1} - d_i + w_i^k)^2 \right\}. \tag{2.6}$$

  Update $\boldsymbol{w}^{k+1} \in \mathbb{R}^n$ by

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \boldsymbol{v}^{k+1} - \boldsymbol{d}^{k+1}.$$

  **if** $\|\boldsymbol{v}^{k+1} - \boldsymbol{v}^k\|^2 \le \epsilon$, $\|\boldsymbol{d}^{k+1} - \boldsymbol{d}^k\|^2 \le \epsilon$, *and* $\|\boldsymbol{v}^{k+1} - \boldsymbol{d}^{k+1}\|^2 \le \epsilon$ **then**

    set $N = k + 1$ and $\boldsymbol{u}^N = \boldsymbol{v}^N$;

    break;

  **end**

**end**

Output the optimal trajectory by

$$\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t) = (\gamma(s; x_1, t, u_1^N, a_1, b_1), \ldots, \gamma(s; x_n, t, u_n^N, a_n, b_n)), \tag{2.7}$$

where the $i$-th component $\gamma(s; x_i, t, u_i^N, a_i, b_i)$ is defined in (2.9), (2.10), (2.11), (2.13), (2.14), and (2.16). Also, output the solution to the HJ PDE by

$$\hat{S}(\boldsymbol{x}, t) = \sum_{i=1}^{n} S\left(x_i, t; u_i^N, a_i, b_i\right) - J^*\left(\boldsymbol{u}^N\right), \tag{2.8}$$

where the $i$-th component $S\left(x_i, t; u_i^N, a_i, b_i\right)$ in the summation is defined in (2.6), (2.7), (2.8), and (2.15).

| **n** | 4 | 8 | 12 | 16 |
|---|---|---|---|---|
| **running time (s)** | 4.2330e-07 | 9.2605e-07 | 1.4404e-06 | 1.9732e-06 |

Table 2.1: Time results in seconds for the average time per call over $102,400$ trials for evaluating the solution of the HJ PDE (2.21) with quadratic initial condition (2.3) for various dimensions $n$.

as

$$S(\boldsymbol{x},t) = -\inf_{\boldsymbol{p}\in\mathbb{R}^n}\left\{-\sum_{i=1}^{n} S(x_i,t;p_i,a_i,b_i) + J^*(\boldsymbol{p})\right\}. \tag{2.9}$$

By Lemma 2.3.1, if $J$ is convex and $t>0$, then the optimization problem in (2.9) is a convex optimization problem with strictly convex, 1-coercive objective function. Thus, the optimal value in (2.9) is finite, and the minimizer exists and is unique. Furthermore, since the objective function is convex, (2.9) can be solved numerically using convex optimization algorithms. Following the discussion in Section 2.2.1, proximal point-based methods would be a reasonable approach. For illustrative purposes, we demonstrate how ADMM can be applied to solve (2.9) when $J^*$ has numerically-computable proximal point. The details of applying ADMM to this problem is described in Algorithm 1.

In each iteration of ADMM in Algorithm 1, we first update $\boldsymbol{v}^{k+1}$ using (2.5). By definition, $\boldsymbol{v}^{k+1}$ is the proximal point of the function $\frac{J^*}{\lambda}$ at the point $\boldsymbol{d}^k-\boldsymbol{w}^k$, which we denote by $\operatorname{prox}_{\frac{J^*}{\lambda}}(\boldsymbol{d}^k-\boldsymbol{w}^k)$. In some cases, this proximal point may not be easy to compute, but the proximal point of $\boldsymbol{x}\mapsto\frac{1}{\lambda}J(\lambda\boldsymbol{x})$ is easy to compute. In these cases, we can use Moreau's identity [102] to obtain $\boldsymbol{v}^{k+1}$ as follows:

$$\begin{aligned}\boldsymbol{v}^{k+1} &= \boldsymbol{d}^k-\boldsymbol{w}^k-\operatorname*{prox}_{\boldsymbol{x}\mapsto\frac{1}{\lambda}J(\lambda\boldsymbol{x})}(\boldsymbol{d}^k-\boldsymbol{w}^k)\\ &= \boldsymbol{d}^k-\boldsymbol{w}^k-\operatorname*{arg\,min}_{\boldsymbol{v}\in\mathbb{R}^n}\left\{J(\lambda\boldsymbol{v})+\frac{\lambda}{2}\left\|\boldsymbol{v}-\boldsymbol{d}^k+\boldsymbol{w}^k\right\|^2\right\}.\end{aligned} \tag{2.10}$$

Thus, we compute $\boldsymbol{v}^{k+1}$ either via (2.5) or via (2.10) using any appropriate optimization algorithm, where the choice of optimization algorithm may depend on the form

of $J^*$ or $J$, respectively.

In the second step of each iteration, we update $\boldsymbol{d}^{k+1}$ componentwise, as in (2.6), which can be done in parallel. More specifically, we compute (2.6) using the numerical solver in Section 2.4.1 with parameters $x = x_i$, $a = a_i$, $b = b_i$, and $c = v_i^{k+1} + w_i^k$. We note that updating $\boldsymbol{d}^{k+1}$ in Algorithm 1 is equivalent to solving (2.21) with quadratic initial cost $J(\boldsymbol{x}) = \frac{1}{2\lambda}\|\boldsymbol{x} + \lambda(\boldsymbol{v}^{k+1} + \boldsymbol{w}^k)\|^2$. Hence, the solver proposed in Section 2.2.1 serves as a building block for ADMM in Algorithm 1.

Finally, to recover the optimal trajectory in (2.20) and the viscosity solution to (2.21), we compute their approximations using (2.7) and (2.8), respectively. In these two formulas, we use $\boldsymbol{u}^N = \boldsymbol{v}^N$ to approximate the maximizer $\boldsymbol{u}^*$ in the original formulas (2.22) and (2.23).

In the following proposition, we prove that our numerical solutions $\hat{\boldsymbol{\gamma}}$ and $\hat{S}$ converge to their respective analytical solutions as the number of ADMM iterates approaches infinity.

*Proposition* 2.2.1. Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function and $\boldsymbol{a}, \boldsymbol{b}$ be two vectors in $(0, +\infty)^n$. Let $\boldsymbol{x}$ be any vector in $\mathbb{R}^n$ and $t > 0$ be any scalar. Let $S$ and $\boldsymbol{\gamma}$ be the functions defined in (2.22) and (2.23), respectively. Let $\lambda > 0$ and the initializations $\boldsymbol{d}^0, \boldsymbol{w}^0 \in \mathbb{R}^n$ be arbitrary parameters for Algorithm 1. Let $\hat{S}^N$ and $\hat{\boldsymbol{\gamma}}^N$ be the output solution and trajectory, respectively, from Algorithm 1 with iteration number $N$. Then, we have

$$\lim_{N\to\infty} \hat{S}^N(\boldsymbol{x}, t) = S(\boldsymbol{x}, t) \text{ and } \lim_{N\to\infty} \sup_{s\in[0,t]} \|\hat{\boldsymbol{\gamma}}^N(s; \boldsymbol{x}, t) - \boldsymbol{\gamma}(s; \boldsymbol{x}, t)\| = 0. \tag{2.11}$$

*Proof.* The proof is provided in Section 2.3.7. □

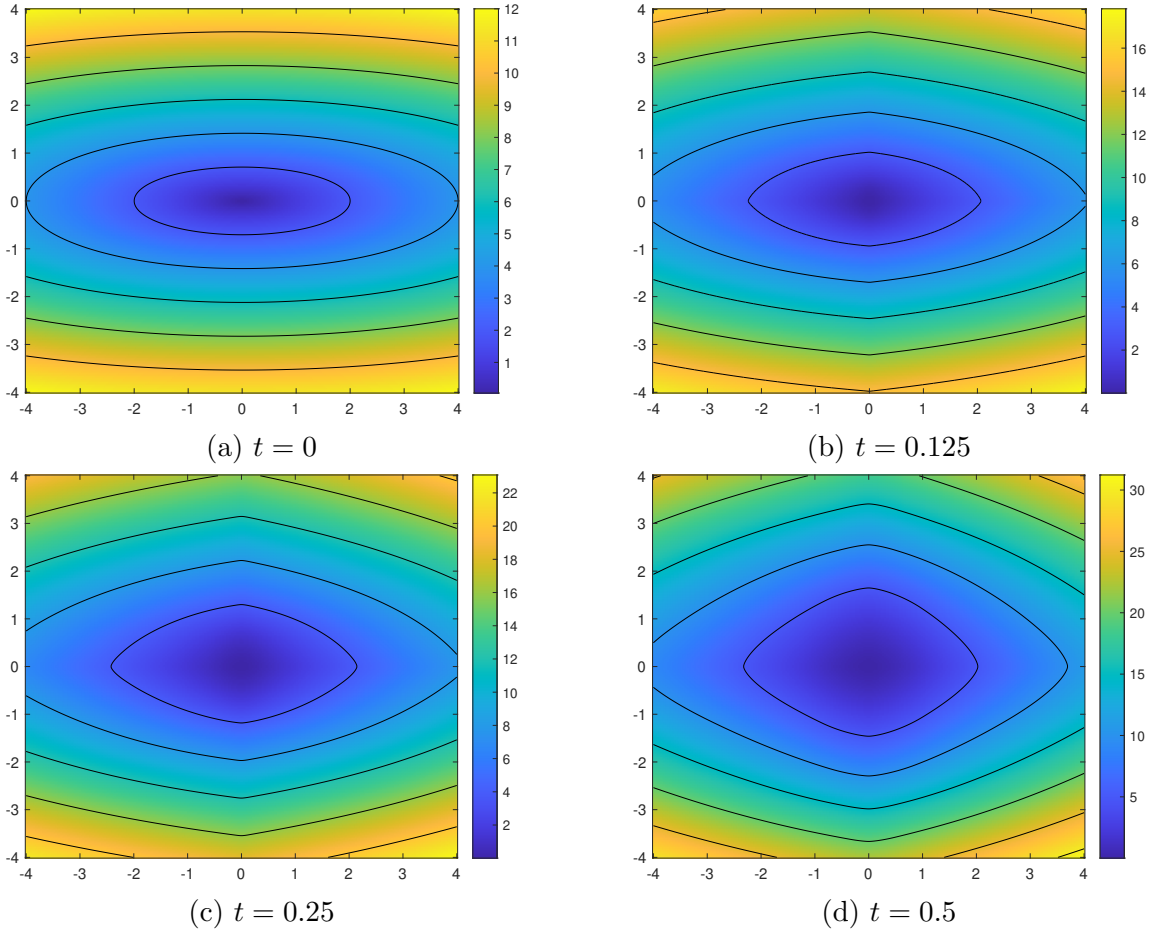
Now, we present two numerical results for the optimal control problem (2.20) and the HJ PDE (2.21) with convex initial cost $J$. For simplicity, we set the parameters in Algorithm 1 to be $\lambda = 1$, $\epsilon = 10^{-8}$, $\boldsymbol{d}^0 = \boldsymbol{x}$, and $\boldsymbol{w}^0 = \boldsymbol{0}$ in all of our numerical experiments.

We first consider the following initial cost

$$J(\boldsymbol{x}) = \sqrt{\langle \boldsymbol{x}, M\boldsymbol{x} \rangle} \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \tag{2.12}$$

where $M$ is a symmetric, positive definite matrix in $\mathbb{R}^{n \times n}$. Then, $J^*(\boldsymbol{p}) = I_{\varepsilon_M}(\boldsymbol{p})$, where $I_C$ is the indicator function defined by $I_C(\boldsymbol{x}) = 0$ if $\boldsymbol{x} \in C$ and $I_C(\boldsymbol{x}) = +\infty$ otherwise and $\varepsilon_M = \{\boldsymbol{x} \in \mathbb{R}^n : \langle \boldsymbol{x}, M^{-1}\boldsymbol{x} \rangle \leq 1\}$ is the ellipsoid associated with $M$. Thus, for this initial cost, $\boldsymbol{v}^{k+1}$ as defined in (2.5) is the projection of $\boldsymbol{d}^k - \boldsymbol{w}^k$ onto $\varepsilon_A$, which can be computed efficiently using the method described in [38, Section 4.3]. We set the parameters $\boldsymbol{a}$ and $\boldsymbol{b}$ to be the values defined in (2.4). For illustrative purposes, we set $M$ to be a diagonal matrix with diagonal elements $(m_{ii} : i = 1, \ldots, n) = (1, 8, 3, 5, 1, 1, \ldots, 1)$. Figure 2.3 depicts two-dimensional slices of the numerical solution $\hat{S}(\boldsymbol{x}, t)$ to the 10-dimensional HJ PDE (2.21) as computed using Algorithm 1 at different positions $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$ and different times $t$.

Figure 2.4 depicts one-dimensional slices of the optimal trajectory $\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t)$ of the corresponding optimal control problem (2.20) using different terminal positions $(x, -x, 0, \ldots, 0) \in \mathbb{R}^{10}$ and different time horizons $t$. In each subfigure, the time horizon $t$ is fixed, and the different trajectories correspond to different terminal positions. We observe that the one-dimensional slices are piecewise-quadratic and continuous in $s$, which is consistent with our formulas for $\gamma$ as defined in (2.9), (2.10), (2.11), (2.13),



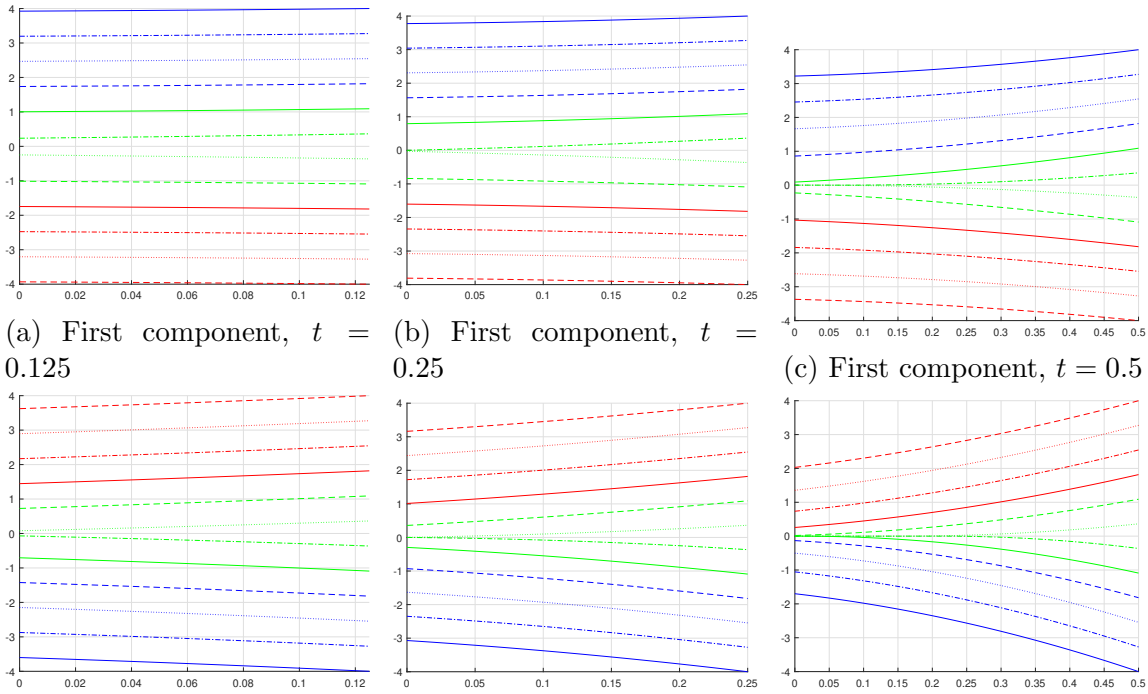
(a) $t = 0$ (b) $t = 0.125$

(c) $t = 0.25$ (d) $t = 0.5$

Figure 2.3: Evaluation of the solution $\hat{S}(\boldsymbol{x}, t)$ of the high-dimensional HJ PDE (2.21) with $\boldsymbol{a} = (4, 6, 5, \ldots, 5) \in \mathbb{R}^{10}$, $\boldsymbol{b} = (3, 9, 6, \ldots, 6) \in \mathbb{R}^{10}$, and initial condition $J(\boldsymbol{x}) = \sqrt{\langle \boldsymbol{x}, M\boldsymbol{x} \rangle}$, where $M$ is a diagonal matrix with diagonal elements $(m_{ii} : i = 1, \ldots, 10) = (1, 8, 3, 5, 1, 1, \ldots, 1)$, for $\boldsymbol{x} \in [-4, 4]^2 \times \{0\}^8$ and different times $t$. Plots for $t = 0$, 0.125, 0.25, and 0.5 are depicted in (a)-(d), respectively. Level lines are superimposed on the plots. The two axes in each figure correspond to the first and second components of the spatial variable $\boldsymbol{x} \in \mathbb{R}^{10}$.

and (2.14).

In Table 2.2, we show the running time of this example for different dimensions $n$. We use the same method to compute the running time as in Section 2.2.1. From Table 2.2, we see that it takes, on average, less than $3 \times 10^{-5}$ seconds to compute the solution at one point in a 16-dimensional problem, which demonstrates the efficiency of our proposed algorithm even in high dimensions.



(a) First component, $t = 0.125$ (b) First component, $t = 0.25$ (c) First component, $t = 0.5$

(d) Second component, $t = 0.125$ (e) Second component, $t = 0.25$ (f) Second component, $t = 0.5$

Figure 2.4: Evaluation of the optimal trajectory $\hat{\boldsymbol{\gamma}}(s; (x, -x, 0, \ldots, 0), t)$ of the optimal control problem (2.20) with $\boldsymbol{a} = (4, 6, 5, \ldots, 5) \in \mathbb{R}^{10}$, $\boldsymbol{b} = (3, 9, 6, \ldots, 6) \in \mathbb{R}^{10}$, and initial cost $J(\boldsymbol{x}) = \sqrt{\langle \boldsymbol{x}, M\boldsymbol{x} \rangle}$, where $M$ is a diagonal matrix with diagonal elements $(m_{ii} : i = 1, \ldots, 10) = (1, 8, 3, 5, 1, 1, \ldots, 1)$ versus $s \in [0, t]$ for different terminal positions $(x, -x, 0, \ldots, 0)$ $(x \in [-4, 4])$ and different time horizons $t$. The different colors and line markers simply differentiate between the different trajectories. Figures (a)-(c) depict the first component of the trajectory versus $s \in [0, t]$ with different time horizons $t$, while (d)-(f) depict the second component of the trajectory versus $s \in [0, t]$ with different time horizons $t$. Plots for time horizons $t = 0.125$, 0.25, and 0.5 are depicted in (a)/(d), (b)/(e), and (c)/(f), respectively.

| **n** | 4 | 8 | 12 | 16 |
|---|---|---|---|---|
| **running time (s)** | 6.9100e-06 | 9.7660e-06 | 1.5178e-05 | 2.1040e-05 |

Table 2.2: Time results in seconds for the average time per call over $102,400$ trials for evaluating the solution of the HJ PDE (2.21) with initial condition $J(\boldsymbol{x}) = \sqrt{\langle \boldsymbol{x}, M\boldsymbol{x} \rangle}$, where $M$ is a diagonal matrix with diagonal elements $(m_{ii} : i = 1, \ldots, n) = (1, 8, 3, 5, 1, 1, \ldots, 1)$, for various dimensions $n$.

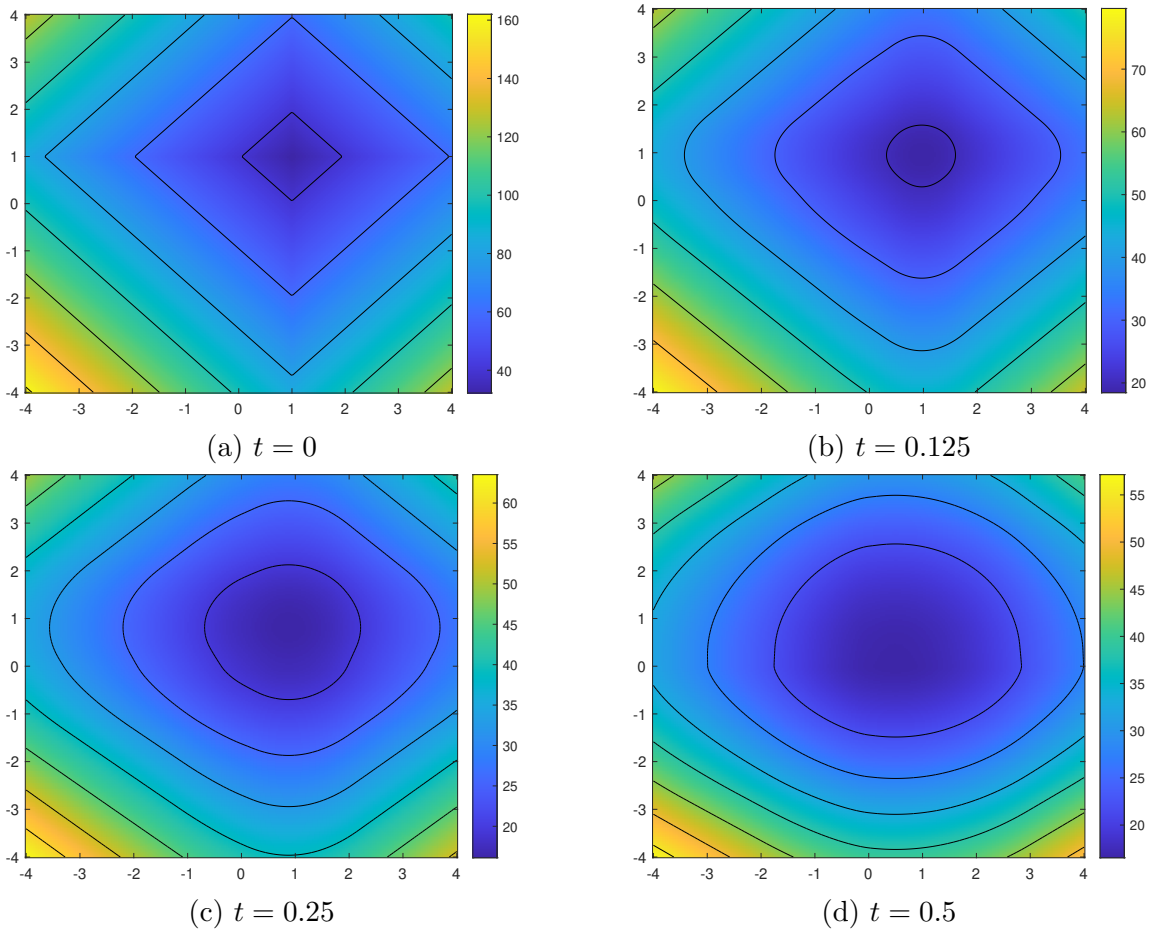
In the second example, we consider the nonsmooth convex initial cost

$$J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2 \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \tag{2.13}$$

where $\|\cdot\|_1$ denotes the $\ell^1$-norm in $\mathbb{R}^n$ and $\mathbf{1}$ is a vector in $\mathbb{R}^n$ whose components are all ones. For this example, we update $\boldsymbol{v}^{k+1}$ using (2.10), where the proximal point of the mapping $\boldsymbol{x} \mapsto \frac{1}{\lambda} J(\lambda \boldsymbol{x})$ is computed efficiently using the method described in [38, Section 4.4]. We set the parameters $\boldsymbol{a}$ and $\boldsymbol{b}$ to the values defined in (2.4).

Figure 2.5 depicts two-dimensional slices of the numerical solution $\hat{S}(\boldsymbol{x}, t)$, as computed using Algorithm 1, of the 10-dimensional HJ PDE (2.21) at different positions $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$ and at different times $t$. Figure 2.6 depicts one-dimensional slices of the optimal trajectory $\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t)$ of the corresponding optimal control problem (2.20), using different terminal positions $\boldsymbol{x} = (x, -x, 0, \ldots, 0) \in \mathbb{R}^{10}$ and different time horizons $t$. We observe that the one-dimensional slices are piecewise quadratic and continuous in $s$, which is consistent with our formulas for $\gamma$, given by (2.9), (2.10), (2.11), (2.13), and (2.14).

In Table 2.3, we present the running time of this example for different dimensions $n$. From Table 2.3, we see that it takes less than $4 \times 10^{-4}$ seconds on average to compute the solution at one point in a 16-dimensional problem, which demonstrates the efficiency of our proposed algorithm even in high dimensions.



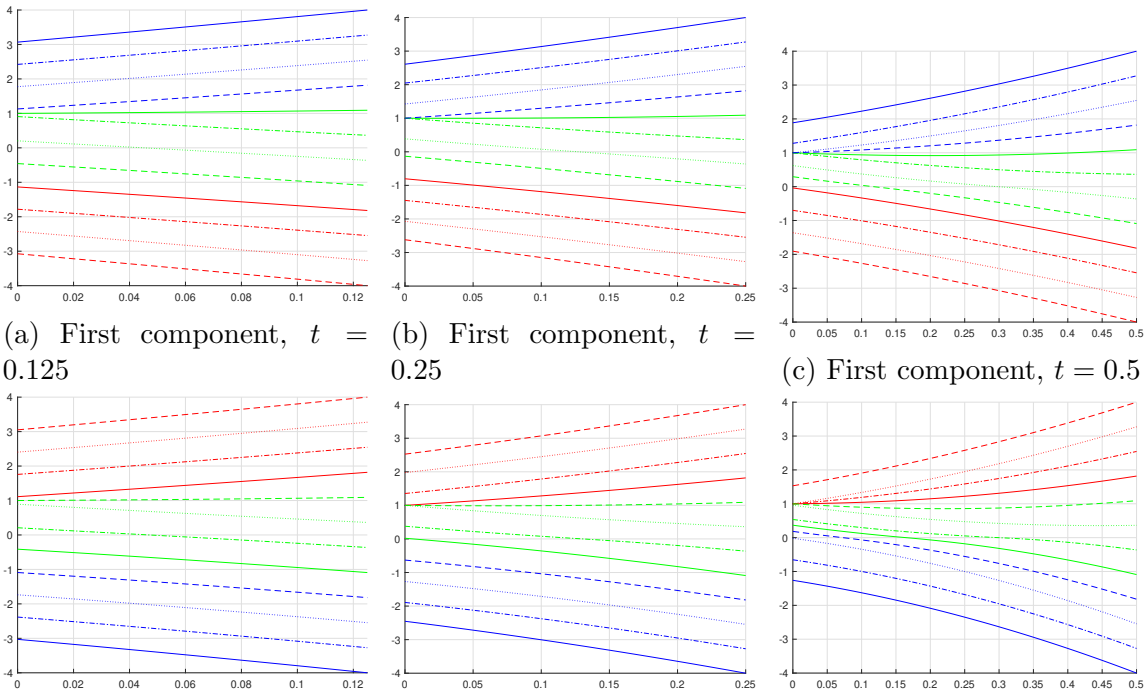
(a) $t = 0$ (b) $t = 0.125$

(c) $t = 0.25$ (d) $t = 0.5$

Figure 2.5: Evaluation of the solution $\hat{S}(\boldsymbol{x}, t)$ of the high-dimensional HJ PDE (2.21) with $\boldsymbol{a} = (4, 6, 5, \ldots, 5)$, $\boldsymbol{b} = (3, 9, 6, \ldots, 6)$, and initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$ for $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0) \in \mathbb{R}^{10}$ and different times $t$. Plots for $t = 0$, 0.125, 0.25, and 0.5 are depicted in (a)-(d), respectively. Level lines are superimposed on the plots.

| **n** | 4 | 8 | 12 | 16 |
|---|---|---|---|---|
| **running time (s)** | 2.1192e-05 | 9.4819e-05 | 2.0531e-04 | 3.2751e-04 |

Table 2.3: Time results in seconds for the average time per call over 102, 400 trials for evaluating the solution of the HJ PDE (2.21) with initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$ for various dimensions $n$.



(a) First component, $t = 0.125$

(b) First component, $t = 0.25$

(c) First component, $t = 0.5$

(d) Second component, $t = 0.125$

(e) Second component, $t = 0.25$

(f) Second component, $t = 0.5$

Figure 2.6: Evaluation of the optimal trajectory $\hat{\boldsymbol{\gamma}}(s;(x,-x,0,\dots,0),t)$ of the optimal control problem (2.20) with $\boldsymbol{a} = (4,6,5,\dots,5)$, $\boldsymbol{b} = (3,9,6,\dots,6)$, and initial cost $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\mathbf{1}\|_1^2$ versus $s \in [0,t]$ for different terminal positions $(x,-x,0,\dots,0)$ ($x \in [-4,4]$) and different time horizons $t$. The different colors and line markers simply differentiate between the different trajectories. Figures (a)-(c) depict the first component of the trajectory versus $s \in [0,t]$ with different time horizons $t$, while (d)-(f) depict the second component of the trajectory versus $s \in [0,t]$ with different time horizons $t$. Plots for time horizons $t = 0.125$, $0.25$, and $0.5$ are depicted in (a)/(d), (b)/(e), and (c)/(f), respectively.

### 2.2.3 A class of non-convex initial costs

**Algorithm 2:** An optimization algorithm for solving the optimal control problem (2.20) and the corresponding HJ PDE (2.21) with nonconvex initial cost $J$ of the form (2.34).

**Inputs :** Parameters $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, terminal position $\boldsymbol{x} \in \mathbb{R}^n$, time horizon $t > 0$, running time $s > 0$ of the trajectory, and the convex functions $J_1, \ldots, J_m$ in (2.34).

**Outputs:** An optimal trajectory $\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t)$ in the optimal control problem (2.20) and the solution value $\hat{S}(\boldsymbol{x}, t)$ to the corresponding HJ PDE (2.21) with nonconvex initial cost $J$ of the form (2.34).

1 **for** $j = 1, 2, \ldots, m$ **do**

2 Numerically solve the $j$-th subproblem (2.16) using any appropriate method (e.g., the solver in Section 2.2.1 or Algorithm 1 in Section 2.2.2), and get the optimal trajectory $\hat{\boldsymbol{\gamma}}_j(s; \boldsymbol{x}, t)$ of optimal control problem (2.20) and the solution $\hat{S}_j(\boldsymbol{x}, t)$ to the corresponding HJ PDE (2.21) with initial data $J_j$;

3 **end**

4 Compute the index $r$ by

$$r \in \underset{j \in \{1,\ldots,m\}}{\arg\min}\, \hat{S}_j(\boldsymbol{x}, t). \tag{2.14}$$

5 Output an optimal trajectory $\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t)$ and the solution value $\hat{S}(\boldsymbol{x}, t)$ using

$$\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t) = \hat{\boldsymbol{\gamma}}_r(s; \boldsymbol{x}, t), \qquad \hat{S}(\boldsymbol{x}, t) = \hat{S}_r(\boldsymbol{x}, t) = \min_{j \in \{1,\ldots,m\}} \hat{S}_j(\boldsymbol{x}, t). \tag{2.15}$$

In this section, we provide an algorithm based on the min-plus technique to solve the high-dimensional problems (2.20) and (2.21) with certain non-convex initial data of the form (2.34). The algorithm is summarized in Algorithm 2.

Recall that the solution $S$ is given by (2.35) and the optimal trajectory $\boldsymbol{\gamma}$ is given by (2.36). Thus, to solve (2.20) and (2.21) with initial cost $J$ of the form (2.34), we first divide the problem into $m$ subproblems. In the $j$-th subproblem, which

corresponds to the initial cost $J_j$, we must solve the following optimization problem:

$$\min_{\boldsymbol{p}\in\mathbb{R}^n}\left\{-\sum_{i=1}^{n} S(x_i,t;p_i,a_i,b_i)+J_j^*(\boldsymbol{p})\right\}. \tag{2.16}$$

We can apply any appropriate algorithm to solve this convex optimization problem, such as the methods in Sections 2.2.1 and 2.2.2. Note that the subproblems can be solved in parallel and possibly using different algorithms, the choice of which depends on the properties of $J_j$. We then compute the final solution to the overall problem using the solutions to each of the subproblems (i.e. the optimal cost $\hat{S}_j(\boldsymbol{x},t)$ and the optimal trajectory $\hat{\boldsymbol{\gamma}}_j(s)$ of the $j$-th subproblem for $j=1,\dots,m$) and (2.15). As defined in (2.14), the index $r$ in (2.15) is the index of the minimal cost $\hat{S}_r(\boldsymbol{x},t)$ among all possible costs $\hat{S}_1(\boldsymbol{x},t),\dots,\hat{S}_m(\boldsymbol{x},t)$. As noted in Remark 2.1.6, the index $r$ and hence the optimal trajectory $\boldsymbol{\gamma}(s;\boldsymbol{x},t)$ may be non-unique due to the nonconvexity of the initial data.

Note that [24, Proposition 8] still holds (since the proof only relies on the min-plus technique), and hence, the convergence of Algorithm 2 is guaranteed. In other words, as long as the algorithm for each subproblem converges, the numerical solution $\hat{S}$ given by Algorithm 2 converges pointwise to the analytical solution. Moreover, any cluster point of the numerical optimal trajectory $\hat{\boldsymbol{\gamma}}$ yields an optimal trajectory in (2.20). As noted previously, since the initial cost $J$ is non-convex, the optimal trajectory of the optimal control problem (2.20) may be non-unique, and thus, the output trajectory $s\mapsto\hat{\boldsymbol{\gamma}}(s;\boldsymbol{x},t)$ may have multiple cluster points. Therefore, the conclusion holds only for the cluster points of $\hat{\boldsymbol{\gamma}}$, and the convergence of $\hat{\boldsymbol{\gamma}}$ is not guaranteed. Specifically, whenever the optimal trajectory is unique, the output trajectories of Algorithm 2 converge as the error in each subproblem converges to zero.

Next, we present a high-dimensional numerical example using nonconvex $J$ of

the form (2.34). More specifically, we consider the following nonconvex initial cost:

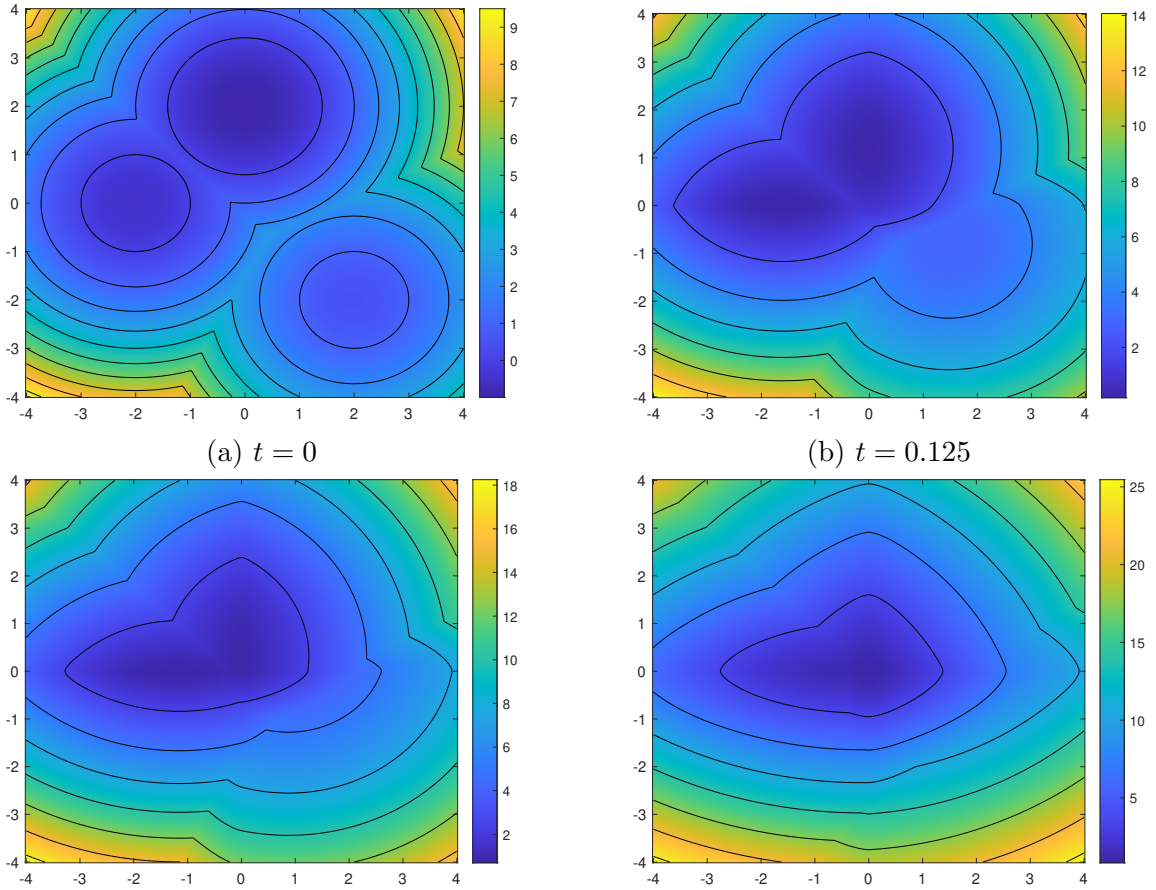
$$J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} \left\{ \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j \right\}, \tag{2.17}$$

where $\boldsymbol{y}_1 = (-2, 0, \ldots, 0)$, $\boldsymbol{y}_2 = (2, -2, -1, 0, \ldots, 0)$, and $\boldsymbol{y}_3 = (0, 2, 0, \ldots, 0)$ are vectors in $\mathbb{R}^n$ and $\alpha_1 = -0.5$, $\alpha_2 = 0$, $\alpha_3 = -1$ are scalars in $\mathbb{R}$. Recall that $\|\cdot\|$ denotes the $\ell^2$-norm in the Euclidean space $\mathbb{R}^n$. We also set $\boldsymbol{a}$ and $\boldsymbol{b}$ to be the vectors defined in (2.4).

Figure 2.7 depicts two-dimensional slices of the solution $\hat{S}(\boldsymbol{x}, t)$, as computed using Algorithm 2, to the 10-dimensional HJ PDE (2.21) for different positions $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$ and different times $t$. In Figure 2.7(a), we clearly see that the initial condition $J$ is not smooth at the interfaces of the quadratics $J_i$, e.g., there are obvious kinks near $(x_1, x_2) = (0, 0)$, $(-2, 2)$, $(0, -2)$, $(2.5, 0)$. We see that over time, the solution also evolves with several kinks (Figures 2.7(b)-(d)). These kinks provide numerical validation that the algorithm does indeed provide the non-smooth viscosity solution to the corresponding HJ PDE.

In Figure 2.8, we show several one-dimensional slices of an optimal trajectory $\hat{\boldsymbol{\gamma}}(s; (x, -x, 0, \ldots, 0), t)$ of the corresponding optimal control problem with $\boldsymbol{a}$ and $\boldsymbol{b}$ defined above and initial cost $J$ defined in (2.17), for different terminal positions $(x, -x, 0, \ldots, 0)$ and different time horizons $t$. We observe that the one-dimensional slices are piecewise quadratic and continuous in $s$, which is consistent with our formulas for $\gamma$ as given by (2.9), (2.10), (2.11), (2.13), and (2.14).

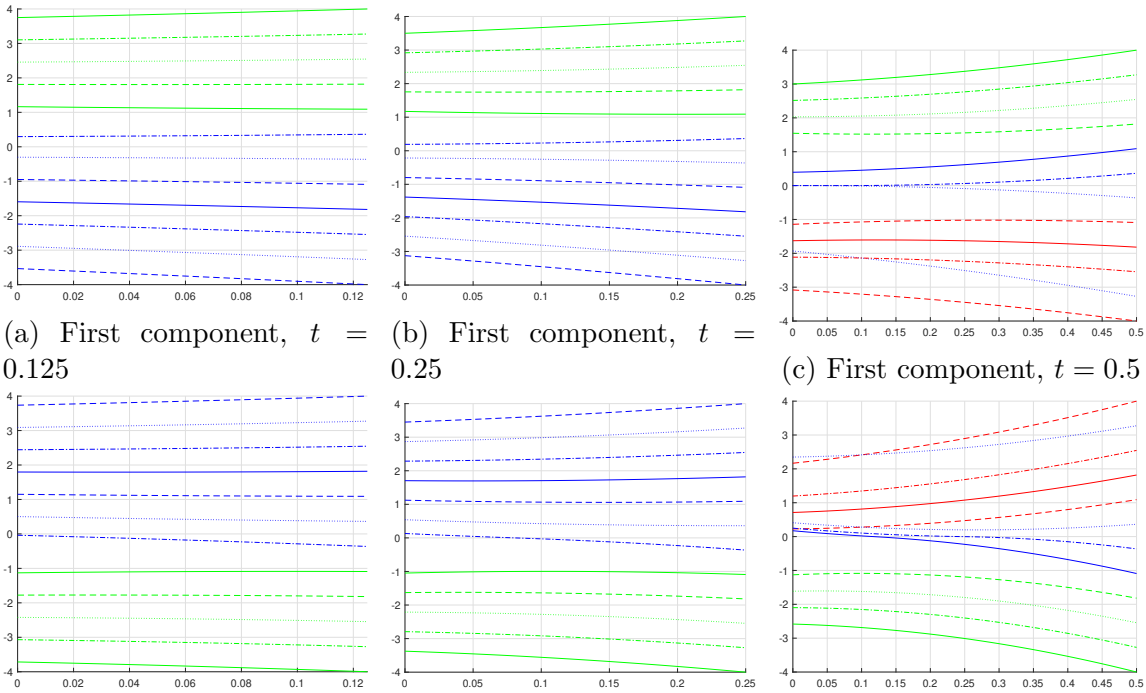
In Table 2.4, we present the running time of this example for different dimensions $n$. From Table 2.4, we see that it takes less than $5 \times 10^{-6}$ seconds on average to compute the solution at one point in a 16-dimensional problem, which demonstrates



(a) $t = 0$ (b) $t = 0.125$

(c) $t = 0.25$ (d) $t = 0.5$

Figure 2.7: Evaluation of the solution $\hat{S}(\boldsymbol{x}, t)$ of the 10-dimensional HJ PDE (2.21) with $\boldsymbol{a} = (4, 6, 5, \ldots, 5)$, $\boldsymbol{b} = (3, 9, 6, \ldots, 6)$, and initial condition $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}}\{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j\}$, where $\boldsymbol{y}_1 = (-2, 0, \ldots, 0)$, $\boldsymbol{y}_2 = (2, -2, -1, 0, \ldots, 0)$, $\boldsymbol{y}_3 = (0, 2, 0, \ldots, 0)$, $\alpha_1 = -0.5$, $\alpha_2 = 0$, and $\alpha_3 = -1$, for $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$ and different times $t$. Plots for $t = 0$, 0.125, 0.25, and 0.5 are depicted in (a)-(d), respectively. Level lines are superimposed on the plots.



(a) First component, $t = 0.125$ (b) First component, $t = 0.25$ (c) First component, $t = 0.5$

(d) Second component, $t = 0.125$ (e) Second component, $t = 0.25$ (f) Second component, $t = 0.5$

Figure 2.8: Evaluation of an optimal trajectory $\hat{\boldsymbol{\gamma}}(s; (x, -x, 0, \dots, 0), t)$ of the 10-dimensional optimal control problem (2.20) with $\boldsymbol{a} = (4, 6, 5, \dots, 5)$, $\boldsymbol{b} = (3, 9, 6, \dots, 6)$, and initial cost $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} \{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j\}$, where $\boldsymbol{y}_1 = (-2, 0, \dots, 0)$, $\boldsymbol{y}_2 = (2, -2, -1, 0, \dots, 0)$, $\boldsymbol{y}_3 = (0, 2, 0, \dots, 0)$, $\alpha_1 = -0.5$, $\alpha_2 = 0$, and $\alpha_3 = -1$, versus $s \in [0, t]$ for different terminal positions $(x, -x, 0, \dots, 0)$ $(x \in [-4, 4])$ and different time horizons $t$. The color of the lines denotes which initial cost was used, i.e., $r \in \arg\min_{j\in\{1,2,3\}} \hat{S}_j(\boldsymbol{x}, t)$ for $r = 1, 2, 3$ corresponds to red, green, and blue, respectively. The different line markers simply differentiate between the different trajectories. Figures (a)-(c) depict the first component of the trajectory versus $s \in [0, t]$ with different time horizons $t$, while (d)-(f) depict the second component of the trajectory versus $s \in [0, t]$ with different time horizons $t$. Plots for time horizons $t = 0.125$, 0.25, and 0.5 are depicted in (a)/(d), (b)/(e), and (c)/(f), respectively.

the efficiency of our proposed algorithm even in high dimensions.

| **n** | 4 | 8 | 12 | 16 |
|---|---|---|---|---|
| **running time (s)** | 9.9695e-07 | 2.2075e-06 | 3.3748e-06 | 4.4818e-06 |

Table 2.4: Time results in seconds for the average time per call over $102,400$ trials for evaluating the solution of the HJ PDE (2.21) with initial condition $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}}\{\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}_j\|^2 + \alpha_j\}$, where $\boldsymbol{y}_1 = (-2, 0, \ldots, 0)$, $\boldsymbol{y}_2 = (2, -2, -1, 0, \ldots, 0)$, $\boldsymbol{y}_3 = (0, 2, 0, \ldots, 0)$, $\alpha_1 = -0.5$, $\alpha_2 = 0$, and $\alpha_3 = -1$ for various dimensions $n$.

### 2.2.4 An FPGA implementation

In this section, we describe an FPGA implementation of our building block from Section 2.2.1, i.e., our efficient solver for computing (2.22), or equivalently (2.1), exactly. Specifically, we present an FPGA implementation with high throughput. Throughput refers to the amount of data that can be processed in a given amount of time. We achieve a high throughput by designing an implementation with an iteration interval (II) of 1, which means that we can begin processing a new input (in our case, a new $(x_i, t, y_i) \in \mathbb{R} \times [0, \infty) \times \mathbb{R}$ as defined in (2.1)) at every FPGA clock cycle.

Our FPGA implementation uses a Xilinx Alveo U280 board with a target design running at 300 MHz. Thus, having an II of 1 means that we can begin processing a new input (i.e., a new $(x_i, t, y_i)$) every 3.3333 nanoseconds. Note that the Alveo U280 board consists of three "chiplets". Crossing chiplets consumes limited routing resources, which can severely reduce the performance. To avoid this issue, we aim for an FPGA design that uses less than 30% of any given FPGA resource type to ensure that no chiplet is crossed.

Table 2.5 shows the amount of FPGA resources and the latencies used to imple-

ment and run our building block on a Xilinx Alveo U280 board for various dimensions $n$ and $102,400$ points $(\boldsymbol{x}, t, \boldsymbol{y}) \in \mathbb{R}^n \times [0, \infty) \times \mathbb{R}^n$. We observe that since our design streams the points $(\boldsymbol{x}, t, \boldsymbol{y})$ elementwise (i.e., our FPGA kernel takes the inputs $(x_i, t, y_i)$), the latency scales linearly in the dimension $n$ and the amount of FPGA resources used remains essentially constant in $n$. We also note that we use less than 30% of the FPGA resources available on the Xilinx Alveo U280 board, which implies that we could either parallelize by implementing multiple copies of our FPGA kernel to maximize usage of the FPGA board or use a smaller (i.e., cheaper) FPGA to implement our building block with similar performance as we report here.

In Table 2.6, we highlight the performance boost that can be achieved using FPGAs by comparing the performance of a CPU implementation of our building block using C++ to our FPGA implementation. The CPU implementation used here is identical to that used in Section 2.2.1 but using a fixed number of Newton iterations (see Section 2.4.1 for more details) for better comparability with our FPGA implementation. We observe that our FPGA implementation has a speedup of about 37 to 40 (depending on the dimension $n$) compared to the CPU implementation. Note that using a Xilinx Alveo U280 board, we could parallelize our FPGA implementation for a further speedup of $\times 3$ (i.e., a total speedup of about 111 to 122 overall depending on the dimension $n$) by simply replicating our FPGA kernel on each chiplet of the board. This parallelization would not experience any performance degradation due to chiplet crossing, as each copy of the FPGA kernel would be independent, and hence, no kernels/chiplets would need to communicate (as demonstrated in Table 2.5, each copy of the FPGA kernel would fit fully within a single chiplet since our FPGA kernel uses less than 30% of the available FPGA resources).

*Remark* 2.2.1. In order to achieve an II of 1, every for loop inside of our FPGA kernel must be unrolled completely. In terms of hardware, completely unrolling

| n | Latency (ns) | BRAM | DSPs | FFs | LUTs |
|---|---|---|---|---|---|
| 4 | 410,604 (1.369e06) | 0 (0%) | 2,042 (22%) | 418,105 (16%) | 154,471 (11%) |
| 8 | 820,218 (2.734e06) | 0 (0%) | 2,042 (22%) | 418,174 (16%) | 154,442 (11%) |
| 12 | 1,229,826 (4.099e06) | 0 (0%) | 2,042 (22%) | 418,434 (16%) | 154,465 (11%) |
| 16 | 1,639,438 (5.464e06) | 0 (0%) | 2,042 (22%) | 418,887 (16%) | 154,611 (11%) |

Table 2.5: FPGA resources and latencies in cycles and nanoseconds (ns) for evaluating the solution of the HJ PDE (2.21) with quadratic initial condition (2.3) at 102,400 points $(\boldsymbol{x}, t, \boldsymbol{y}) \in \mathbb{R}^n \times [0, \infty) \times \mathbb{R}^n$ for various dimensions $n$ using double precision floating points on a Xilinx Alveo U280 board with a frequency of 300 MHz.

| n | CPU time (s) | FPGA time (s) | Speedup |
|---|---|---|---|
| 4 | 4.9845e-07 | 1.3369e-08 | 37.2840 |
| 8 | 9.9128e-07 | 2.6699e-08 | 37.1280 |
| 12 | 1.6092e-06 | 4.0029e-08 | 40.2009 |
| 16 | 2.1701e-06 | 5.3359e-08 | 40.6698 |

Table 2.6: Comparison of the average time per call over 102,400 runs for evaluating the solution of the HJ PDE (2.21) with quadratic initial condition (2.3) for various dimensions $n$ using a CPU implementation on a single Intel Core i5-8250U versus an FPGA implementation on a Xilinx Alveo U280 board with a frequency of 300 MHz.

a loop means that each iterate of the loop is implemented with its own resources on the FPGA board. Following the discussion in Section 2.4, the only step in our implementation of the building block that requires a (completely unrolled) for loop is the computation of the candidate minimizer $p_3$ using Newton's method (2.4). Thus, in order to prevent an explosion in resource usage, we fix the number of Newton iterations to be 5. Since $p_3$ is the root of a relatively well-behaved function and we use a reasonable initialization, we find that 5 Newton iterations allows us to compute $p_3$ sufficiently accurately.

## 2.3 Some proofs and technical lemmas

### 2.3.1 Technical lemmas for Section 2.1

*Lemma* 2.3.1. Let $S$ be the function defined in (2.6) and (2.15). Let $x \in \mathbb{R}$ and $t, a, b > 0$. Then, the function $\mathbb{R} \ni p \mapsto -S(x,t;p,a,b) \in \mathbb{R}$ is strictly convex, 1-coercive, and continuously differentiable.

*Proof.* We first compute the derivatives of $f_1, f_2, f_3, f_4, f_5$ with respect to $p$ to obtain

$$\begin{aligned}
\frac{\partial f_1(x,t;p,a,b)}{\partial p} &= -\frac{at^2}{2} - pt + x,\\
\frac{\partial f_2(x,t;p,a,b)}{\partial p} &= \frac{bt^2}{2} - pt + x,\\
\frac{\partial f_3(x,t;p,a,b)}{\partial p} &= \frac{a+b}{(a+2b)^2}\left(-(bt-p)^2 + (p-bt)\sqrt{(bt-p)^2+2x(a+2b)}\right) \\
&\qquad + \frac{bx}{a+2b} + \frac{bt^2}{2} - pt,\\
\frac{\partial f_4(x,t;p,a,b)}{\partial p} &= \frac{\partial f_5(x,t;p,a,b)}{\partial p} = -\frac{p^2}{2b}.
\end{aligned} \tag{2.1}$$

From the above formulas, it is clear that $f_1, \dots, f_5$ are continuously differentiable with respect to $p$ in their domains. Now, we compute their second-order derivatives, which read as follows:

$$\begin{aligned}
\frac{\partial^2 f_1(x,t;p,a,b)}{\partial p^2} &= \frac{\partial^2 f_2(x,t;p,a,b)}{\partial p^2} = -t < 0,\\
\frac{\partial^2 f_3(x,t;p,a,b)}{\partial p^2} &= \frac{2(a+b)}{(a+2b)^2}\left(bt - p + \frac{(bt-p)^2 + x(a+2b)}{\sqrt{(bt-p)^2+2x(a+2b)}}\right) - t,\\
\frac{\partial^2 f_4(x,t;p,a,b)}{\partial p^2} &= \frac{\partial^2 f_5(x,t;p,a,b)}{\partial p^2} = -\frac{p}{b} \le 0.
\end{aligned} \tag{2.2}$$

Hence, the second-order derivatives of $f_1$ and $f_2$ with respect to $p$ are negative, which

implies that $f_1$ and $f_2$ are strongly concave with respect to $p$ in their domains. Note that the domains of $f_4$ and $f_5$ with respect to $p$ are included in $[0, +\infty)$, and hence, their second-order derivatives with respect to $p$ are negative almost everywhere. The strict concavity of $f_4$ and $f_5$ follows. Now, we prove the statement by considering different cases.

First, consider the case where $x \in [0, \frac{at^2}{2}]$. In this case, according to the definitions (2.6) and (2.15), the function $p \mapsto S(x, t; p, a, b)$ reads

$$S(x,t;p,a,b) = \begin{cases} f_2(-x,t;-p,b,a) & p < \sqrt{2ax} - at, \\ f_5(-x,t;-p,b,a) & \sqrt{2ax} - at \le p < 0, \\ f_4(x,t;p,a,b) & 0 \le p < b\left(t - \sqrt{\dfrac{2x}{a}}\right), \\ f_3(x,t;p,a,b) & p \ge b\left(t - \sqrt{\dfrac{2x}{a}}\right). \end{cases} \tag{2.3}$$

By straightforward calculation, the function $p \mapsto S(x, t; p, a, b)$ is continuously differentiable. Now, we prove the 1-coercivity of $-S$ by computing the limit of the derivatives as $p$ approaches $+\infty$ or $-\infty$. As $p$ approaches $-\infty$, we have

$$\lim_{p\to-\infty} \frac{\partial S(x,t;p,a,b)}{\partial p} = -\lim_{q\to+\infty} \frac{\partial f_2(-x,t;q,b,a)}{\partial q} = -\lim_{q\to+\infty} \left\{ \frac{at^2}{2} - qt - x \right\} = +\infty, \tag{2.4}$$

where the change of variable $q = -p$ is performed to obtain the first equality. Simi-

larly, when $p$ approaches $+\infty$, we obtain

$$\begin{aligned}
&\lim_{p\to+\infty}\frac{\partial S(x,t;p,a,b)}{\partial p}=\lim_{p\to+\infty}\frac{\partial f_3(x,t;p,a,b)}{\partial p}\\
&=\lim_{p\to+\infty}\frac{a+b}{(a+2b)^2}\left(-(bt-p)^2+(p-bt)\sqrt{(bt-p)^2+2x(a+2b)}\right)\\
&\qquad\qquad+\frac{bx}{a+2b}+\frac{bt^2}{2}-pt\\
&=\lim_{p\to+\infty}\frac{a+b}{(a+2b)^2}(p-bt)\frac{2x(a+2b)}{\sqrt{(bt-p)^2+2x(a+2b)}-bt+p}+\frac{bx}{a+2b}+\frac{bt^2}{2}-pt\\
&=\lim_{p\to+\infty}\frac{2x(a+b)}{a+2b}\frac{1}{\sqrt{1+\frac{2x(a+2b)}{(bt-p)^2}}+1}+\frac{bx}{a+2b}+\frac{bt^2}{2}-pt=-\infty,
\end{aligned}\tag{2.5}$$

where the last equality holds since we assume $t>0$. By (2.4) and (2.5), we conclude that $p\mapsto -S(x,t;p,a,b)$ is 1-coercive when $x\in[0,\frac{at^2}{2}]$. It remains to prove the strict concavity of $S$ with respect to $p$ in this case. Since $S$ is defined piecewise and continuously differentiable, it suffices to prove that each piece is strictly concave. Recall that we proved the strict concavity of $f_2, f_4, f_5$ using (2.2), and hence, the first three lines in (2.3) are strictly concave functions. It remains to consider $f_3$. Some computation shows that $\frac{\partial^2 f_3(x,t;p,a,b)}{\partial p^2}\le 0$ holds if and only if there holds

$$(a+b)^2x^2-t\left((a+b)p+\frac{a^2t}{4}\right)\left((p-bt)^2+2x(a+2b)\right)\le 0.\tag{2.6}$$

The left-hand side in (2.6) is a second-order polynomial with respect to $x$ with positive leading coefficient. As a result, to check that the inequality in (2.6) holds for all $x\in[0,\frac{at^2}{2}]$, it suffices to prove that the inequality holds at $x=0$ and $x=\frac{at^2}{2}$.

Denote the left-hand side of (2.6) by $P(x)$. Then, we have that

$$\begin{aligned} P(0) &= -t\left((a+b)p+\frac{a^2t}{4}\right)(p-bt)^2 \le 0, \\ P\left(\frac{at^2}{2}\right) &= -tp\left((a+b)(bt-p)^2+\frac{a^2pt}{4}+\left(\left(a+\frac{b}{2}\right)(a^2+2ab)+ab^2\right)t^2\right) \le 0, \end{aligned} \tag{2.7}$$

where $P(0) = 0$ holds if and only if $p = bt$ and where $P(\frac{at^2}{2}) = 0$ holds if and only if $p = 0$. In other words, $\frac{\partial^2 f_3(x,t;p,a,b)}{\partial p^2} \le 0$ holds in its corresponding domain $p > bt - b\sqrt{\frac{2x}{a}}$ except at finitely many points, which implies the strict concavity of $S$ with respect to $p$ in this domain. Therefore, we conclude that the function $p \mapsto -S(x,t;p,a,b)$ is strictly convex, 1-coercive, and continuously differentiable for $x \in [0, \frac{at^2}{2}]$.

Next, we consider the case where $x > \frac{at^2}{2}$. In this case, the function $p \mapsto S(x,t;p,a,b)$ reads:

$$S(x,t;p,a,b) = \begin{cases} f_2(-x,t;-p,b,a) & p < 0, \\ f_1(x,t;p,a,b) & 0 \le p \le \dfrac{x}{t} - \dfrac{at}{2}, \\ f_3(x,t;p,a,b) & p > \dfrac{x}{t} - \dfrac{at}{2}. \end{cases} \tag{2.8}$$

By straightforward calculation, $p \mapsto S(x,t;p,a,b)$ is continuously differentiable. Note that (2.4) and (2.5) still hold in this case, and hence, the function $p \mapsto -S(x,t;p,a,b)$ is 1-coercive. As in the first case, the strict concavity of the first two lines in (2.8) follows from (2.2). Hence, to prove the strict concavity of $p \mapsto S(x,t;p,a,b)$, it suffices to show that $\frac{\partial^2 f_3(x,t;p,a,b)}{\partial p^2} \le 0$ when $x > \frac{at^2}{2}$ and $p > \frac{x}{t} - \frac{at}{2}$, i.e., it suffices to check (2.6) for all $x \in (\frac{at^2}{2}, tp + \frac{at^2}{2})$. Again, denote the left-hand side of (2.6) by $P(x)$. Since $P(x)$ is a second-order polynomial with respect to $x$ with positive leading coefficient, it suffices to show that $P(x) \le 0$

at $x = \frac{at^2}{2}$ and $x = tp + \frac{at^2}{2}$. According to (2.7), $P(\frac{at^2}{2}) \leq 0$. After some calculations, we get that

$$P\left(tp + \frac{at^2}{2}\right) = -pt^3\left(b(a+b)^2 + \frac{a^2}{2}(a+b)\right) - p^2t^2\left((a+b)^2 + \frac{a^2}{4}\right) - (a+b)p^3t,$$

which is $\leq 0$ we have $a, b, p, t > 0$. Therefore, for $p > \frac{x}{t} - \frac{at}{2}$, the function $p \mapsto f_3(x, t; p, a, b)$ is strictly concave. As a result, we have shown that the function $p \mapsto -S(x, t; p, a, b)$ is strictly convex, 1-coercive, and continuously differentiable for $x \in (\frac{at^2}{2}, +\infty)$.

Now, we consider the case where $x < 0$. By (2.15), for all $x, p \in \mathbb{R}$, $t \geq 0$, we have $S(x, t; p, a, b) = S(-x, t; -p, b, a)$. Therefore, the conclusion for $x < 0$ also holds since we have already proved that the function $-p \mapsto S(-x, t; -p, b, a)$ is strictly convex, 1-coercive, and continuously differentiable. □

*Lemma* 2.3.2. Let $J\colon \mathbb{R} \to \mathbb{R}$ be defined by $J(x) = px$ for some $p \in \mathbb{R}$. Let $K\colon \mathbb{R} \to \mathbb{R}$ be the function defined by (2.4) for some constants $a, b > 0$. Let $S$ be the function defined in (2.6) and (2.15). Then, the following statements hold:

(a) The function $(x, t) \mapsto S(x, t; p, a, b)$ is a continuously differentiable solution to the corresponding HJ PDE (2.5).

(b) For all $t \geq 0$, the function $x \mapsto S(x, t; p, a, b)$ is convex.

*Proof.* (a) We first prove the statement for non-negative $p$. Assume $p \geq 0$. By

straightforward calculation, the derivatives of $f_3$ with respect to $x$ and $t$ read:

$$\begin{aligned}\frac{\partial f_3(x,t;p,a,b)}{\partial t} &= \frac{b(a+b)}{(a+2b)^2}\left((bt-p)^2+(bt-p)\sqrt{(bt-p)^2+2x(a+2b)}\right)\\ &\quad -\frac{b^2x}{a+2b}-\frac{b^2t^2}{2}+bpt-\frac{p^2}{2},\\ \frac{\partial f_3(x,t;p,a,b)}{\partial x} &= \frac{a+b}{a+2b}\sqrt{(bt-p)^2+2x(a+2b)}-\frac{b(bt-p)}{a+2b}.\end{aligned} \tag{2.9}$$

The derivatives of the functions $f_1, f_2, f_4, f_5$ read:

$$\begin{aligned}&\frac{\partial f_1(x,t;p,a,b)}{\partial t} = -\frac{a^2t^2}{2}-apt+ax-\frac{p^2}{2}, && \frac{\partial f_1(x,t;p,a,b)}{\partial x} = at+p,\\ &\frac{\partial f_2(x,t;p,a,b)}{\partial t} = -\frac{b^2t^2}{2}+bpt-bx-\frac{p^2}{2}, && \frac{\partial f_2(x,t;p,a,b)}{\partial x} = -bt+p,\\ &\frac{\partial f_4(x,t;p,a,b)}{\partial t} = 0, && \frac{\partial f_4(x,t;p,a,b)}{\partial x} = \sqrt{2ax},\\ &\frac{\partial f_5(x,t;p,a,b)}{\partial t} = 0, && \frac{\partial f_5(x,t;p,a,b)}{\partial x} = -\sqrt{-2bx}.\end{aligned} \tag{2.10}$$

It is straightforward to check that these five functions all satisfy the differential equation in (2.5). By straightforward calculation, the function $(x,t)\mapsto S(x,t;p,a,b)$ is continuously differentiable and satisfies the initial condition in (2.5). Therefore, $S$ is a continuously differentiable solution to the HJ PDE (2.5).

Next, we consider negative $p$. Assume $p<0$. For all $\alpha,\beta>0$, denote by $K_{\alpha,\beta}(x)$ the function $K$ defined in (2.4) with constants $a=\alpha$ and $b=\beta$. Note that there holds $K_{\alpha,\beta}(x)=K_{\beta,\alpha}(-x)$ for all $x\in\mathbb{R}$. By (2.15), we have

$$\begin{aligned}&\frac{\partial S(x,t;p,a,b)}{\partial t}+\frac{1}{2}(\nabla_x S(x,t;p,a,b))^2+K_{a,b}(x)\\ =\ &\frac{\partial S(-x,t;-p,b,a)}{\partial t}+\frac{1}{2}(-\nabla_x S(-x,t;-p,b,a))^2+K_{b,a}(-x)\\ =\ &\frac{\partial S(y,t;-p,b,a)}{\partial t}+\frac{1}{2}(\nabla_x S(y,t;-p,b,a))^2+K_{b,a}(y)\\ =\ &0,\end{aligned}$$

where the second equality follows from the change of variable $y = -x$ and the last equality holds since the function $(y,t) \mapsto S(y,t;-p,b,a)$ is a solution to the HJ PDE (2.5) with potential energy $K_{b,a}$, according to the proof above for positive $p$. We check the initial condition as follows:

$$S(x,0;p,a,b) = S(-x,0;-p,b,a) = (-p)(-x) = px.$$

Therefore, the function $(x,t) \mapsto S(x,t;p,a,b)$ defined in (2.15) solves (2.5). It is continuously differentiable since it equals $S(-x,t;-p,b,a)$, which is continuously differentiable with respect to $(-x,t)$.

(b) When $t = 0$, we have $S(x,t;p,a,b) = px$, which is convex with respect to $x$. It remains to consider the case when $t > 0$. Recall that we have proved above that the function $S$ is continuously differentiable. To show the convexity of $S$ with respect to $x$, it suffices to show the convexity of each $f_i$ in its domain. Let $p \geq 0$. By straightforward calculation, we obtain

$$\frac{\partial^2 f_1(x,t;p,a,b)}{\partial x^2} = \frac{\partial^2 f_2(x,t;p,a,b)}{\partial x^2} = 0, \qquad \frac{\partial^2 f_4(x,t;p,a,b)}{\partial x^2} = \frac{a}{\sqrt{2ax}} > 0,$$
$$\frac{\partial^2 f_3(x,t;p,a,b)}{\partial x^2} = \frac{a+b}{\sqrt{(bt-p)^2 + 2x(a+2b)}} > 0, \quad \frac{\partial^2 f_5(x,t;p,a,b)}{\partial x^2} = \frac{b}{\sqrt{-2bx}} > 0.$$

Therefore, the function $x \mapsto S(x,t;p,a,b)$ is convex for all $p \geq 0$. The convexity of $x \mapsto S(x,t;p,a,b)$ for $p < 0$ follows from (2.15). □

*Lemma* 2.3.3. Let $a, b > 0$ be any positive scalars and $S$ be the function defined in (2.6) and (2.15). Then, for all $p \in \mathbb{R}$, there exists a constant $C_p \in \mathbb{R}$, such that there holds

$$px + C_p \leq S(x,t;p,a,b) \leq px + t(a+b)|x| \quad \forall x \in \mathbb{R}, t \geq 0. \tag{2.11}$$

*Proof.* In this proof, whenever there is no ambiguity, we write $S(x,t;p)$ instead of $S(x,t;p,a,b)$.

We begin by proving the first, leftmost inequality in (2.11). Let $p \geq 0$ be an arbitrary non-negative number. We first prove the statement for this $p$. For this, it suffices to prove that

$$\inf_{\substack{(x,t)\in\mathbb{R}\times[0,+\infty)\\ s.t.\ (x,t,p)\in\Omega_i}} \{S(x,t;p) - px\} > -\infty, \qquad \forall i \in \{1,2,3,4,5\}.$$

For $(x,t,p) \in \Omega_1$, we have $x \geq pt + \frac{at^2}{2}$ and

$$\begin{aligned} S(x,t;p) - px &= -\frac{a^2t^3}{6} - \frac{apt^2}{2} + atx - \frac{p^2t}{2} \geq -\frac{a^2t^3}{6} - \frac{apt^2}{2} + at\left(pt + \frac{at^2}{2}\right) - \frac{p^2t}{2} \\ &= \frac{a^2t^3}{3} + \frac{apt^2}{2} - \frac{p^2t}{2}. \end{aligned}$$

Note that the function $t \mapsto \frac{a^2t^3}{3} + \frac{apt^2}{2} - \frac{p^2t}{2}$ is a third-order polynomial with positive leading coefficient $\frac{a^2}{3}$. Thus, this function is continuous and coercive for $t \in [0,+\infty)$, and hence, it has a finite lower bound. In other words, we have

$$\inf_{\substack{(x,t)\in\mathbb{R}\times[0,+\infty)\\ s.t.\ (x,t,p)\in\Omega_1}} \{S(x,t;p) - px\} \geq \inf_{t\geq 0}\left\{\frac{a^2t^3}{3} + \frac{apt^2}{2} - \frac{p^2t}{2}\right\} > -\infty.$$

For $(x,t,p) \in \Omega_2$ and $t < \frac{p}{b}$, we have $x < 0$, and hence,

$$\begin{aligned} S(x,t;p) - px &= -\frac{b^2t^3}{6} + \frac{bpt^2}{2} - \frac{p^2t}{2} - btx \geq -\frac{b^2t^3}{6} + \frac{bpt^2}{2} - \frac{p^2t}{2} \\ &\geq -\frac{p^3}{6b} + 0 - \frac{p^3}{2b} = -\frac{2p^3}{3b}. \end{aligned}$$

For $(x,t,p) \in \Omega_2$ and $t \geq \frac{p}{b}$, we have $x < -\frac{b}{2}(t-\frac{p}{b})^2$, and hence,

$$\begin{aligned} S(x,t;p) - px &= -\frac{b^2t^3}{6} + \frac{bpt^2}{2} - \frac{p^2t}{2} - btx \geq -\frac{b^2t^3}{6} + \frac{bpt^2}{2} - \frac{p^2t}{2} + \frac{b^2t}{2}\left(t-\frac{p}{b}\right)^2 \\ &= \frac{b^2t^3}{3} - \frac{bpt^2}{2}. \end{aligned}$$

The function $t \mapsto \frac{b^2t^3}{3} - \frac{bpt^2}{2}$ is a third-order polynomial with positive leading coefficient, which implies that this function is bounded from below for $t \in [0,+\infty)$. Therefore, we obtain

$$\inf_{\substack{(x,t)\in\mathbb{R}\times[0,+\infty) \\ s.t.\ (x,t,p)\in\Omega_2}} \{S(x,t;p) - px\} \geq \min\left\{-\frac{2p^3}{3b}, \inf_{t\geq 0}\left\{\frac{b^2t^3}{3} - \frac{bpt^2}{2}\right\}\right\} > -\infty.$$

Consider $(x,t,p) \in \Omega_3$ and let $\Delta := (bt-p)^2 + 2x(a+2b)$. If $0 \leq t < \frac{p}{b}$, we have $0 \leq x < pt + \frac{at^2}{2}$. By Lemma 2.3.2, $S(x,t;p) - px$ is continuous with respect to $(x,t)$, and hence, it is bounded from below in the compact domain $\{(x,t) \in \mathbb{R}^2 \colon 0 \leq t \leq \frac{p}{b},\ 0 \leq x \leq pt + \frac{at^2}{2}\}$. If $t \geq \frac{p}{b}$, we have $\frac{a}{2}(t-\frac{p}{b})^2 \leq x < pt + \frac{at^2}{2}$, and hence, $\Delta \geq (bt-p)^2 + a(a+2b)(t-\frac{p}{b})^2 = (1+\frac{a}{b})^2(bt-p)^2$. Therefore, we obtain

$$\begin{aligned} & S(x,t;p) - px \\ =& \frac{a+b}{3(a+2b)^2}\left((bt-p)^3 + \Delta^{3/2}\right) - \frac{b}{a+2b}(bt-p)x - \frac{b^2t^3}{6} + \frac{bpt^2}{2} - \frac{p^2t}{2} - px \\ \geq& \frac{(a+b)(bt-p)^3}{3(a+2b)^2}\left(1 + \left(1+\frac{a}{b}\right)^3\right) - \frac{(b^2t+(a+b)p)x}{a+2b} - \frac{b^2t^3}{6} + \frac{bpt^2}{2} - \frac{p^2t}{2} \\ \geq& \frac{(a+b)(bt-p)^3}{3(a+2b)^2}\left(1 + \left(1+\frac{a}{b}\right)^3\right) - \frac{b^2t+(a+b)p}{a+2b}\left(pt + \frac{at^2}{2}\right) \\ & \qquad - \frac{b^2t^3}{6} + \frac{bpt^2}{2} - \frac{p^2t}{2} \\ =:& g(t), \end{aligned} \tag{2.12}$$

where the function $g$ defined in the last line is a third-order polynomial with respect

to $t$ whose leading coefficient reads:

$$\begin{aligned}
&\frac{(a+b)b^3}{3(a+2b)^2}\left(1+\left(1+\frac{a}{b}\right)^3\right)-\frac{b^2a}{2(a+2b)}-\frac{b^2}{6}\\
=&\frac{(a+b)b^3+(a+b)^4}{3(a+2b)^2}-\frac{3ab^2+b^2(a+2b)}{6(a+2b)}\\
=&\frac{(a+b)(a^2+ab+b^2)-b^2(2a+b)}{3(a+2b)}=\frac{a^2}{3}>0.
\end{aligned}$$

Therefore, the function $g$ is bounded from below for $t\in[0,+\infty)$, which implies

$$\begin{aligned}
\inf_{\substack{(x,t)\in\mathbb{R}\times[0,+\infty)\\ s.t.\ (x,t,p)\in\Omega_3}}\{S(x,t;p)-px\}\geq&\min\left\{\inf_{0\leq t\leq\frac{p}{b},0\leq x\leq pt+\frac{at^2}{2}}\{S(x,t;p)-px\},\inf_{t\geq0}g(t)\right\}\\
>&-\infty.
\end{aligned}$$

If $(x,t,p)\in\Omega_4$, we have $x\geq0$. By (2.10), we obtain

$$\frac{\partial}{\partial x}(S(x,t;p)-px)=\sqrt{2ax}-p\in\begin{cases}[0,+\infty) & x\geq\dfrac{p^2}{2a},\\ (-\infty,0] & 0\leq x<\dfrac{p^2}{2a}.\end{cases}$$

Hence, the minimal value of $S(x,t;p)-px$ is attained at $x=\frac{p^2}{2a}$, and we have

$$\inf_{\substack{(x,t)\in\mathbb{R}\times[0,+\infty)\\ s.t.\ (x,t,p)\in\Omega_4}}\{S(x,t;p)-px\}\geq S\left(\frac{p^2}{2a},t;p\right)-\frac{p^3}{2a}=\frac{p^3}{3a}-\frac{p^3}{6b}-\frac{p^3}{2a}>-\infty.$$

If $(x,t,p)\in\Omega_5$, we have $x\leq0$. By (2.10), we obtain

$$\frac{\partial}{\partial x}(S(x,t;p)-px)=-\sqrt{-2bx}-p\leq0.$$

Then, the minimal value is attained at $x=0$, and we have

$$\inf_{\substack{(x,t)\in\mathbb{R}\times[0,+\infty)\\ s.t.\ (x,t,p)\in\Omega_5}}\{S(x,t;p)-px\}\geq S\left(0,t;p\right)=-\frac{p^3}{6b}>-\infty.$$

Therefore, we have $\inf_{x\in\mathbb{R},t\geq 0}\{S(x,t;p)-px\} > -\infty$ for all $p \geq 0$, and hence, the first inequality in (2.11) holds for all $p \geq 0$. If $p < 0$, we get

$$\begin{aligned}\inf_{x\in\mathbb{R},t\geq 0}\{S(x,t;p,a,b)-px\} &= \inf_{x\in\mathbb{R},t\geq 0}\{S(-x,t;-p,b,a)-px\}\\ &= \inf_{y\in\mathbb{R},t\geq 0}\{S(y,t;q,b,a)-qy\} > -\infty,\end{aligned}$$

where the first equality holds by (2.15), the second equality holds by the change of variables $q = -p$ and $y = -x$, and the last inequality holds since we have already proved the first inequality in (2.11) for the positive $q$ case. Therefore, the first inequality in (2.11) also holds for all $p < 0$.

Now, we prove the second inequality in (2.11). We first consider the case where $p \geq 0$. Since the function $x \mapsto px + t(a+b)|x|$ is linear in $[0,+\infty)$ and $(-\infty,0]$ and the function $x \mapsto S(x,t;p)$ is convex by Lemma 2.3.2(b), then to prove that $S(x,t;p) \leq px + t(a+b)|x|$ holds for all $x \in \mathbb{R}$, it suffices to prove it for $x = 0$, $x \geq C$, and $x \leq -C$ for some large scalar $C > 0$. In other words, it suffices to consider the cases where $x = 0$, $(x,t,p) \in \Omega_1$, and $(x,t,p) \in \Omega_2$. Note that for all $p \geq 0$ and $t \leq \frac{p}{b}$, the value $S(x,t;p)$ at $x = 0$ equals the function $f_2(0,t;p,a,b)$, according to the continuity of $S$. Therefore, we only need to consider the following three cases:

1. If $x = 0$ and $t > \frac{p}{b}$, we have

$$S(x,t;p) = S(0,t;p) = f_4(0,t;p,a,b) = -\frac{p^3}{6b} \leq 0 = px + t(a+b)|x|.$$

2. If $(x,t,p) \in \Omega_1$, we have $x \geq 0$ and

$$S(x,t;p) = -\frac{a^2t^3}{6} - \frac{apt^2}{2} + atx - \frac{p^2t}{2} + px \leq atx + px \leq px + t(a+b)|x|.$$

3. If $(x,t,p) \in \Omega_2$, we have $x \le 0$ and

$$\begin{aligned} S(x,t;p) &= -\frac{b^2t^3}{6} + \frac{bpt^2}{2} - \frac{p^2t}{2} - btx + px \\ &= px + bt|x| - \frac{t}{6}\left(bt - \frac{3p}{2}\right)^2 - \frac{p^2t}{8} \\ &\le px + bt|x| \le px + t(a+b)|x|. \end{aligned}$$

Therefore, the second inequality in (2.11) holds for all $t, p \ge 0$, and $x \in \mathbb{R}$. If $p < 0$, letting $q = -p$ and $y = -x$, we have

$$S(x,t;p,a,b) = S(y,t;q,b,a) \le qy + t(a+b)|y| = px + t(a+b)|x|,$$

where the first equality holds by (2.15) and the first inequality holds since we have already proved it above for the positive $q$ case. Therefore, the second inequality in (2.11) also holds for all $p < 0$. □

*Lemma* 2.3.4. Let $a, b > 0$ be positive scalars and $S$ be the function defined in (2.6) and (2.15). Then, there holds

$$\left|\frac{\partial S(x,t;p,a,b)}{\partial x}\right| \le C(R_x, R_t, R_p) < +\infty, \tag{2.13}$$

for all $x \in [-R_x, R_x]$, $t \in [0, R_t]$, and $p \in [-R_p, R_p]$.

*Proof.* We first consider the case where $p \ge 0$. Let $|x| \le R_x$, $t \in [0, R_t]$, and $p \in [0, R_p]$. We obtain the derivatives of $S$ with respect to $x$ in (2.9) and (2.10). If $(x,t,p) \in \Omega_1$, we have

$$\left|\frac{\partial S(x,t;p,a,b)}{\partial x}\right| = |at + p| \le aR_t + R_p =: C_1.$$

If $(x,t,p) \in \Omega_2$, we have

$$\left|\frac{\partial S(x,t;p,a,b)}{\partial x}\right| = |-bt+p| \le bR_t + R_p =: C_2.$$

If $(x,t,p) \in \Omega_3$, we have

$$\begin{aligned}\left|\frac{\partial S(x,t;p,a,b)}{\partial x}\right| &= \left|\frac{(a+b)\sqrt{(bt-p)^2+2x(a+2b)}}{a+2b} - \frac{b}{a+2b}(bt-p)\right| \\ &\le \frac{(a+b)\sqrt{2b^2R_t^2+2R_p^2+2(a+2b)R_x}+b^2R_t+bR_p}{a+2b} =: C_3.\end{aligned}$$

If $(x,t,p) \in \Omega_4$, we have

$$\left|\frac{\partial S(x,t;p,a,b)}{\partial x}\right| = \sqrt{2ax} \le \sqrt{2aR_x} =: C_4.$$

If $(x,t,p) \in \Omega_5$, we have

$$\left|\frac{\partial S(x,t;p,a,b)}{\partial x}\right| = \left|-\sqrt{-2bx}\right| \le \sqrt{2bR_x} =: C_5.$$

Therefore, the bound in (2.13) holds at $(x,t,p)$ for the constant $C(R_x,R_t,R_p) = C_{a,b}$ defined by

$$C_{a,b} := \max\{C_1,C_2,C_3,C_4,C_5\}.$$

Now, we consider the case where $p<0$. Let $|x| \le R_x$, $t \in [0,R_t]$, and $p \in [-R_p,0)$. Let $q=-p$ and $y=-x$. Hence, we have $q \in (0,R_p]$ and $y \in [-R_x,R_x]$. By (2.15), we have

$$\left|\frac{\partial S(x,t;p,a,b)}{\partial x}\right| = \left|-\frac{\partial S(y,t;q,b,a)}{\partial y}\right| \le C_{b,a},$$

where the inequality was proved in the beginning of this proof since the parameter $q$ is positive. Therefore, the inequality (2.13) holds for all $x \in [-R_x,R_x]$, $t \in [0,R_t]$, and $p \in [-R_p,R_p]$ with the constant $C(R_x,R_t,R_p) := \max\{C_{a,b},C_{b,a}\}$. □

*Lemma* 2.3.5. Let $a, b > 0$, $x, p \in \mathbb{R}$, and $t > 0$. Let $S$ be the function defined in (2.6) and (2.15) and $K$ be the function defined in (2.4) with parameters $a, b$. Let $[0, t] \ni s \mapsto \gamma(s; x, t, p, a, b) \in \mathbb{R}$ be the trajectory defined in (2.9), (2.10), (2.11), (2.13), (2.14), and (2.16) for different cases. Then, we have

$$\begin{aligned}\int_0^t \left(\frac{1}{2}\left(\frac{d}{ds}\gamma(s; x, t, p, a, b)\right)^2 - K(\gamma(s; x, t, p, a, b))\right) ds + p\gamma(0; x, t, p, a, b) \\ = S(x, t; p, a, b).\end{aligned} \tag{2.14}$$

*Proof.* If $(x, t, p) \in \Omega_1$, the left-hand side of (2.14) equals

$$\begin{aligned}
&\int_0^t \left(\frac{1}{2}(p + as)^2 + a\left(x - p(t - s) - \frac{a}{2}(t^2 - s^2)\right)\right) ds + p\left(x - pt - \frac{at^2}{2}\right) \\
&= \int_0^t \left(a^2 s^2 + 2aps + \frac{p^2}{2} + ax - apt - \frac{a^2 t^2}{2}\right) ds + px - p^2 t - \frac{apt^2}{2} \\
&= \frac{a^2 t^3}{3} + apt^2 + \frac{p^2 t}{2} + atx - apt^2 - \frac{a^2 t^3}{2} + px - p^2 t - \frac{apt^2}{2} \\
&= -\frac{a^2 t^3}{6} - \frac{apt^2}{2} + atx - \frac{p^2 t}{2} + px \\
&= S(x, t; p, a, b).
\end{aligned}$$

If $(x, t, p) \in \Omega_2$, the left-hand side of (2.14) equals

$$\begin{aligned}
&\int_0^t \left(\frac{1}{2}(p - bs)^2 - b\left(x - p(t - s) + \frac{b}{2}(t^2 - s^2)\right)\right) ds + p\left(x - pt + \frac{bt^2}{2}\right) \\
&= \int_0^t \left(b^2 s^2 - 2bps + \frac{p^2}{2} - bx + bpt - \frac{b^2 t^2}{2}\right) ds + px - p^2 t + \frac{bpt^2}{2} \\
&= \frac{b^2 t^3}{3} - bpt^2 + \frac{p^2 t}{2} - bxt + bpt^2 - \frac{b^2 t^3}{2} + px - p^2 t + \frac{bpt^2}{2} \\
&= -\frac{b^2 t^3}{6} + \frac{bpt^2}{2} - \frac{p^2 t}{2} + px - btx \\
&= S(x, t; p, a, b).
\end{aligned}$$

If $(x,t,p) \in \Omega_3$, let $\Delta \in \mathbb{R}$ be defined by $\Delta := (bt-p)^2 + 2(2b+a)x$ and $\tau$ be the scalar defined in (2.12). Then, the left-hand side of (2.14) equals

$$\begin{aligned}
&\int_0^\tau \left(\frac{1}{2}(p-bs)^2 - b\left(-p(\tau-s)+\frac{b}{2}(\tau^2-s^2)\right)\right)ds + p\left(-p\tau+\frac{b\tau^2}{2}\right)\\
&\qquad + \int_\tau^t \left(\frac{1}{2}(p-b\tau+a(s-\tau))^2 + a\left((p-b\tau)(s-\tau)+\frac{a}{2}(s-\tau)^2\right)\right)ds\\
&= \int_0^\tau \left(b^2s^2 - 2bps + \frac{p^2}{2} + bp\tau - \frac{b^2\tau^2}{2}\right)ds - p^2\tau + \frac{bp\tau^2}{2}\\
&\qquad + \int_0^{t-\tau} \left(\frac{1}{2}(p-b\tau+as)^2 + a\left((p-b\tau)s+\frac{a}{2}s^2\right)\right)ds\\
&= \frac{b^2\tau^3}{3} - bp\tau^2 + \frac{p^2\tau}{2} + bp\tau^2 - \frac{b^2\tau^3}{2} - p^2\tau + \frac{bp\tau^2}{2}\\
&\qquad + \int_0^{t-\tau}\left(a^2s^2 + 2a(p-b\tau)s + \frac{(p-b\tau)^2}{2}\right)ds\\
&= -\frac{b^2\tau^3}{6} + \frac{bp\tau^2}{2} - \frac{p^2\tau}{2} + \frac{a^2(t-\tau)^3}{3} + a(p-b\tau)(t-\tau)^2 + \frac{(p-b\tau)^2(t-\tau)}{2}.
\end{aligned}$$

Let $r := t-\tau = \frac{bt-p+\sqrt{\Delta}}{2b+a}$ and $A := bt-p$. After some calculations, the left-hand side of (2.14) equals

$$\begin{aligned}
&\frac{(a+b)(a+2b)r^3}{3} + (p-bt)\frac{(2a+3b)r^2}{2} + (bt-p)^2 r - \frac{b^2t^3}{6} + \frac{bpt^2}{2} - \frac{p^2t}{2}\\
&= \frac{a+b}{3(a+2b)^2}\left(4A^3 + 3A^2\sqrt{\Delta} + 6(a+2b)xA + \sqrt{\Delta}^3\right)\\
&\qquad - \frac{2a+3b}{(a+2b)^2}\left(A^3 + A^2\sqrt{\Delta} + (a+2b)xA\right)\\
&\qquad + \frac{A^3 + A^2\sqrt{\Delta}}{a+2b} - \frac{b^2t^3}{6} + \frac{bpt^2}{2} - \frac{p^2t}{2}\\
&= \frac{a+b}{3(a+2b)^2}\left(A^3 + \sqrt{\Delta}^3\right) - \frac{bxA}{a+2b} - \frac{b^2t^3}{6} + \frac{bpt^2}{2} - \frac{p^2t}{2}\\
&= S(x,t;p,a,b).
\end{aligned}$$

If $(x,t,p) \in \Omega_4$, the left-hand side of (2.14) equals

$$\begin{aligned}
&\int_{t-\sqrt{\frac{2x}{a}}}^{t} \left( \frac{1}{2} \left( a \left( s - t - \sqrt{\frac{2x}{a}} \right) \right)^2 + a \left( \frac{a}{2} \left( s - t + \sqrt{\frac{2x}{a}} \right)^2 \right) \right) ds \\
&\qquad + \int_0^{\frac{p}{b}} \left( \frac{1}{2} (p - bs)^2 + b \left( \frac{1}{2b} (p - bs)^2 \right) \right) ds + p \left( -\frac{p^2}{2b} \right) \\
&= \int_{t-\sqrt{\frac{2x}{a}}}^{t} a^2 \left( s - t + \sqrt{\frac{2x}{a}} \right)^2 ds + \int_0^{\frac{p}{b}} (p - bs)^2 \, ds - \frac{p^3}{2b} \\
&= \int_0^{\sqrt{\frac{2x}{a}}} a^2 s^2 ds + \int_0^{\frac{p}{b}} b^2 s^2 ds - \frac{p^3}{2b} \\
&= \frac{a^2}{3} \left( \frac{2x}{a} \right)^{3/2} + \frac{b^2}{3} \left( \frac{p}{b} \right)^3 - \frac{p^3}{2b} \\
&= S(x,t;p,a,b).
\end{aligned}$$

If $(x,t,p) \in \Omega_5$, the left-hand side of (2.14) equals

$$\begin{aligned}
&\int_{t-\sqrt{\frac{2|x|}{b}}}^{t} \left( \frac{1}{2} \left( b \left( s - t - \sqrt{\frac{2|x|}{b}} \right) \right)^2 + b \left( \frac{b}{2} \left( s - t + \sqrt{\frac{2|x|}{b}} \right)^2 \right) \right) ds \\
&\qquad + \int_0^{\frac{p}{b}} \left( \frac{1}{2} (p - bs)^2 + b \left( \frac{1}{2b} (p - bs)^2 \right) \right) ds + p \left( -\frac{p^2}{2b} \right) \\
&= \int_{t-\sqrt{\frac{2|x|}{b}}}^{t} b^2 \left( s - t + \sqrt{\frac{2|x|}{b}} \right)^2 ds + \int_0^{\frac{p}{b}} (p - bs)^2 \, ds - \frac{p^3}{2b} \\
&= \int_0^{\sqrt{\frac{2|x|}{b}}} b^2 s^2 ds + \int_0^{\frac{p}{b}} b^2 s^2 ds - \frac{p^3}{2b} \\
&= \frac{b^2}{3} \left( \frac{2|x|}{b} \right)^{3/2} + \frac{b^2}{3} \left( \frac{p}{b} \right)^3 - \frac{p^3}{2b} \\
&= S(x,t;p,a,b).
\end{aligned}$$

Therefore, (2.14) holds for any $x \in \mathbb{R}$, $p, t \geq 0$.

Now, we consider the case where $p < 0$. By definition, we have $\gamma(s;x,t,p,a,b) = -\gamma(s;-x,t,-p,b,a)$ and $S(x,t;p,a,b) = S(-x,t;-p,b,a)$. Let $K_{b,a}$ denote the fol-

lowing function:

$$K_{b,a}(y) := K(-y) = \begin{cases} -by & y \geq 0, \\ ay & y < 0. \end{cases} \tag{2.15}$$

Then, the left-hand side of (2.14) equals

$$\begin{aligned} &\int_0^t \left( \frac{1}{2} \left( -\frac{d}{ds}\gamma(s; -x, t, -p, b, a) \right)^2 - K(-\gamma(s; -x, t, -p, b, a)) \right) ds \\ &\qquad\qquad - p\gamma(0; -x, t, -p, b, a) \\ &= \int_0^t \left( \frac{1}{2} \left( \frac{d}{ds}\gamma(s; y, t, q, b, a) \right)^2 - K_{b,a}(\gamma(s; y, t, q, b, a)) \right) ds + q\gamma(0; y, t, q, b, a) \\ &= S(y, t; q, b, a) = S(-x, t; -p, b, a) = S(x, t; p, a, b), \end{aligned}$$

where the first equality holds by (2.15) and the change of variables $y = -x$ and $q = -p$ and the second equality holds since we have already proved above that (2.14) holds in the positive $q$ case. Therefore, (2.14) holds for all $x, p \in \mathbb{R}$ and $t \geq 0$. □

*Lemma* 2.3.6. Let $a, b > 0$, $x, p \in \mathbb{R}$, and $t > 0$. Let $S$ be the function defined in (2.6) and (2.15) and $K$ be the function defined in (2.4) with parameters $a, b$. Let $[0, t] \ni s \mapsto \gamma(s; x, t, p, a, b) \in \mathbb{R}$ be the trajectory defined in (2.9), (2.10), (2.11), (2.13), (2.14), and (2.16) for different cases. Then, we have

$$\frac{\partial S}{\partial p}(x, t; p, a, b) = \gamma(0; x, t, p, a, b). \tag{2.16}$$

*Proof.* We prove (2.16) using (2.1) and straightforward calculation. If $(x, t, p) \in \Omega_1$, we have

$$\frac{\partial S}{\partial p}(x, t; p, a, b) = -\frac{at^2}{2} - pt + x = \gamma(0; x, t, p, a, b).$$

If $(x,t,p) \in \Omega_2$, we have

$$\frac{\partial S}{\partial p}(x,t;p,a,b) = \frac{bt^2}{2} - pt + x = \gamma(0;x,t,p,a,b).$$

If $(x,t,p) \in \Omega_3$, let $\Delta := (bt-p)^2 + 2x(2b+a)$. Then, we have

$$\begin{aligned}\frac{\partial S}{\partial p}(x,t;p,a,b) &= \frac{a+b}{(a+2b)^2}\left(-(bt-p)^2 + (p-bt)\sqrt{\Delta}\right) + \frac{bx}{a+2b} + \frac{bt^2}{2} - pt \\ &= -\frac{(a+b)(bt-p)^2}{(a+2b)^2} + \frac{bx}{a+2b} + \frac{bt^2}{2} - pt + \frac{(a+b)(p-bt)\sqrt{\Delta}}{(a+2b)^2} \\ &= -\frac{(a+b)p^2}{(a+2b)^2} - \frac{(a^2+2ab+2b^2)pt}{(a+2b)^2} + \frac{(a^2+2ab+2b^2)bt^2}{2(a+2b)^2} \\ &\qquad + \frac{bx}{a+2b} + \frac{(a+b)(p-bt)\sqrt{\Delta}}{(a+2b)^2}.\end{aligned} \tag{2.17}$$

Let $c \in \mathbb{R}$ be defined by $c := (a+b)t + p$ and $\tau$ be the scalar defined in (2.12), which equals $\frac{c-\sqrt{\Delta}}{2b+a}$. Then, by definition, we have

$$\begin{aligned}\gamma(0;x,t,p,a,b) &= -p\tau + \frac{b\tau^2}{2} = \frac{p\sqrt{\Delta} - pc}{2b+a} + \frac{bc^2 + b\Delta - 2bc\sqrt{\Delta}}{2(2b+a)^2} \\ &= \frac{bc^2 + b\Delta - 2pc(2b+a)}{2(2b+a)^2} + \frac{2(2b+a)p - 2bc}{2(2b+a)^2}\sqrt{\Delta}.\end{aligned} \tag{2.18}$$

By some calculations, we have

$$\begin{aligned}&bc^2 + b\Delta - 2pc(2b+a) \\ =& b\left((a+b)^2t^2 + p^2 + 2(a+b)tp + (bt-p)^2 + 2x(2b+a)\right) \\ &\qquad - 2(2b+a)(a+b)tp - 2(2b+a)p^2 \\ =& b(bt-p)^2 - (3b+2a)p^2 - 2(a+b)^2tp + b(a+b)^2t^2 + 2bx(2b+a) \\ =& -2(a+b)p^2 - 2((a+b)^2+b^2)tp + ((a+b)^2+b^2)bt^2 + 2(2b+a)bx,\end{aligned} \tag{2.19}$$

and

$$2(2b+a)p - 2bc = 2(2b+a)p - 2b(a+b)t - 2bp = 2(a+b)p - 2(a+b)bt. \quad (2.20)$$

Combining (2.17), (2.18), (2.19), and (2.20), we obtain (2.16) for $(x,t,p) \in \Omega_3$.

If $(x,t,p) \in \Omega_4 \cup \Omega_5$, we have

$$\frac{\partial S}{\partial p}(x,t;p,a,b) = -\frac{p^2}{2b} = \gamma(0;x,t,p,a,b).$$

Thus, we have proved (2.16) for any $x \in \mathbb{R}$ and $t, p \geq 0$.

If $p < 0$, let $q = -p$. Then, we have

$$\frac{\partial S}{\partial p}(x,t;p,a,b) = -\frac{\partial S}{\partial q}(-x,t;q,b,a) = -\gamma(0;-x,t,q,b,a) = \gamma(0;x,t,p,a,b),$$

where the first equality holds by (2.15), the second equality holds since we have already proved (2.16) in the positive $q$ case, and the third equality holds by (2.16). Hence, (2.16) also holds for any $x \in \mathbb{R}$, $t \geq 0$, and $p < 0$. □

*Lemma* 2.3.7. Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function. Let $\{a_i, b_i\}_{i=1}^n$ be positive constants and $K_i\colon \mathbb{R} \to \mathbb{R}$ be the function defined by (2.4) with constants $a = a_i$ and $b = b_i$ for each $i \in \{1, \ldots, n\}$. Define the function $\ell\colon \mathbb{R}^n \times [0, +\infty) \times \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ by

$$\ell(\boldsymbol{x}, t, \boldsymbol{p}) := -\sum_{i=1}^{n} S(x_i, t; p_i, a_i, b_i) + J^*(\boldsymbol{p}), \quad (2.21)$$

for all $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$, $\boldsymbol{p} = (p_1, \ldots, p_n) \in \mathbb{R}^n$, and $t \geq 0$, where each function $S(x_i, t; p_i, a_i, b_i)$ on the right-hand side is the function defined in (2.6) and (2.15). Let $M > 0$ and $\alpha \in \mathbb{R}$ be arbitrary scalars. Then, there exists $M_{\boldsymbol{p}} > 0$, such that

$\ell(\boldsymbol{x}, t, \boldsymbol{p}) \geq \alpha$ holds for all $t \in [0, M]$ and for all $\boldsymbol{x}, \boldsymbol{p} \in \mathbb{R}^n$ satisfying $\|\boldsymbol{x}\| \leq M$ and $\|\boldsymbol{p}\| \geq M_{\boldsymbol{p}}$.

*Proof.* Since $J$ is a finite-valued convex function, its Legendre-Fenchel transform $J^*$ is 1-coercive, and hence, the function $\boldsymbol{p} \mapsto J^*(\boldsymbol{p}) - M\|\boldsymbol{p}\|$ is also 1-coercive. As a result, there exists $M_{\boldsymbol{p}} > 0$, such that

$$J^*(\boldsymbol{p}) - M\|\boldsymbol{p}\| \geq M^2 \sum_{i=1}^{n} (a_i + b_i) + \alpha, \tag{2.22}$$

for all $\boldsymbol{p} \in \mathbb{R}^n$ satisfying $\|\boldsymbol{p}\| \geq M_{\boldsymbol{p}}$. By straightforward calculation, for all $t \in [0, M]$ and for all $\boldsymbol{p}, \boldsymbol{x} \in \mathbb{R}^n$ satisfying $\|\boldsymbol{p}\| \geq M_{\boldsymbol{p}}$ and $\|\boldsymbol{x}\| \leq M$, we have

$$\begin{aligned}
\ell(\boldsymbol{x}, t, \boldsymbol{p}) &= -\sum_{i=1}^{n} S(x_i, t; p_i, a_i, b_i) + J^*(\boldsymbol{p}) \geq -\sum_{i=1}^{n} (p_i x_i + t(a_i + b_i)|x_i|) + J^*(\boldsymbol{p}) \\
&\geq -\|\boldsymbol{p}\|\|\boldsymbol{x}\| - t\|\boldsymbol{x}\| \sum_{i=1}^{n} (a_i + b_i) + J^*(\boldsymbol{p}) \\
&\geq -M\|\boldsymbol{p}\| - M^2 \sum_{i=1}^{n} (a_i + b_i) + J^*(\boldsymbol{p}) \\
&\geq \alpha,
\end{aligned}$$

where the first inequality holds by Lemma 2.3.3 and the last inequality follows from (2.22). □

*Lemma* 2.3.8. Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function. Let $\{a_i, b_i\}_{i=1}^n$ be positive constants and $K_i\colon \mathbb{R} \to \mathbb{R}$ be the function defined by (2.4) with constants $a = a_i$ and $b = b_i$ for each $i \in \{1, \ldots, n\}$. Let $S\colon \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$ be the function defined in (2.22). Then, the function $S$ is continuous in $\mathbb{R}^n \times [0, +\infty)$.

*Proof.* First, we show the existence of the maximizer in (2.22). Let $\ell\colon \mathbb{R}^n \times [0, +\infty) \times \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be the function defined in (2.21). The function $\boldsymbol{p} \mapsto$

$-\ell(\boldsymbol{x}, t, \boldsymbol{p})$ is the objective function in the maximization problem (2.22) at $(\boldsymbol{x}, t)$. Since $J$ is finite-valued and convex, its Legendre-Fenchel transform $J^*$ is convex, lower semi-continuous, and 1-coercive. By Lemma 2.3.1, if $t > 0$, the function $p_i \mapsto -S(x_i, t; p_i, a_i, b_i)$ is convex for each $i \in \{1, \ldots, n\}$. If $t = 0$, by straightforward calculation, we have $-S(x_i, t; p_i, a_i, b_i) = -p_i x_i$, which is a convex function with respect to $p_i$. Therefore, for all $\boldsymbol{x} \in \mathbb{R}^n$ and $t \geq 0$, the function $\ell$ is convex, lower semi-continuous, and 1-coercive with respect to $\boldsymbol{p}$. As a result, the maximizer in (2.22) exists for all $\boldsymbol{x} \in \mathbb{R}^n$ and $t \geq 0$ (see [63, Definition IV.3.2.6]), and hence, the function $S$ is finite-valued in $\mathbb{R}^n \times [0, +\infty)$.

Let $\boldsymbol{x} \in \mathbb{R}^n$ and $t \geq 0$. Now, we show the continuity of the function $S$ at $(\boldsymbol{x}, t)$ by showing it is lower semi-continuous and upper semi-continuous. We begin by proving lower semi-continuity. Let $\boldsymbol{p}^*$ be a maximizer in (2.22) at $(\boldsymbol{x}, t)$. By Lemma 2.3.2, each function $(x_i, t) \mapsto S(x_i, t; p_i, a_i, b_i)$ is continuous, and hence, the function $(\boldsymbol{x}, t) \mapsto \ell(\boldsymbol{x}, t, \boldsymbol{p})$ is continuous for all $\boldsymbol{p}$ in the domain of $J^*$. For any sequence $\{(\boldsymbol{x}^k, t^k)\} \subseteq \mathbb{R}^n \times [0, +\infty)$ converging to $(\boldsymbol{x}, t)$, we have

$$\liminf_{k\to\infty} S(\boldsymbol{x}^k, t^k) \geq \liminf_{k\to\infty} -\ell(\boldsymbol{x}^k, t^k, \boldsymbol{p}^*) = -\ell(\boldsymbol{x}, t, \boldsymbol{p}^*) = S(\boldsymbol{x}, t),$$

where the inequality holds by definition of $S$ in (2.22), the first equality holds since the function $(\boldsymbol{x}, t) \mapsto \ell(\boldsymbol{x}, t, \boldsymbol{p}^*)$ is continuous, and the last equality holds since $\boldsymbol{p}^*$ is a maximizer in (2.22) at $(\boldsymbol{x}, t)$. Therefore, the function $S$ is lower semi-continuous at $(\boldsymbol{x}, t)$.

Now, we prove that $S$ is upper semi-continuous at $(\boldsymbol{x}, t)$. Let $\delta > 0$ be an arbitrary positive number. Our goal is to find a neighborhood $\mathcal{N}_{\boldsymbol{x},t}$ of $(\boldsymbol{x}, t)$ in $\mathbb{R}^n \times [0, +\infty)$,

such that there holds

$$S(\boldsymbol{y}, s) \leq S(\boldsymbol{x}, t) + \delta \quad \forall (\boldsymbol{y}, s) \in \mathcal{N}_{\boldsymbol{x},t}. \tag{2.23}$$

Note that by definition of $S$ in (2.22), the inequality in (2.23) holds if and only if there holds

$$-\ell(\boldsymbol{y}, s, \boldsymbol{p}) \leq S(\boldsymbol{x}, t) + \delta \quad \forall \boldsymbol{p} \in \mathbb{R}^n. \tag{2.24}$$

Therefore, it suffices to find a neighborhood $\mathcal{N}_{\boldsymbol{x},t}$ of $(\boldsymbol{x}, t)$ in $\mathbb{R}^n \times [0, +\infty)$, such that (2.24) holds for all $(\boldsymbol{y}, s) \in \mathcal{N}_{\boldsymbol{x},t}$ and $\boldsymbol{p} \in \mathbb{R}^n$. Let $M > 0$ be a scalar. By Lemma 2.3.7 with $\alpha = -S(\boldsymbol{x}, t) - \delta$, there exists $M_{\boldsymbol{p}} > 0$, such that for all $\boldsymbol{y}, \boldsymbol{p} \in \mathbb{R}^n$ and $s \geq 0$ satisfying $\|\boldsymbol{y}\| \leq M$, $s \leq M$, and $\|\boldsymbol{p}\| \geq M_{\boldsymbol{p}}$, we have

$$\ell(\boldsymbol{y}, s, \boldsymbol{p}) \geq \alpha = -S(\boldsymbol{x}, t) - \delta. \tag{2.25}$$

Now, we consider the case when $\|\boldsymbol{p}\| < M_{\boldsymbol{p}}$ holds. By Lemmas 2.3.1 and 2.3.2, the function $(y, s, p) \mapsto S(y, s; p, a_i, b_i)$ is continuous in $\mathbb{R} \times [0, +\infty) \times \mathbb{R}$ for each $i \in \{1, \ldots, n\}$. Hence, there exists a neighborhood $\tilde{\mathcal{N}}_{\boldsymbol{x},t} \subset \mathbb{R}^n \times [0, +\infty)$ of $(\boldsymbol{x}, t)$, such that $|S(y_i, s; p_i, a_i, b_i) - S(x_i, t; p_i, a_i, b_i)| \leq \frac{\delta}{n}$ holds for all $(\boldsymbol{y}, s) \in \tilde{\mathcal{N}}_{\boldsymbol{x},t}$, $\|\boldsymbol{p}\| \leq M_{\boldsymbol{p}}$ and $i \in \{1, \ldots, n\}$. Recall that we use $x_i$, $y_i$, and $p_i$ to denote the $i$-th component of the vectors $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{p}$, respectively. Therefore, for all $(\boldsymbol{y}, s) \in \tilde{\mathcal{N}}_{\boldsymbol{x},t}$ and $\|\boldsymbol{p}\| \leq M_{\boldsymbol{p}}$, we have

$$\begin{aligned}
\ell(\boldsymbol{y}, s, \boldsymbol{p}) &= -\sum_{i=1}^{n} S(y_i, s; p_i, a_i, b_i) + J^*(\boldsymbol{p}) \\
&\geq -\sum_{i=1}^{n} \left( S(x_i, t; p_i, a_i, b_i) + \frac{\delta}{n} \right) + J^*(\boldsymbol{p}) \\
&= -\sum_{i=1}^{n} S(x_i, t; p_i, a_i, b_i) + J^*(\boldsymbol{p}) - \delta \\
&= \ell(\boldsymbol{x}, t, \boldsymbol{p}) - \delta \geq -S(\boldsymbol{x}, t) - \delta.
\end{aligned} \tag{2.26}$$

Combining (2.25) and (2.26), we conclude that (2.24) holds for all $\boldsymbol{p} \in \mathbb{R}^n$ and $(\boldsymbol{y}, s) \in \tilde{\mathcal{N}}_{\boldsymbol{x},t} \cap (B_M(\mathbb{R}^n) \times [0, M])$. Therefore, the function $S$ is upper semi-continuous at $(\boldsymbol{x}, t)$.

Since $(\boldsymbol{x}, t)$ is an arbitrary point in $\mathbb{R}^n \times [0, +\infty)$, we conclude that the function $S$ is continuous in $\mathbb{R}^n \times [0, +\infty)$. □

*Lemma* 2.3.9. Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function. Let $\{a_i, b_i\}_{i=1}^n$ be positive constants and $K_i\colon \mathbb{R} \to \mathbb{R}$ be the function defined by (2.4) with constants $a = a_i$ and $b = b_i$ for each $i \in \{1, \ldots, n\}$. Let $\boldsymbol{p}^*(\boldsymbol{x}, t)$ be the set of maximizers in (2.22). When the maximizer is unique, we abuse notation and also use $\boldsymbol{p}^*(\boldsymbol{x}, t)$ to denote the unique maximizer (as opposed to the singleton containing the maximizer), whenever there is no ambiguity. Then, the following statements hold:

(a) For any $\boldsymbol{x} \in \mathbb{R}^n$ and $t > 0$, the maximizer in (2.22) exists and is unique.

(b) For any $\boldsymbol{x} \in \mathbb{R}^n$ and $t = 0$, the maximizer in (2.22) exists, and we have $\boldsymbol{p}^*(\boldsymbol{x}, 0) = \partial J(\boldsymbol{x})$.

(c) For any $\boldsymbol{x} \in \mathbb{R}^n$, $t \geq 0$, and any neighborhood $\mathcal{N}_{\boldsymbol{p}}$ of $\boldsymbol{p}^*(\boldsymbol{x}, t)$, there exists a neighborhood $\mathcal{N}_{\boldsymbol{x},t}$ of $(\boldsymbol{x}, t)$ in $\mathbb{R}^n \times [0, +\infty)$, such that $\boldsymbol{p}^*(\boldsymbol{y}, s) \subseteq \mathcal{N}_{\boldsymbol{p}}$ holds for all $(\boldsymbol{y}, s) \in \mathcal{N}_{\boldsymbol{x},t}$.

(d) The function $\boldsymbol{p}^*\colon \mathbb{R}^n \times (0, +\infty) \to \mathbb{R}^n$ is continuous.

(e) For any $R > 0$, the set $\{\boldsymbol{p}\colon \boldsymbol{x} \in B_R(\mathbb{R}^n),\ t \in [0, R],\ \boldsymbol{p} \in \boldsymbol{p}^*(\boldsymbol{x}, t)\}$ is bounded, where $B_R(\mathbb{R}^n)$ denotes the closed ball in $\mathbb{R}^n$ centered at $\mathbf{0}$ with radius $R$.

*Proof.* In the proof of Lemma 2.3.8, we have proved the existence of the maximizer in (2.22) for all $\boldsymbol{x} \in \mathbb{R}^n$ and $t \geq 0$. Hence, the set $\boldsymbol{p}^*(\boldsymbol{x}, t)$ is non-empty for all

$\boldsymbol{x} \in \mathbb{R}^n$ and $t \geq 0$. Let $\ell$ be the function defined in (2.21). Recall that the function $\ell$ is convex and lower semi-continuous with respect to $\boldsymbol{p}$. Now, we prove the statements as follows:

(a) Let $\boldsymbol{x} \in \mathbb{R}^n$ and $t > 0$. By Lemma 2.3.1, the function $p_i \mapsto -S(x_i, t; p_i, a_i, b_i)$ is strictly convex for each $i \in \{1, \ldots, n\}$. Therefore, the function $\boldsymbol{p} \mapsto \ell(\boldsymbol{x}, t, \boldsymbol{p})$ is strictly convex, and hence, the maximizer in (2.22) is unique. We have already proved the existence of the maximizer in the proof of Lemma 2.3.8. As a result, the set $\boldsymbol{p}^*(\boldsymbol{x}, t)$ is a singleton.

(b) Let $\boldsymbol{x} \in \mathbb{R}^n$ and $t = 0$. We have $S(x_i, 0; p_i, a_i, b_i) = p_i x_i$ for each $i \in \{1, \ldots, n\}$ and any $p_i \in \mathbb{R}$. Hence, the maximization problem in (2.22) becomes

$$S(\boldsymbol{x}, 0) = \sup_{\boldsymbol{p} \in \mathbb{R}^n} \left\{ \sum_{i=1}^{n} x_i p_i - J^*(\boldsymbol{p}) \right\} = \sup_{\boldsymbol{p} \in \mathbb{R}^n} \{\langle \boldsymbol{x}, \boldsymbol{p} \rangle - J^*(\boldsymbol{p})\} = J(\boldsymbol{x}). \quad (2.27)$$

The set of maximizers equals $\partial J(\boldsymbol{x})$, which is a non-empty, convex, and compact set.

(c) We prove this statement by contradiction. Assume it does not hold. Then, there exist $\boldsymbol{x} \in \mathbb{R}^n$, $t \geq 0$, an open neighborhood $\mathcal{N}_{\boldsymbol{p}}$ of $\boldsymbol{p}^*(\boldsymbol{x}, t)$, and a sequence $\{(\boldsymbol{y}^k, s^k)\}$ in $\mathbb{R}^n \times [0, +\infty)$ converging to $(\boldsymbol{x}, t)$, such that $\boldsymbol{p}^k \notin \mathcal{N}_{\boldsymbol{p}}$ holds for all $k \in \mathbb{N}$, where $\boldsymbol{p}^k$ is a maximizer in (2.22) at $(\boldsymbol{y}^k, s^k)$. Let $\delta > 0$ be any positive scalar. Since the function $S$ is continuous according to Lemma 2.3.8, we assume

$$|S(\boldsymbol{y}^k, s^k) - S(\boldsymbol{x}, t)| < \delta \quad \forall k \in \mathbb{N} \quad (2.28)$$

by taking the tail of the sequence $\{(\boldsymbol{y}^k, s^k)\}$. Note that the sequence $\{(\boldsymbol{y}^k, s^k)\}$ is bounded. Then, according to Lemma 2.3.7, there exists $M_{\boldsymbol{p}} > 0$, such that

there holds

$$\ell(\boldsymbol{y}^k, s^k, \boldsymbol{p}) \geq -S(\boldsymbol{x}, t) + \delta, \tag{2.29}$$

for all $\boldsymbol{p} \in \mathbb{R}^n$ satisfying $\|\boldsymbol{p}\| \geq M_{\boldsymbol{p}}$. Since $\boldsymbol{p}^k$ is a maximizer in (2.22) at $(\boldsymbol{y}^k, s^k)$, we have

$$\ell(\boldsymbol{y}^k, s^k, \boldsymbol{p}^k) = -S(\boldsymbol{y}^k, s^k) < -S(\boldsymbol{x}, t) + \delta \quad \forall k \in \mathbb{N}, \tag{2.30}$$

where the inequality follows from (2.28). Note that (2.30) contradicts with (2.29), and hence, we get $\|\boldsymbol{p}^k\| < M_{\boldsymbol{p}}$ for all $k \in \mathbb{N}$. Therefore, the sequence $\{\boldsymbol{p}^k\}$ is bounded. By taking a convergent subsequence, we assume $\{\boldsymbol{p}^k\}$ converges to a point in $\mathbb{R}^n$ denoted by $\boldsymbol{p}^0$. After some calculation, we obtain

$$S(\boldsymbol{x}, t) = \lim_{k\to\infty} S(\boldsymbol{y}^k, s^k) = -\lim_{k\to\infty} \ell(\boldsymbol{y}^k, s^k, \boldsymbol{p}^k) \leq -\ell(\boldsymbol{x}, t, \boldsymbol{p}^0),$$

where the first equality holds since $S$ is continuous by Lemma 2.3.8, the second equality holds since $\boldsymbol{p}^k$ is a maximizer in (2.22), and the last inequality holds since $\ell$ is lower semi-continuous and $\{(\boldsymbol{y}^k, s^k, \boldsymbol{p}^k)\}$ converges to $(\boldsymbol{x}, t, \boldsymbol{p}^0)$. Then, by definition of $S(\boldsymbol{x}, t)$, we conclude that $\boldsymbol{p}^0$ is a maximizer in (2.22) at $(\boldsymbol{x}, t)$. Hence, we have $\boldsymbol{p}^0 \in \boldsymbol{p}^*(\boldsymbol{x}, t) \subseteq \mathcal{N}_{\boldsymbol{p}}$. However, since $\boldsymbol{p}^0$ is the limit of $\{\boldsymbol{p}^k\}$ and each $\boldsymbol{p}^k$ is not in the open set $\mathcal{N}_{\boldsymbol{p}}$, the limit point $\boldsymbol{p}^0$ is also not in the set $\mathcal{N}_{\boldsymbol{p}}$, which gives a contradiction.

(d) By (a), the maximizer in (2.22) at any $(\boldsymbol{x}, t) \in \mathbb{R}^n \times (0, +\infty)$ is unique, and we denote the unique maximizer by $\boldsymbol{p}^*(\boldsymbol{x}, t)$. According to (c), for any $\boldsymbol{x} \in \mathbb{R}^n$, $t > 0$, and any neighborhood $\mathcal{N}_{\boldsymbol{p}}$ of $\boldsymbol{p}^*(\boldsymbol{x}, t)$, there exists a neighborhood $\mathcal{N}_{\boldsymbol{x},t}$ of $(\boldsymbol{x}, t)$ in $\mathbb{R}^n \times (0, +\infty)$, such that $\boldsymbol{p}^*(\boldsymbol{y}, s) \in \mathcal{N}_{\boldsymbol{p}}$ holds for all $(\boldsymbol{y}, s) \in \mathcal{N}_{\boldsymbol{x},t}$. This proves the continuity of the function $\boldsymbol{p}^* \colon \mathbb{R}^n \times (0, +\infty) \to \mathbb{R}^n$.

(e) We first prove the boundedness of $\boldsymbol{p}^*(\boldsymbol{x}, t)$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and $t \geq 0$. If $t > 0$, the set $\boldsymbol{p}^*(\boldsymbol{x}, t)$ is a singleton by (a), and hence, it is a bounded set. If $t = 0$, the set $\boldsymbol{p}^*(\boldsymbol{x}, t)$ equals $\partial J(\boldsymbol{x})$, which is a bounded set since $J$ is a finite-valued convex function. Now, we prove (e) by contradiction. Assume (e) does not hold. Then, there exist sequences $\{\boldsymbol{x}^k\} \subset B_R(\mathbb{R}^n)$, $\{t^k\} \subset [0, R]$, and $\{\boldsymbol{p}^k\} \subset \mathbb{R}^n$, such that $\boldsymbol{p}^k$ is in $\boldsymbol{p}^*(\boldsymbol{x}^k, t^k)$ for all $k \in \mathbb{N}$ and $\|\boldsymbol{p}^k\|$ increases to infinity as $k$ goes to infinity. Since $\{(\boldsymbol{x}^k, t^k)\}$ is a bounded sequence, by replacing it with a convergent subsequence, we can assume that $\{(\boldsymbol{x}^k, t^k)\}$ converges to a point denoted by $(\boldsymbol{x}^*, t^*) \in \mathbb{R}^n \times [0, +\infty)$. Let $\mathcal{N}_{\boldsymbol{p}}$ be a bounded neighborhood of $\boldsymbol{p}^*(\boldsymbol{x}^*, t^*)$, which exists since we have proved the boundedness of the set $\boldsymbol{p}^*(\boldsymbol{x}^*, t^*)$. According to (c), there exists a neighborhood $\mathcal{N}_{\boldsymbol{x}^*, t^*}$ of $(\boldsymbol{x}^*, t^*)$ such that $\boldsymbol{p}^*(\boldsymbol{y}, s) \subseteq \mathcal{N}_{\boldsymbol{p}}$ holds for all $(\boldsymbol{y}, s) \in \mathcal{N}_{\boldsymbol{x}^*, t^*}$. Since the sequence $\{(\boldsymbol{x}^k, t^k)\}$ converges to $(\boldsymbol{x}^*, t^*)$, by replacing it with a tail sequence, we can assume that $\{(\boldsymbol{x}^k, t^k)\}$ is in the neighborhood $\mathcal{N}_{\boldsymbol{x}^*, t^*}$. Hence, the set $\boldsymbol{p}^*(\boldsymbol{x}^k, t^k)$ is included in the bounded set $\mathcal{N}_{\boldsymbol{p}}$ for all $k \in \mathbb{N}$, which contradicts the assumption that $\boldsymbol{p}^k$ is in $\boldsymbol{p}^*(\boldsymbol{x}^k, t^k)$ for all $k \in \mathbb{N}$ and $\|\boldsymbol{p}^k\|$ increases to infinity as $k$ goes to infinity. Therefore, statement (e) holds.

□

*Lemma* 2.3.10. Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function. Let $\{a_i, b_i\}_{i=1}^n$ be positive constants and $K_i\colon \mathbb{R} \to \mathbb{R}$ be the function defined by (2.4) with constants $a = a_i$ and $b = b_i$ for each $i \in \{1, \ldots, n\}$. Let $S\colon \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$ be the function defined in (2.22). Then, the function $S$ is continuously differentiable in $\mathbb{R}^n \times (0, +\infty)$, and

its gradient at $(\boldsymbol{x}, t) \in \mathbb{R}^n \times (0, +\infty)$ equals

$$\nabla S(\boldsymbol{x}, t) = \left( \frac{\partial S(x_1, t; p_1^*(\boldsymbol{x}, t), a_1, b_1)}{\partial x}, \ldots, \frac{\partial S(x_n, t; p_n^*(\boldsymbol{x}, t), a_n, b_n)}{\partial x}, \right. \\ \left. \sum_{i=1}^{n} \frac{\partial S(x_i, t; p_i^*(\boldsymbol{x}, t), a_i, b_i)}{\partial t} \right), \tag{2.31}$$

where $\boldsymbol{p}^*(\boldsymbol{x}, t) = (p_1^*(\boldsymbol{x}, t), \ldots, p_n^*(\boldsymbol{x}, t))$ denotes the unique maximizer in (2.22) at $(\boldsymbol{x}, t)$ and the functions $\frac{\partial S(x_i, t; p_i^*(\boldsymbol{x}, t), a_i, b_i)}{\partial x}$ and $\frac{\partial S(x_i, t; p_i^*(\boldsymbol{x}, t), a_i, b_i)}{\partial t}$ on the right-hand side denote the derivatives of the function defined in (2.6) and (2.15) with respect to $x$ and $t$, respectively.

*Proof.* In this proof, whenever there is no ambiguity, we write $\boldsymbol{p}^* = (p_1^*, \ldots, p_n^*)$ instead of $\boldsymbol{p}^*(\boldsymbol{x}, t) = (p_1^*(\boldsymbol{x}, t), \ldots, p_n^*(\boldsymbol{x}, t))$, and we write $S_i(x, t; p)$ instead of $S(x, t; p, a_i, b_i)$, for simplicity. Let $\boldsymbol{x} \in \mathbb{R}^n$ and $t > 0$. Let $\boldsymbol{p}^* = (p_1^*, \ldots, p_n^*)$ be the maximizer in (2.22) at $(\boldsymbol{x}, t)$. By Lemma 2.3.9(a), the maximizer $\boldsymbol{p}^*$ exists and is unique.

We first compute the directional derivative of $S$ at $(\boldsymbol{x}, t)$. Consider the spatial direction $\boldsymbol{y} \in \mathbb{R}^n$ and time direction $s \in \mathbb{R}$. Let $(\boldsymbol{y}^k, s^k) \in \mathbb{R}^n \times (0, +\infty)$ be the perturbed vector around $(\boldsymbol{x}, t)$ along the direction $(\boldsymbol{y}, s)$. To be specific, define $\boldsymbol{y}^k := \boldsymbol{x} + \alpha^k \boldsymbol{y}$ and $s^k := t + \alpha^k s$ where $\{\alpha^k\}$ is a sequence of positive numbers converging to zero. Let $\boldsymbol{p}^k \in \mathbb{R}^n$ be the unique maximizer in (2.22) at $(\boldsymbol{y}^k, s^k)$. Denote the $i$-th component of $\boldsymbol{y}^k$ and $\boldsymbol{p}^k$ by $y_i^k$ and $p_i^k$, respectively. By definition of $S$, we have

$$S(\boldsymbol{y}^k, s^k) = \sum_{i=1}^{n} S_i(y_i^k, s^k; p_i^k) - J^*(\boldsymbol{p}^k) \geq \sum_{i=1}^{n} S_i(y_i^k, s^k; p_i^*) - J^*(\boldsymbol{p}^*),$$
$$S(\boldsymbol{x}, t) = \sum_{i=1}^{n} S_i(x_i, t; p_i^*) - J^*(\boldsymbol{p}^*) \geq \sum_{i=1}^{n} S_i(x_i, t; p_i^k) - J^*(\boldsymbol{p}^k),$$

for all $k \in \mathbb{N}$. On the one hand, we have

$$\begin{aligned}
&\liminf_{k\to\infty} \frac{S(\boldsymbol{y}^k, s^k) - S(\boldsymbol{x}, t)}{\alpha^k} \\
\geq &\liminf_{k\to\infty} \frac{\sum_{i=1}^n S_i(y_i^k, s^k; p_i^*) - J^*(\boldsymbol{p}^*) - (\sum_{i=1}^n S_i(x_i, t; p_i^*) - J^*(\boldsymbol{p}^*))}{\alpha^k} \\
= &\sum_{i=1}^n \liminf_{k\to\infty} \frac{S_i(y_i^k, s^k; p_i^*) - S_i(x_i, t; p_i^*)}{\alpha^k} \\
= &\sum_{i=1}^n \langle \nabla S_i(x_i, t; p_i^*), (y_i, s) \rangle,
\end{aligned}$$

where $\nabla S_i(x_i, t; p_i^*)$ denotes the gradient of the function $(x, t) \mapsto S(x, t; p_i^*, a_i, b_i)$ at $(x_i, t)$ and $y_i$ denotes the $i$-th component of $\boldsymbol{y}$. On the other hand, we have

$$\begin{aligned}
&\limsup_{k\to\infty} \frac{S(\boldsymbol{y}^k, s^k) - S(\boldsymbol{x}, t)}{\alpha^k} \\
\leq &\limsup_{k\to\infty} \frac{\sum_{i=1}^n S_i(y_i^k, s^k; p_i^k) - J^*(\boldsymbol{p}^k) - \left(\sum_{i=1}^n S_i(x_i, t; p_i^k) - J^*(\boldsymbol{p}^k)\right)}{\alpha^k} \\
= &\sum_{i=1}^n \limsup_{k\to\infty} \frac{S_i(y_i^k, s^k; p_i^k) - S_i(x_i, t; p_i^k)}{\alpha^k} \\
= &\sum_{i=1}^n \limsup_{k\to\infty} \langle \nabla S_i(x_i + \xi_i^k y_i, t + \xi_i^k s; p_i^k), (y_i, s) \rangle, \\
= &\sum_{i=1}^n \langle \nabla S_i(x_i, t; p_i^*), (y_i, s) \rangle,
\end{aligned}$$

where the second equality holds for some constants $\xi_i^k \in [0, \alpha^k]$ for each $i \in \{1, \ldots, n\}$ and $k \in \mathbb{N}$ by Taylor's theorem and the last equality holds since we have $\lim_{k\to\infty} \xi_i^k = 0$ for each $i \in \{1, \ldots, n\}$, $\lim_{k\to\infty} \boldsymbol{p}^k = \boldsymbol{p}^*$ by Lemma 2.3.9(d), and each function $S_i$ is continuously differentiable with respect to $(x, t, p)$ by Lemmas 2.3.1 and 2.3.2.

Therefore, we conclude that

$$\begin{aligned}
&\lim_{k\to\infty} \frac{S(\boldsymbol{y}^k, s^k) - S(\boldsymbol{x}, t)}{\alpha^k} \\
&= \sum_{i=1}^{n} \langle \nabla S_i(x_i, t; p_i^*), (y_i, s) \rangle \\
&= \left\langle \left( \frac{\partial S_1}{\partial x}(x_1, t; p_1^*), \ldots, \frac{\partial S_n}{\partial x}(x_n, t; p_n^*), \sum_{i=1}^{n} \frac{\partial S_i}{\partial t}(x_i, t; p_i^*) \right), (\boldsymbol{y}, s) \right\rangle.
\end{aligned}$$

This equality holds for any direction $(\boldsymbol{y}, s) \in \mathbb{R}^n \times (0, +\infty)$. As a result, the function $S$ is differentiable at $(\boldsymbol{x}, t)$, and the gradient satisfies (2.31). Note that $(\boldsymbol{x}, t)$ is an arbitrary point in $\mathbb{R}^n \times (0, +\infty)$, and hence, the function $S$ is differentiable in $\mathbb{R}^n \times (0, +\infty)$ with gradient equal to (2.31).

It remains to prove the continuity of the gradient of $S$. By Lemmas 2.3.1 and 2.3.2, each function $(x, t, p) \mapsto S(x, t; p, a_i, b_i)$ on the right-hand side of (2.31) is continuously differentiable in $\mathbb{R} \times (0, +\infty) \times \mathbb{R}$. Moreover, the function $\boldsymbol{p}^*$ is also continuous by Lemma 2.3.9(d). Therefore, the right-hand side of (2.31) is continuous with respect to $(\boldsymbol{x}, t) \in \mathbb{R}^n \times (0, +\infty)$. As a result, the function $S$ is continuously differentiable with respect to $(\boldsymbol{x}, t)$ in $\mathbb{R}^n \times (0, +\infty)$, and the gradient satisfies (2.31). □

*Lemma* 2.3.11. Assume $J\colon \mathbb{R}^n \to \mathbb{R}$ is a continuous function satisfying (2.24) and is bounded from below by an affine function. Let $K\colon \mathbb{R}^n \to (-\infty, 0]$ be a Lipschitz continuous function and $M$ be a symmetric positive definite matrix with $n$ rows and $n$ columns. Let $S\colon \mathbb{R}^n \times [0, +\infty) \to \mathbb{R} \cup \{-\infty\}$ be the value function defined by (2.1). Then, the following statements holds:

(a) The function $S$ is finite-valued for all $\boldsymbol{x} \in \mathbb{R}^n$ and $t \geq 0$, and it is a viscosity solution to the HJ PDE (2.2) in the solution set $\mathcal{G}$ defined in (2.25).

(b) Assume that for all $\boldsymbol{\alpha} \in \mathbb{R}^n$ such that $\boldsymbol{x} \mapsto J(\boldsymbol{x}) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle$ is bounded from below, the function $(\boldsymbol{x}, t) \mapsto S(\boldsymbol{x}, t) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle$ is also bounded from below. Then, the function $S$ is the unique viscosity solution to the HJ PDE (2.1) in the solution set $\mathcal{G}$.

*Proof.* (a) Since $J$ is bounded from below by an affine function, there exists a vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ and a scalar $\beta \in \mathbb{R}$ satisfying $J(\boldsymbol{x}) \geq \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle + \beta$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Define $\tilde{J} \colon \mathbb{R}^n \to \mathbb{R}$ by

$$\tilde{J}(\boldsymbol{x}) := J(\boldsymbol{x}) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle \quad \forall \boldsymbol{x} \in \mathbb{R}^n. \tag{2.32}$$

Then, the function $\tilde{J}$ is bounded from below. Now, consider another optimal control problem, which reads:

$$\tilde{S}(\boldsymbol{x}, t) = \inf \left\{ \int_0^t \ell(\boldsymbol{x}(s), \boldsymbol{u}(s)) ds + \tilde{J}(\boldsymbol{x}(0)) : \right. \\ \left. \dot{\boldsymbol{x}}(s) = f(\boldsymbol{x}(s), \boldsymbol{u}(s)) \ \forall s \in (0, t), \ \boldsymbol{x}(t) = \boldsymbol{x} \right\}, \tag{2.33}$$

where the Lagrangian function $\ell \colon \mathbb{R}^n \times A \to \mathbb{R}$ and the source term $f \colon \mathbb{R}^n \times A \to \mathbb{R}$ are defined by

$$\ell(\boldsymbol{x}, \boldsymbol{u}) := \frac{1}{2} \|\boldsymbol{u}\|_{M^{-1}}^2 - K(\boldsymbol{x}) - \frac{1}{2} \|\boldsymbol{\alpha}\|_M^2, \quad f(\boldsymbol{x}, \boldsymbol{u}) := \boldsymbol{u} + M\boldsymbol{\alpha}, \quad \forall \boldsymbol{x}, \boldsymbol{u} \in \mathbb{R}^n. \tag{2.34}$$

Here and in the rest of this proof, the set $A$ denotes the domain of the control variable $\boldsymbol{u}$, which equals $\mathbb{R}^n$ in our case. By straightforward calculation, the cost in (2.33)

equals

$$
\begin{aligned}
&\int_0^t \left( \frac{\|\dot{\boldsymbol{x}}(s) - M\boldsymbol{\alpha}\|_{M^{-1}}^2}{2} - K(\boldsymbol{x}(s)) - \frac{\|\boldsymbol{\alpha}\|_M^2}{2} \right) ds + \tilde{J}(\boldsymbol{x}(0)) \\
= &\int_0^t \left( \frac{\|\dot{\boldsymbol{x}}(s)\|_{M^{-1}}^2}{2} - K(\boldsymbol{x}(s)) - \langle \boldsymbol{\alpha}, \dot{\boldsymbol{x}}(s) \rangle \right) ds + J(\boldsymbol{x}(0)) - \langle \boldsymbol{\alpha}, \boldsymbol{x}(0) \rangle \\
= &\int_0^t \left( \frac{\|\dot{\boldsymbol{x}}(s)\|_{M^{-1}}^2}{2} - K(\boldsymbol{x}(s)) \right) ds - \langle \boldsymbol{\alpha}, \boldsymbol{x}(t) - \boldsymbol{x}(0) \rangle + J(\boldsymbol{x}(0)) - \langle \boldsymbol{\alpha}, \boldsymbol{x}(0) \rangle \\
= &\int_0^t \left( \frac{\|\dot{\boldsymbol{x}}(s)\|_{M^{-1}}^2}{2} - K(\boldsymbol{x}(s)) \right) ds + J(\boldsymbol{x}(0)) - \langle \boldsymbol{\alpha}, \boldsymbol{x}(t) \rangle.
\end{aligned}
$$

As a result, there holds

$$
\begin{aligned}
\tilde{S}(\boldsymbol{x}, t) = \inf \Bigg\{ &\int_0^t \left( \frac{\|\boldsymbol{u}(s)\|_{M^{-1}}^2}{2} - K(\boldsymbol{x}) - \frac{\|\boldsymbol{\alpha}\|_M^2}{2} \right) ds + \tilde{J}(\boldsymbol{x}(0)) \\
&: \dot{\boldsymbol{x}}(s) = \boldsymbol{u}(s) + M\boldsymbol{\alpha} \ \forall s \in (0, t), \ \boldsymbol{x}(t) = \boldsymbol{x} \Bigg\} \\
= \inf \left\{ \int_0^t \left( \frac{\|\dot{\boldsymbol{x}}(s) - M\boldsymbol{\alpha}\|_{M^{-1}}^2}{2} - K(\boldsymbol{x}) - \frac{\|\boldsymbol{\alpha}\|_M^2}{2} \right) ds + \tilde{J}(\boldsymbol{x}(0)) \colon \boldsymbol{x}(t) = \boldsymbol{x} \right\} \\
= \inf \left\{ \int_0^t \left( \frac{\|\dot{\boldsymbol{x}}(s)\|_{M^{-1}}^2}{2} - K(\boldsymbol{x}(s)) \right) ds + J(\boldsymbol{x}(0)) - \langle \boldsymbol{\alpha}, \boldsymbol{x}(t) \rangle \colon \boldsymbol{x}(t) = \boldsymbol{x} \right\} \\
= \inf \left\{ \int_0^t \left( \frac{\|\dot{\boldsymbol{x}}(s)\|_{M^{-1}}^2}{2} - K(\boldsymbol{x}(s)) \right) ds + J(\boldsymbol{x}(0)) \colon \boldsymbol{x}(t) = \boldsymbol{x} \right\} - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle \\
= S(\boldsymbol{x}, t) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle.
\end{aligned}
$$

Therefore, we have

$$
\tilde{S}(\boldsymbol{x}, t) = S(\boldsymbol{x}, t) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle \quad \forall \boldsymbol{x} \in \mathbb{R}^n, t \geq 0. \tag{2.35}
$$

Now, by applying [9, Theorem 3.2], we will prove that the function $\tilde{S}$ defined in (2.33) is the unique viscosity solution to the HJ PDE defined by

$$
\begin{cases}
\dfrac{\partial \tilde{S}}{\partial t}(\boldsymbol{x}, t) + \dfrac{1}{2} \|\nabla_{\boldsymbol{x}} \tilde{S}(\boldsymbol{x}, t) + \boldsymbol{\alpha}\|_M^2 + K(\boldsymbol{x}) = 0 & \boldsymbol{x} \in \mathbb{R}^n, t \in (0, +\infty), \\
\tilde{S}(\boldsymbol{x}, 0) = \tilde{J}(\boldsymbol{x}) & \boldsymbol{x} \in \mathbb{R}^n,
\end{cases} \tag{2.36}
$$

in the solution set $\tilde{\mathcal{G}}$ defined by

$$\tilde{\mathcal{G}} := \{W \in C(\mathbb{R}^n \times [0,+\infty)) \colon W \text{ is bounded from below in } \mathbb{R}^n \times [0,T]\, \forall T > 0 \text{ and } \|W\|_R < +\infty\ \forall R > 0\}, \tag{2.37}$$

where $\|\cdot\|_R$ is defined in (2.26). Here, we need to check the assumptions of [9, Theorem 3.2], which include:

($H_0$) The domain $A$ of the control variable $\boldsymbol{u}$ is a closed subset of a normed space.

($H_1$) $f \colon \mathbb{R}^n \times A \to \mathbb{R}^n$ is continuous, and there exists a constant $L > 0$, such that

$$\langle f(\boldsymbol{x},\boldsymbol{u}) - f(\boldsymbol{y},\boldsymbol{u}), \boldsymbol{x}-\boldsymbol{y}\rangle \le L\|\boldsymbol{x}-\boldsymbol{y}\|^2 \quad \forall \boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^n, \boldsymbol{u} \in A.$$

($H_2$) $\ell \colon \mathbb{R}^n \times A \to \mathbb{R}$ is continuous and bounded from below.

($H_3$) $\tilde{J} \colon \mathbb{R}^n \to \mathbb{R}$ is continuous and bounded from below.

($H_4$) $\lambda \ge 0$. Note that $\lambda$ is the discounting factor in [9]. In our case, there is no discounting factor, and hence, $\lambda$ is always zero.

($H_5$) There exists $\sigma \ge 1$, such that for any compact $K \subseteq \mathbb{R}^n$, there exists a constant $f_K > 0$ satisfying

$$\|f(\boldsymbol{x},\boldsymbol{u})\| \le f_K(1+\|\boldsymbol{u}\|^\sigma) \quad \forall (\boldsymbol{x},\boldsymbol{u}) \in K \times A.$$

($H_6$) There exist $\ell_0 > 0$, $C_0 \ge 0$, $\delta_1 > \sigma$ (where $\sigma$ is the constant in (H5)), such that

$$\ell(\boldsymbol{x},\boldsymbol{u}) \ge \ell_0\|\boldsymbol{u}\|^{\delta_1} - C_0 \quad \forall (\boldsymbol{x},\boldsymbol{u}) \in \mathbb{R}^n \times A.$$

(2.1) There exist $\delta_2 \geq 0$ and $\bar{\ell} > 0$, such that there holds

$$|\ell(\boldsymbol{x}, \boldsymbol{u}) - \ell(\boldsymbol{y}, \boldsymbol{u})| \leq \bar{\ell}\|\boldsymbol{x} - \boldsymbol{y}\|(1 + \|\boldsymbol{u}\|^{\delta_1} + \|\boldsymbol{x}\|^{\delta_2} + \|\boldsymbol{y}\|^{\delta_2}) \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \boldsymbol{u} \in A. \tag{2.38}$$

(2.13) There exists $\delta_2 \geq 0$ and $\bar{g} > 0$, such that there holds

$$|\tilde{J}(\boldsymbol{x}) - \tilde{J}(\boldsymbol{y})| \leq \bar{g}\|\boldsymbol{x} - \boldsymbol{y}\|(1 + \|\boldsymbol{x}\|^{\delta_2} + \|\boldsymbol{y}\|^{\delta_2}) \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \boldsymbol{u} \in A. \tag{2.39}$$

In our case, the set $A$ equals $\mathbb{R}^n$, the function $f$ does not depend on the state variable $\boldsymbol{x}$, and there is no discounting factor in the optimal control problem (i.e., the parameter $\lambda$ in [9] is zero). Hence, assumptions $(H_0)$, $(H_1)$, and $(H_4)$ hold. By definition of $f$ in (2.34), assumption $(H_5)$ holds with $\sigma = 1$ and $f_K = \max\{1, \|M\boldsymbol{\alpha}\|\}$. Since the potential energy $K$ is continuous and non-positive, the function $\ell$ is continuous and bounded from below by $-\frac{1}{2}\|\boldsymbol{\alpha}\|_M^2$, which implies $(H_2)$. Moreover, we obtain a lower bound for $\ell$ by

$$\ell(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{u}\|_{M^{-1}}^2 - K(\boldsymbol{x}) - \frac{1}{2}\|\boldsymbol{\alpha}\|_M^2 \geq \frac{1}{2}\|\boldsymbol{u}\|_{M^{-1}}^2 - \frac{1}{2}\|\boldsymbol{\alpha}\|_M^2 \geq \ell_0\|\boldsymbol{u}\|^2 - \frac{1}{2}\|\boldsymbol{\alpha}\|_M^2,$$

for any $\boldsymbol{x}, \boldsymbol{u} \in \mathbb{R}^n$, where $\ell_0$ is the smallest eigenvalue of the matrix $M^{-1}$, which is positive since $M^{-1}$ is symmetric positive definite. Hence, assumption $(H_6)$ holds for $\delta_1 = 2$ and $C_0 = \frac{1}{2}\|\boldsymbol{\alpha}\|_M^2$. Note that we have $\sigma = 1$ in $(H_5)$ and $\delta_1 = 2$ in $(H_6)$, and thus, the inequality $\sigma < \delta_1$ in $(H_6)$ is satisfied. After some calculation, we have

$$|\ell(\boldsymbol{x}, \boldsymbol{u}) - \ell(\boldsymbol{y}, \boldsymbol{u})| = |K(\boldsymbol{x}) - K(\boldsymbol{y})| \leq \bar{\ell}\|\boldsymbol{x} - \boldsymbol{y}\| \quad \forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u} \in \mathbb{R}^n,$$

where $\bar{\ell}$ is the Lipschitz constant of $K$, and hence, (2.38) holds. Now, it remains to check the assumptions for the initial condition $\tilde{J}$. Recall that $\tilde{J}$ is continuous and bounded from below, and thus, $(H_3)$ is satisfied. By assumption, $J$ satisfies (2.24).

Then, by (2.32), we obtain

$$|\tilde{J}(\boldsymbol{x})-\tilde{J}(\boldsymbol{y})| \leq |J(\boldsymbol{x})-J(\boldsymbol{y})|+\|\boldsymbol{\alpha}\|\|\boldsymbol{x}-\boldsymbol{y}\| \leq C\|\boldsymbol{x}-\boldsymbol{y}\|(1+\|\boldsymbol{x}\|^{\delta}+\|\boldsymbol{y}\|^{\delta})+\|\boldsymbol{\alpha}\|\|\boldsymbol{x}-\boldsymbol{y}\|,$$

for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. Hence, (2.39) holds with $\bar{g} = C + \|\boldsymbol{\alpha}\|$ and $\delta_2 = \delta$. Therefore, all the assumptions in [9, Theorem 3.2] are satisfied. Then, applying [9, Theorem 3.2] (with the time reversal technique applied to the optimal control problem) to any time horizon $T > 0$, the function $\tilde{S}$ defined in (2.33) is the unique viscosity solution in the solution set $\tilde{\mathcal{G}}$ in (2.37) to the HJ PDE whose initial condition is $\tilde{J}$, and the Hamiltonian equals

$$\begin{aligned} H(\boldsymbol{x}, \boldsymbol{p}) &= \sup_{\boldsymbol{u}\in\mathbb{R}^n} \{\langle f(\boldsymbol{x}, \boldsymbol{u}), \boldsymbol{p}\rangle - \ell(\boldsymbol{x}, \boldsymbol{u})\} \\ &= \sup_{\boldsymbol{u}\in\mathbb{R}^n} \left\{ \langle \boldsymbol{u} + M\boldsymbol{\alpha}, \boldsymbol{p}\rangle - \frac{1}{2}\|\boldsymbol{u}\|^2_{M^{-1}} + K(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{\alpha}\|^2_M \right\} \\ &= \frac{1}{2}\|\boldsymbol{p} + \boldsymbol{\alpha}\|^2_M + K(\boldsymbol{x}). \end{aligned}$$

Therefore, the function $\tilde{S}$ is the unique viscosity solution to the HJ PDE (2.36) in the solution set $\tilde{\mathcal{G}}$. By straightforward calculation using (2.35), we obtain

$$D^- S(\boldsymbol{x}, t) = D^- \tilde{S}(\boldsymbol{x}, t) + \{(\boldsymbol{\alpha}, 0)\}, \quad D^+ S(\boldsymbol{x}, t) = D^+ \tilde{S}(\boldsymbol{x}, t) + \{(\boldsymbol{\alpha}, 0)\}, \tag{2.40}$$

for all $\boldsymbol{x} \in \mathbb{R}^n, t > 0$, where $D^- W$ and $D^+ W$ respectively denote the subdifferential and superdifferential of a continuous function $W$. By applying (2.40), (2.32), and (2.35) to (2.36), we conclude that the function $S$ is a viscosity solution to the HJ PDE (2.2).

To prove statement (a), it remains to prove that the function $S$ is in the solution set $\mathcal{G}$. Since the function $\tilde{S}$ is in the solution set $\tilde{\mathcal{G}}$, the function $\tilde{S}$ is continuous in $\mathbb{R}^n \times [0, +\infty)$, and hence, $S$ is also in continuous in $\mathbb{R}^n \times [0, +\infty)$. For all $T > 0$,

the function $\tilde{S}$ is bounded from below in $\mathbb{R}^n \times [0,T]$, and thus, the function $(\boldsymbol{x},t) \mapsto S(\boldsymbol{x},t) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle$ is bounded from below in $\mathbb{R}^n \times [0,T]$. Moreover, by straightforward calculation, for all $R > 0$, there holds

$$\begin{aligned}
\|S\|_R &= \sup \left\{ |S(\boldsymbol{x},t)| + \|\boldsymbol{p}\| \colon (\boldsymbol{x},t) \in B_R(\mathbb{R}^n) \times [0,R], \boldsymbol{p} \in D_{\boldsymbol{x}}^- S(\boldsymbol{x},t) \right\} \\
&= \sup \left\{ \left| \tilde{S}(\boldsymbol{x},t) + \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle \right| + \|\boldsymbol{q} + \boldsymbol{\alpha}\| \colon (\boldsymbol{x},t) \in B_R(\mathbb{R}^n) \times [0,R], \boldsymbol{q} \in D_{\boldsymbol{x}}^- \tilde{S}(\boldsymbol{x},t) \right\} \\
&\leq \sup \left\{ \left| \tilde{S}(\boldsymbol{x},t) \right| + \|\boldsymbol{q}\| \colon (\boldsymbol{x},t) \in B_R(\mathbb{R}^n) \times [0,R], \boldsymbol{q} \in D_{\boldsymbol{x}}^- \tilde{S}(\boldsymbol{x},t) \right\} + \|\boldsymbol{\alpha}\| R + \|\boldsymbol{\alpha}\| \\
&= \|\tilde{S}\|_R + \|\boldsymbol{\alpha}\| R + \|\boldsymbol{\alpha}\| < +\infty,
\end{aligned}$$

where the second equality follows from (2.35) and (2.40). Therefore, the function $S$ is in the solution set $\mathcal{G}$, and the statement is proved.

(b) Let $W$ be a viscosity solution to (2.2) in the solution set $\mathcal{G}$. By definition of $\mathcal{G}$, there exists $\boldsymbol{\alpha} \in \mathbb{R}^n$, such that for all $T > 0$, the function $(\boldsymbol{x},t) \mapsto W(\boldsymbol{x},t) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle$ is bounded from below in $\mathbb{R}^n \times [0,T]$. By picking $t = 0$, we get that the function $\boldsymbol{x} \mapsto J(\boldsymbol{x}) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle = W(\boldsymbol{x},0) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle$ is bounded from below in $\mathbb{R}^n$. Hence, by assumption, the function $(\boldsymbol{x},t) \mapsto S(\boldsymbol{x},t) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle$ is also bounded from below in $\mathbb{R}^n \times [0,+\infty)$. Define the functions $\tilde{S}, \tilde{W} \colon \mathbb{R}^n \times [0,+\infty) \to \mathbb{R}$ by

$$\tilde{S}(\boldsymbol{x},t) = S(\boldsymbol{x},t) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle, \quad \tilde{W}(\boldsymbol{x},t) = W(\boldsymbol{x},t) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle, \quad \forall \boldsymbol{x} \in \mathbb{R}^n,\, t \geq 0. \tag{2.41}$$

By definition, (2.40) holds for both functions $S$ and $W$. Note that $S$ and $W$ are both viscosity solutions to the HJ PDE (2.5) in the solution set $\mathcal{G}$ (recall that $S$ was proved to be a viscosity solution in (a)). By straightforward calculation using (2.40), the functions $\tilde{S}$ and $\tilde{W}$ are two viscosity solutions to the HJ PDE (2.36) in the solution set $\tilde{\mathcal{G}}$. However, by [9, Theorem 3.2], the viscosity solution to (2.36) is unique in $\tilde{\mathcal{G}}$, which implies $\tilde{S} = \tilde{W}$, and hence, $S = W$ holds. Therefore, $S$ is the unique viscosity solution to the HJ PDE (2.2) in the solution set $\mathcal{G}$. □

### 2.3.2 Proof of Proposition 2.1.3

(a) By Lemma 2.3.8, the function $S$ is continuous in $\mathbb{R}^n \times [0, +\infty)$. By Lemma 2.3.10, the function $S$ is continuously differentiable, and its gradient at any point $(\boldsymbol{x}, t) \in \mathbb{R}^n \times (0, +\infty)$ satisfies

$$\begin{aligned}\frac{\partial S(\boldsymbol{x}, t)}{\partial t} &= \sum_{i=1}^{n} \frac{\partial S}{\partial t}(x_i, t; p_i^*(\boldsymbol{x}, t), a_i, b_i) \\ &= -\sum_{i=1}^{n} \left( \frac{1}{2} \left( \frac{\partial S}{\partial x}(x_i, t; p_i^*(\boldsymbol{x}, t), a_i, b_i) \right)^2 + K_i(x_i) \right) \\ &= -\frac{1}{2} \|\nabla_{\boldsymbol{x}} S(\boldsymbol{x}, t)\|^2 - K(\boldsymbol{x}),\end{aligned}$$

where $p_i^*(\boldsymbol{x}, t)$ denotes the $i$-th component of the unique maximizer in (2.22) at $(\boldsymbol{x}, t)$, the first and the third equalities hold by (2.31), and the second equality holds since each function $(x_i, t) \mapsto S(x_i, t; p_i^*, a_i, b_i)$ satisfies their corresponding one-dimensional HJ PDE by Lemma 2.3.2. Hence, the function $S$ satisfies the differential equation in (2.21). Also, the initial condition is satisfied according to (2.27) in the proof of Lemma 2.3.9. Therefore, the function $S$ is a continuously differentiable solution to the HJ PDE (2.21).

(b) To prove that the function $S$ is the unique viscosity solution in the solution set $\mathcal{G}$, we first prove that the function $S$ is in $\mathcal{G}$. Let $\boldsymbol{p}$ be any vector in the domain

of $J^*$. By (2.22) and Lemma 2.3.3, we have

$$\begin{aligned} S(\boldsymbol{x},t) &\geq \sum_{i=1}^{n} S(x_i,t;p_i,a_i,b_i) - J^*(\boldsymbol{p}) \geq \sum_{i=1}^{n}(p_i x_i + C_i) - J^*(\boldsymbol{p}) \\ &= \langle \boldsymbol{p},\boldsymbol{x}\rangle + \sum_{i=1}^{n} C_i - J^*(\boldsymbol{p}), \end{aligned} \tag{2.42}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $t \geq 0$, where $C_i$ is the constant $C$ in the lower bound in Lemma 2.3.3 with constants $a = a_i$, $b = b_i$, and $p = p_i$. Hence, $S$ is bounded below by an affine function. Then, to prove $S \in \mathcal{G}$, it remains to prove that $\|S\|_R$ is finite for all $R > 0$. Let $R > 0$ be an arbitrary number, and let $\boldsymbol{p}^*(\boldsymbol{x},t)$ denote the set of maximizers in (2.22) for any $\boldsymbol{x} \in \mathbb{R}^n$ and $t \geq 0$. If the set is a singleton, by a slight abuse of notation, we denote both the set and the element in the set by $\boldsymbol{p}^*(\boldsymbol{x},t)$ and we denote the $i$-th component of the element by $p_i^*(\boldsymbol{x},t)$. By Lemma 2.3.10 and straightforward computation, we get

$$D_{\boldsymbol{x}}^- S(\boldsymbol{x},t) = \begin{cases} \{\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)\} & \boldsymbol{x} \in \mathbb{R}^n, t > 0, \\ \partial J(\boldsymbol{x}) = \boldsymbol{p}^*(\boldsymbol{x},0) & \boldsymbol{x} \in \mathbb{R}^n, t = 0, \end{cases} \tag{2.43}$$

where

$$\{\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)\} = \left\{ \left( \frac{\partial S(x_1,t;p_1^*(\boldsymbol{x},t),a_1,b_1)}{\partial x}, \ldots, \frac{\partial S(x_n,t;p_n^*(\boldsymbol{x},t),a_n,b_n)}{\partial x} \right) \right\}.$$

For any $x, p \in \mathbb{R}$ and $a, b > 0$, the function $x \mapsto S(x,0;p,a,b)$ equals $xp$, and hence we get $\frac{\partial S(x,0;p,a,b)}{\partial x} = p$. Therefore, (2.43) is simplified to

$$\begin{aligned} &D_{\boldsymbol{x}}^- S(\boldsymbol{x},t) \\ &= \left\{ \left( \frac{\partial S(x_1,t;p_1,a_1,b_1)}{\partial x}, \ldots, \frac{\partial S(x_n,t;p_n,a_n,b_n)}{\partial x} \right) : (p_1,\ldots,p_n) \in \boldsymbol{p}^*(\boldsymbol{x},t) \right\}, \end{aligned} \tag{2.44}$$

for any $\boldsymbol{x} \in \mathbb{R}^n, t \geq 0$. By Lemma 2.3.9(e), the set $\{\|\boldsymbol{p}\| : \boldsymbol{x} \in B_R(\mathbb{R}^n),\ t \in [0,R],\ \boldsymbol{p} \in \boldsymbol{p}^*(\boldsymbol{x},t)\}$ is bounded for all $R > 0$, and we denote the bound by $R_p$. Then, by Lemma 2.3.4, for all $\boldsymbol{x} \in B_R(\mathbb{R}^n)$, $t \in [0,R]$, and $\boldsymbol{p} \in \boldsymbol{p}^*(\boldsymbol{x},t)$, we get

$$\begin{aligned} &\left\| \left( \frac{\partial S(x_1,t;p_1,a_1,b_1)}{\partial x}, \ldots, \frac{\partial S(x_n,t;p_n,a_n,b_n)}{\partial x} \right) \right\| \\ =&\sqrt{\sum_{i=1}^n \left| \frac{\partial S(x_i,t;p_i,a_i,b_i)}{\partial x} \right|^2} \leq \sqrt{\sum_{i=1}^n C_i(R,R,R_p)^2}, \end{aligned} \tag{2.45}$$

where $C_i$ is the function in the upper bound defined in Lemma 2.3.4 for the parameters $a_i$ and $b_i$. (Note that in different contexts, we may reuse the notation $C_i$ to denote different bounds if there is no ambiguity.) Combining (2.44) and (2.45), we have

$$\|S\|_R \leq \sup_{\boldsymbol{x} \in B_R(\mathbb{R}), t \in [0,R]} |S(\boldsymbol{x},t)| + \sqrt{\sum_{i=1}^n C_i(R,R,R_p)^2} < +\infty.$$

Therefore, $S$ is a function in $\mathcal{G}$. We have proved in (a) that $S$ is a continuously differentiable solution, and hence, $S$ is a viscosity solution in $\mathcal{G}$.

Then, we apply Lemma 2.3.11 to prove the uniqueness of the viscosity solution. To apply Lemma 2.3.11, we need to check its assumptions. The assumption on $J$ is satisfied since $J$ is a convex function satisfying (2.24). The assumption on $M$ is satisfied since $M$ is the identity matrix in this section. The assumption on $K$ is satisfied since $K$ is a non-positive 1-homogeneous concave function, which implies that $K$ is Lipschitz continuous. Now, it remains to check the assumption on $S$ in Lemma 2.3.11(b). Let $\boldsymbol{\alpha} \in \mathbb{R}^n$ be a vector such that $\boldsymbol{x} \mapsto J(\boldsymbol{x}) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle$ is bounded from below. The convexity of $J$ implies that $\boldsymbol{\alpha}$ is in the domain of $J^*$. Using (2.42) with $\boldsymbol{p} = \boldsymbol{\alpha}$, we have that

$$S(\boldsymbol{x},t) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle \geq \sum_{i=1}^n C_i - J^*(\boldsymbol{\alpha}) \in \mathbb{R}.$$

In other words, the function $(\boldsymbol{x}, t) \mapsto S(\boldsymbol{x}, t) - \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle$ is bounded from below. Therefore, the assumptions for Lemma 2.3.11(b) holds, and the viscosity solution to (2.21) in $\mathcal{G}$ is unique. □

### 2.3.3 Proof of Proposition 2.1.4

Let $\boldsymbol{x} \in \mathbb{R}^n$ and $t > 0$. In this proof, we write $\boldsymbol{\gamma}(s)$ instead of $\boldsymbol{\gamma}(s; \boldsymbol{x}, t)$ whenever there is no ambiguity. By Lemma 2.3.5 and the definition of $\boldsymbol{\gamma}$, we have

$$\begin{aligned} &\int_0^t \left( \frac{\|\dot{\boldsymbol{\gamma}}(s)\|^2}{2} - K(\boldsymbol{\gamma}(s)) \right) ds + J(\boldsymbol{\gamma}(0)) \\ &= \sum_{i=1}^n \int_0^t \left( \frac{(\dot{\boldsymbol{\gamma}}_i(s))^2}{2} - K_i(\boldsymbol{\gamma}_i(s)) \right) ds + J(\boldsymbol{\gamma}(0)) \\ &= \sum_{i=1}^n S(x_i, t; p_i^*, a_i, b_i) - \langle \boldsymbol{p}^*, \boldsymbol{\gamma}(0) \rangle + J(\boldsymbol{\gamma}(0)), \end{aligned} \tag{2.46}$$

where $\boldsymbol{\gamma}_i$ denotes the $i$-th component of $\boldsymbol{\gamma}$ and $\boldsymbol{p}^* = (p_1^*, \ldots, p_n^*)$ is the unique maximizer in (2.22). By Lemma 2.3.6, for each $i \in \{1, \ldots, n\}$, we have $\frac{\partial S}{\partial p}(x_i, t; p_i^*, a_i, b_i) = \gamma(0; x_i, t, p_i^*, a_i, b_i)$. Note that (2.22) is a concave optimization problem by Lemma 2.3.1 and the convexity of $J^*$. Since $\boldsymbol{p}^* = (p_1^*, \ldots, p_n^*)$ is the maximizer in (2.22), by the first order optimality condition, we have

$$\left( \frac{\partial S}{\partial p}(x_1, t; p_1^*, a_1, b_1), \ldots, \frac{\partial S}{\partial p}(x_n, t; p_n^*, a_n, b_n) \right) \in \partial J^*(\boldsymbol{p}^*),$$

where $\partial J^*$ denotes the subdifferential operator of $J^*$. Therefore, we conclude that $\boldsymbol{\gamma}(0) \in \partial J^*(\boldsymbol{p}^*)$, which implies that $J(\boldsymbol{\gamma}(0)) - \langle \boldsymbol{p}^*, \boldsymbol{\gamma}(0) \rangle + J^*(\boldsymbol{p}^*) = 0$. In other words, by (2.46), we have

$$\int_0^t \left( \frac{\|\dot{\boldsymbol{\gamma}}(s)\|^2}{2} - K(\boldsymbol{\gamma}(s)) \right) ds + J(\boldsymbol{\gamma}(0)) = \sum_{i=1}^n S(x_i, t; p_i^*, a_i, b_i) - J^*(\boldsymbol{p}^*) = S(\boldsymbol{x}, t).$$

Moreover, by Lemma 2.3.11(a) with $M$ being the identity matrix (whose assumptions are proved in the proof of Proposition 2.1.3(b)), the value $S(\boldsymbol{x}, t)$ is the optimal value of the optimal control problem (2.20). Hence, the cost of $\boldsymbol{\gamma}$ equals the optimal cost $S(\boldsymbol{x}, t)$. By straightforward calculation, the function $\boldsymbol{\gamma}$ is Lipschitz continuous and satisfies $\boldsymbol{\gamma}(t) = \boldsymbol{x}$. Therefore, $\boldsymbol{\gamma}$ is an optimal trajectory, and the corresponding optimal value equals $S(\boldsymbol{x}, t)$. The optimal trajectory is unique since the problem (2.20) is a strictly convex problem. □

### 2.3.4 Proof of Proposition 2.1.5

Define $\tilde{S}\colon \mathbb{R}^n \times [0, +\infty) \to \mathbb{R} \cup \{+\infty\}$ by

$$\tilde{S}(\boldsymbol{y}, t) := S(P\boldsymbol{y} + \boldsymbol{u}_0, t) = \sup_{\boldsymbol{p} \in \mathbb{R}^n} \left\{ \sum_{i=1}^{n} S(y_i, t; p_i, a_i, b_i) - \tilde{J}^*(\boldsymbol{p}) \right\}, \tag{2.47}$$

for any $\boldsymbol{y} \in \mathbb{R}^n, t \geq 0$, where the second equality follows directly from the definition of $S$ in (2.28). By definition (2.29), the function $\tilde{J}$ is the composition of the convex function $J$ with an affine map, and hence, $\tilde{J}$ is also convex. By (2.47) and Proposition 2.1.3, $\tilde{S}$ is a continuously differentiable solution to the HJ PDE (2.21) with the convex initial data $\tilde{J}$. By straightforward calculation, we have

$$\nabla \tilde{S}(\boldsymbol{y}, t) = \left( P^T \nabla_{\boldsymbol{x}} S(P\boldsymbol{y} + \boldsymbol{u}_0, t), \frac{\partial S}{\partial t}(P\boldsymbol{y} + \boldsymbol{u}_0, t) \right). \tag{2.48}$$

Applying (2.48) and the change of variable $\boldsymbol{x} = P\boldsymbol{y} + \boldsymbol{u}_0$ to the HJ PDE (2.21), we conclude that $S$ is a continuously differentiable solution of the following PDE:

$$\begin{cases} \dfrac{\partial S}{\partial t}(\boldsymbol{x},t) + \dfrac{\|P^T\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)\|^2}{2} + \displaystyle\sum_{i=1}^{n} K_i((P^{-1}\boldsymbol{x} - P^{-1}\boldsymbol{u}_0)_i) = 0 & \boldsymbol{x}\in\mathbb{R}^n, t>0, \\ S(\boldsymbol{x},0) = J(\boldsymbol{x}) & \boldsymbol{x}\in\mathbb{R}^n. \end{cases} \tag{2.49}$$

Since we assume $M = PP^T$, after some computation, we get that

$$\begin{aligned} \frac{1}{2}\|P^T\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)\|^2 &= \frac{1}{2}\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)^T PP^T\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t) = \frac{1}{2}\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)^T M\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t) \\ &= \frac{1}{2}\|\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)\|_M^2. \end{aligned}$$

Then, to prove that $S$ solves (2.2), it suffices to prove that $\tilde{K} = K$, where $\tilde{K}\colon \mathbb{R}^n \to (-\infty, 0]$ is defined by

$$\tilde{K}(\boldsymbol{x}) := \sum_{i=1}^{n} K_i((P^{-1}\boldsymbol{x} - P^{-1}\boldsymbol{u}_0)_i) \quad \forall\, \boldsymbol{x}\in\mathbb{R}^n.$$

Recall that each function $K_i\colon \mathbb{R}\to(-\infty,0]$ is 1-homogeneous, and hence, the function $\boldsymbol{x}\mapsto \tilde{K}(\boldsymbol{x}+\boldsymbol{u}_0) = \sum_{i=1}^n K_i((P^{-1}\boldsymbol{x})_i) \in (-\infty,0]$ is also 1-homogeneous. Since any non-positive 1-homogeneous function is uniquely determined by its superlevel set at $-1$, to prove $\tilde{K} = K$, it suffices to prove

$$\{\boldsymbol{x}\in\mathbb{R}^n\colon \tilde{K}(\boldsymbol{x}+\boldsymbol{u}_0)\geq -1\} = \{\boldsymbol{x}\in\mathbb{R}^n\colon K(\boldsymbol{x}+\boldsymbol{u}_0)\geq -1\}. \tag{2.50}$$

Denote by $\Delta_n$ the simplex set, i.e. define $\Delta_n$ by

$$\Delta_n := \{(\alpha_1,\dots,\alpha_n)\in[0,1]^n\colon \sum_{i=1}^{n}\alpha_i = 1\}.$$

By straightforward calculation, we have

$$
\begin{aligned}
&\{\boldsymbol{x} \in \mathbb{R}^n \colon \tilde{K}(\boldsymbol{x} + \boldsymbol{u}_0) \geq -1\} \\
=&\left\{\boldsymbol{x} \in \mathbb{R}^n \colon \sum_{i=1}^{n} K_i((P^{-1}\boldsymbol{x})_i) \geq -1\right\} \\
=&\bigcup_{\boldsymbol{\alpha} \in \Delta_n} \{\boldsymbol{x} \in \mathbb{R}^n \colon K_i((P^{-1}\boldsymbol{x})_i) \geq -\alpha_i \, \forall i \in \{1, \ldots, n\}\} \\
=&\bigcup_{\boldsymbol{\alpha} \in \Delta_n} \left\{\boldsymbol{x} \in \mathbb{R}^n \colon (P^{-1}\boldsymbol{x})_i \in \left[-\frac{\alpha_i}{b_i}, \frac{\alpha_i}{a_i}\right] \, \forall i \in \{1, \ldots, n\}\right\} \\
=&\left\{\boldsymbol{x} \in \mathbb{R}^n \colon P^{-1}\boldsymbol{x} \in \operatorname{co}\left(\bigcup_{j=1}^{n} \left\{\frac{\boldsymbol{e}_j}{a_j}, -\frac{\boldsymbol{e}_j}{b_j}\right\}\right)\right\} \\
=&\operatorname{co}\left(\bigcup_{j=1}^{n} \left\{\frac{1}{a_j}\boldsymbol{u}_j, -\frac{1}{b_j}\boldsymbol{u}_j\right\}\right) \\
=&\{\boldsymbol{x} \in \mathbb{R}^n \colon K(\boldsymbol{x} + \boldsymbol{u}_0) \geq -1\},
\end{aligned}
$$

where $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n \in \mathbb{R}^n$ denote the standard basis vectors in $\mathbb{R}^n$. Therefore, (2.50) holds, and we obtain $\tilde{K} = K$. As a result, the HJ PDE (2.49) coincides with (2.2), and hence, the function $S$ defined in (2.28) is a continuously differentiable solution to the HJ PDE (2.2).

Now assume that $J$ satisfies (2.24). Then, after straightforward calculation, we obtain that for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$,

$$
\begin{aligned}
\left|\tilde{J}(\boldsymbol{x}) - \tilde{J}(\boldsymbol{y})\right| &= |J(P\boldsymbol{x} + \boldsymbol{u}_0) - J(P\boldsymbol{y} + \boldsymbol{u}_0)| \\
&\leq C\|P\boldsymbol{x} - P\boldsymbol{y}\|(1 + \|P\boldsymbol{x} + \boldsymbol{u}_0\|^\delta + \|P\boldsymbol{y} + \boldsymbol{u}_0\|^\delta) \\
&\leq C\|P\|\|\boldsymbol{x} - \boldsymbol{y}\| \left(1 + 2^\delta(\|P\|^\delta\|\boldsymbol{x}\|^\delta + \|\boldsymbol{u}_0\|^\delta) + 2^\delta(\|P\|^\delta\|\boldsymbol{y}\|^\delta + \|\boldsymbol{u}_0\|^\delta)\right) \\
&\leq C\|P\| \max\{1 + 2^{\delta+1}\|\boldsymbol{u}_0\|^\delta, 2^\delta\|P\|^\delta\}\|\boldsymbol{x} - \boldsymbol{y}\| \left(1 + \|\boldsymbol{x}\|^\delta + \|\boldsymbol{y}\|^\delta\right).
\end{aligned}
$$

Hence, $\tilde{J}$ is a convex function satisfying (2.24). By Proposition 2.1.3(b), the function $\tilde{S}$ defined in (2.47) is in the solution set $\mathcal{G}$. Since the function $S$ is the composition

of $\tilde{S}$ and an affine function, it is straightforward to check that the function $S$ is also in the solution set $\mathcal{G}$.

Now, we prove the uniqueness of the viscosity solution to the HJ PDE (2.2) in the solution set $\mathcal{G}$. Note that for any viscosity solution $W \in \mathcal{G}$ to the HJ PDE (2.2), the corresponding function $\tilde{W}\colon \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$ defined by

$$\tilde{W}(\boldsymbol{y}, t) := W(P\boldsymbol{y} + \boldsymbol{u}_0, t) \quad \forall \boldsymbol{y} \in \mathbb{R}^n, t \geq 0$$

is a viscosity solution to the HJ PDE (2.21) with initial condition $\tilde{J}$. By Proposition 2.1.3(b), the function $\tilde{W}$ is the unique viscosity solution in the solution set $\mathcal{G}$ to the HJ PDE (2.21) with the initial condition $\tilde{J}$. Then, the uniqueness of the viscosity solution $W$ follows since we have $W(\boldsymbol{x}, t) = \tilde{W}(P^{-1}(\boldsymbol{x} - \boldsymbol{u}_0), t)$ by definition of $\tilde{W}$. In other words, the function $S$ is the unique viscosity solution in the solution set $\mathcal{G}$ to the HJ PDE (2.2). □

### 2.3.5 Proof of Proposition 2.1.6

Let $\boldsymbol{x} \in \mathbb{R}^n$ and $t > 0$. First, we prove that the problem (2.1) is equivalent to another optimal control problem in the form of (2.20). For any trajectory $s \mapsto \boldsymbol{x}(s)$ satisfying the constraint in (2.1), define another trajectory $\boldsymbol{y}(s) := P^{-1}(\boldsymbol{x}(s) - \boldsymbol{u}_0)$. Note that the trajectory $s \mapsto \boldsymbol{y}(s)$ satisfies the constraint in the following optimal control problem:

$$\inf \left\{ \int_0^t \left( \frac{1}{2} \|\dot{\boldsymbol{y}}(s)\|^2 - \sum_{i=1}^n K_i(\boldsymbol{y}(s)) \right) ds + \tilde{J}(\boldsymbol{y}(0))\colon \boldsymbol{y}(t) = P^{-1}(\boldsymbol{x} - \boldsymbol{u}_0) \right\}. \quad (2.51)$$

Now, we prove that the cost of $\boldsymbol{x}(\cdot)$ in (2.1) equals the cost of $\boldsymbol{y}(\cdot)$ in (2.51). By straightforward calculation, the cost of $\boldsymbol{x}(\cdot)$ in the problem (2.1) equals

$$\begin{aligned}
&\int_0^t \left(\frac{1}{2}\|\dot{\boldsymbol{x}}(s)\|_{M^{-1}}^2 - K(\boldsymbol{x}(s))\right) ds + J(\boldsymbol{x}(0)) \\
&= \int_0^t \left(\frac{1}{2}\|P\dot{\boldsymbol{y}}(s)\|_{M^{-1}}^2 - K(P\boldsymbol{y}(s)+\boldsymbol{u}_0)\right) ds + J(P\boldsymbol{y}(0)+\boldsymbol{u}_0) \\
&= \int_0^t \left(\frac{1}{2}\|P\dot{\boldsymbol{y}}(s)\|_{M^{-1}}^2 - K(P\boldsymbol{y}(s)+\boldsymbol{u}_0))\right) ds + \tilde{J}(\boldsymbol{y}(0)),
\end{aligned} \tag{2.52}$$

where the last equality holds by definition of $\tilde{J}$ in (2.29). Since $M = PP^T$ holds and $P$ is invertible, we have

$$\begin{aligned}
\|P\dot{\boldsymbol{y}}(s)\|_{M^{-1}}^2 &= \dot{\boldsymbol{y}}(s)^T P^T M^{-1} P\dot{\boldsymbol{y}}(s) = \dot{\boldsymbol{y}}(s)^T P^T (P^T)^{-1} P^{-1} P\dot{\boldsymbol{y}}(s) \\
&= \dot{\boldsymbol{y}}(s)^T \dot{\boldsymbol{y}}(s) = \|\dot{\boldsymbol{y}}(s)\|^2.
\end{aligned} \tag{2.53}$$

By assumption, the function $\boldsymbol{y} \mapsto K(P\boldsymbol{y}+\boldsymbol{u}_0)$ is non-positive, concave, and 1-homogeneous and its superlevel set at the value $-1$ is calculated as follows:

$$\begin{aligned}
\{\boldsymbol{y} \in \mathbb{R}^n \colon K(P\boldsymbol{y}+\boldsymbol{u}_0) \geq -1\} &= P^{-1}\left(\operatorname{co}\left(\bigcup_{j=1}^n \left\{\frac{1}{a_j}\boldsymbol{u}_j, -\frac{1}{b_j}\boldsymbol{u}_j\right\}\right)\right) \\
&= \operatorname{co}\left(\bigcup_{j=1}^n \left\{\frac{1}{a_j}\boldsymbol{e}_j, -\frac{1}{b_j}\boldsymbol{e}_j\right\}\right),
\end{aligned}$$

where the first equality holds by (2.27) and the second equality follows from straightforward calculation (recall that $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n \in \mathbb{R}^n$ denote the standard basis vectors in $\mathbb{R}^n$). As a result, we have

$$K(P\boldsymbol{y}+\boldsymbol{u}_0) = \sum_{i=1}^n K_i(y_i) \quad \forall \boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n, \tag{2.54}$$

where each $K_i$ is the function defined in (2.4) with parameters $a = a_i$ and $b = b_i$.

Combining (2.52), (2.53), and (2.54), we conclude that the cost of $\boldsymbol{x}(\cdot)$ in (2.1) equals the cost of $\boldsymbol{y}(\cdot)$ in (2.51). Moreover, this relation between the trajectory $\boldsymbol{x}(\cdot)$ and the trajectory $\boldsymbol{y}(\cdot)$ is a bijection. Hence, the two problems (2.1) and (2.51) are equivalent. By Proposition 2.1.4 (whose assumptions are checked in the proof of Proposition 2.1.5), the unique optimal trajectory of the problem (2.51) is $s \mapsto \boldsymbol{y}(s) := \tilde{\boldsymbol{\gamma}}(s; P^{-1}(\boldsymbol{x} - \boldsymbol{u}_0), t)$ as defined in (2.33), and the optimal value of the problem (2.51) equals $S(\boldsymbol{x}, t)$ in (2.28). Therefore, the unique optimal trajectory of the problem (2.1) is the corresponding trajectory $\boldsymbol{x}(\cdot)$ given by $\boldsymbol{x}(s) = P\boldsymbol{y}(s) + \boldsymbol{u}_0 = P\tilde{\boldsymbol{\gamma}}(s; P^{-1}(\boldsymbol{x} - \boldsymbol{u}_0), t) + \boldsymbol{u}_0 = \boldsymbol{\gamma}(s; \boldsymbol{x}, t)$, and the optimal value of the problem (2.1) also equals $S(\boldsymbol{x}, t)$.

### 2.3.6 Proof of Proposition 2.1.7

We prove (b) first. Since each $J_j$ is a convex function satisfying (2.24), by Proposition 2.1.6, the value $S_{J_j}(\boldsymbol{x}, t)$ is the optimal value of the optimal control problem (2.1) with initial cost $J_j$. In other words, we have that

$$S_{J_j}(\boldsymbol{x}, t) = \inf \left\{ \int_0^t \left( \frac{1}{2} \|\dot{\boldsymbol{x}}(s)\|_{M^{-1}}^2 - K(\boldsymbol{x}(s)) \right) ds + J_j(\boldsymbol{x}(0)) \colon \boldsymbol{x}(t) = \boldsymbol{x} \right\},$$

for any $j \in \{1, \ldots, m\}$. After some calculations, we obtain

$$\begin{aligned} S(\boldsymbol{x}, t) &= \min_{j \in \{1, \ldots, m\}} S_{J_j}(\boldsymbol{x}, t) \\ &= \min_{j \in \{1, \ldots, m\}} \inf \left\{ \int_0^t \left( \frac{1}{2} \|\dot{\boldsymbol{x}}(s)\|_{M^{-1}}^2 - K(\boldsymbol{x}(s)) \right) ds + J_j(\boldsymbol{x}(0)) \colon \boldsymbol{x}(t) = \boldsymbol{x} \right\} \\ &= \inf \left\{ \int_0^t \left( \frac{1}{2} \|\dot{\boldsymbol{x}}(s)\|_{M^{-1}}^2 - K(\boldsymbol{x}(s)) \right) ds + \min_{j \in \{1, \ldots, m\}} J_j(\boldsymbol{x}(0)) \colon \boldsymbol{x}(t) = \boldsymbol{x} \right\} \\ &= \inf \left\{ \int_0^t \left( \frac{1}{2} \|\dot{\boldsymbol{x}}(s)\|_{M^{-1}}^2 - K(\boldsymbol{x}(s)) \right) ds + J(\boldsymbol{x}(0)) \colon \boldsymbol{x}(t) = \boldsymbol{x} \right\}. \end{aligned}$$

Therefore, $S(\boldsymbol{x}, t)$ is the optimal value of the problem (2.1) with initial cost $J$.

Now, we show that the trajectory $s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t)$ is an optimal trajectory. We abuse notation and use $\boldsymbol{\gamma}(s; \boldsymbol{x}, t)$ and $\boldsymbol{\gamma}(s)$ interchangeably whenever there is no ambiguity. Let $r$ be the index in (2.36). By Proposition 2.1.6 and (2.36), $\boldsymbol{\gamma}$ is the optimal trajectory of the problem (2.1) with initial cost $J_r$. As a result, its cost equals the optimal value $S_{J_r}(\boldsymbol{x}, t)$, which equals $S(\boldsymbol{x}, t)$ since $r$ is a minimizer in (2.34). Hence, we get

$$\begin{aligned} S(\boldsymbol{x}, t) &= \int_0^t \left( \frac{1}{2} \|\dot{\boldsymbol{\gamma}}(s)\|_{M^{-1}}^2 - K(\boldsymbol{\gamma}(s)) \right) ds + J_r(\boldsymbol{\gamma}(0)) \\ &\geq \int_0^t \left( \frac{1}{2} \|\dot{\boldsymbol{\gamma}}(s)\|_{M^{-1}}^2 - K(\boldsymbol{\gamma}(s)) \right) ds + \min_{j \in \{1, \dots, m\}} J_j(\boldsymbol{\gamma}(0)) \geq S(\boldsymbol{x}, t). \end{aligned}$$

Therefore, the inequalities above are equalities. In other words, the cost of $\boldsymbol{\gamma}$ in the optimal control problem (2.1) with initial cost $J$ equals the optimal value $S(\boldsymbol{x}, t)$, and hence, $\boldsymbol{\gamma}$ is an optimal trajectory of (2.1) with initial cost $J$.

It remains to prove (a). We will prove (a) by applying Lemma 2.3.11. We need to check the assumptions in Lemma 2.3.11. Note that each $J_j$ satisfies (2.24) for some constants $C_j, \delta_j \geq 0$. Choose $C := 3 \max_{j \in \{1, \dots, m\}} C_j \geq 0$ and $\delta := \max_{j \in \{1, \dots, m\}} \delta_j \geq 0$. For each $j \in \{1, \dots, m\}$, we have

$$\begin{aligned} |J_j(\boldsymbol{z}) - J_j(\boldsymbol{y})| &\leq C_j \|\boldsymbol{z} - \boldsymbol{y}\| (1 + \|\boldsymbol{z}\|^{\delta_j} + \|\boldsymbol{y}\|^{\delta_j}) \\ &\leq C_j \|\boldsymbol{z} - \boldsymbol{y}\| \left(1 + (1 + \|\boldsymbol{z}\|^{\delta}) + (1 + \|\boldsymbol{y}\|^{\delta})\right) \\ &\leq 3 C_j \|\boldsymbol{z} - \boldsymbol{y}\| (1 + \|\boldsymbol{z}\|^{\delta} + \|\boldsymbol{y}\|^{\delta}) \\ &\leq C \|\boldsymbol{z} - \boldsymbol{y}\| (1 + \|\boldsymbol{z}\|^{\delta} + \|\boldsymbol{y}\|^{\delta}) \qquad \forall \boldsymbol{z}, \boldsymbol{y} \in \mathbb{R}^n. \end{aligned}$$

For any $\boldsymbol{z}, \boldsymbol{y} \in \mathbb{R}^n$, letting $k \in \arg\min_{j\in\{1,\dots,m\}} J_j(\boldsymbol{y})$, there holds

$$J(\boldsymbol{z}) - J(\boldsymbol{y}) = J(\boldsymbol{z}) - J_k(\boldsymbol{y}) \leq J_k(\boldsymbol{z}) - J_k(\boldsymbol{y}) \leq C\|\boldsymbol{z}-\boldsymbol{y}\|(1+\|\boldsymbol{z}\|^\delta + \|\boldsymbol{y}\|^\delta).$$

Similarly, letting $k \in \arg\min_{j\in\{1,\dots,m\}} J_j(\boldsymbol{z})$, we have

$$J(\boldsymbol{z}) - J(\boldsymbol{y}) = J_k(\boldsymbol{z}) - J(\boldsymbol{y}) \geq J_k(\boldsymbol{z}) - J_k(\boldsymbol{y}) \geq -C\|\boldsymbol{z}-\boldsymbol{y}\|(1+\|\boldsymbol{z}\|^\delta + \|\boldsymbol{y}\|^\delta).$$

Therefore, the function $J$ also satisfies (2.24). Recall that by assumption, $J$ is continuous and bounded below by an affine function. Thus, the assumptions of Lemma 2.3.11 on $J$ hold. The assumptions of Lemma 2.3.11 on $K$ and $M$ also hold by straightforward reasoning. It remains to show that the assumptions on $S$ in Lemma 2.3.11(b) hold. Let $\boldsymbol{\alpha} \in \mathbb{R}^n$ be a vector such that $\boldsymbol{x} \mapsto J(\boldsymbol{x}) - \langle \boldsymbol{\alpha}, \boldsymbol{x}\rangle$ is bounded from below, and denote the lower bound by $\beta \in \mathbb{R}$. By (2.34), for each $j \in \{1,\dots,m\}$, there holds

$$J_j(\boldsymbol{x}) - \langle \boldsymbol{\alpha}, \boldsymbol{x}\rangle \geq J(\boldsymbol{x}) - \langle \boldsymbol{\alpha}, \boldsymbol{x}\rangle \geq \beta \quad \forall \boldsymbol{x} \in \mathbb{R}^n,$$

which implies that $\boldsymbol{\alpha}$ is in the domain of $J_j^*$. Recall that the function $S_{J_j}$ is defined by (2.28) with initial data $J_j$, and hence, $S_{J_j}$ satisfies (2.31) with initial condition $J_j$. By Lemma 2.3.3, we have

$$\begin{aligned}
S_{J_j}(\boldsymbol{x}, t) &\geq \sum_{i=1}^n S\left((P^{-1}\boldsymbol{x} - P^{-1}\boldsymbol{u}_0)_i, t; (P^T\boldsymbol{\alpha})_i, a_i, b_i\right) - J_j^*(\boldsymbol{\alpha}) + \langle \boldsymbol{\alpha}, \boldsymbol{u}_0\rangle \\
&\geq \sum_{i=1}^n \left((P^{-1}\boldsymbol{x} - P^{-1}\boldsymbol{u}_0)_i (P^T\boldsymbol{\alpha})_i + C_i\right) - J_j^*(\boldsymbol{\alpha}) + \langle \boldsymbol{\alpha}, \boldsymbol{u}_0\rangle \\
&= \langle P^{-1}\boldsymbol{x} - P^{-1}\boldsymbol{u}_0, P^T\boldsymbol{\alpha}\rangle + \sum_{i=1}^n C_i - J_j^*(\boldsymbol{\alpha}) + \langle \boldsymbol{\alpha}, \boldsymbol{u}_0\rangle \\
&= \langle \boldsymbol{\alpha}, \boldsymbol{x}\rangle + \sum_{i=1}^n C_i - J_j^*(\boldsymbol{\alpha}),
\end{aligned}$$

where each $C_i$ is the constant in the lower bound in Lemma 2.3.3 with constants $a = a_i$, $b = b_i$, and $p = (P^T\boldsymbol{\alpha})_i$. Then, by (2.35), there holds

$$\begin{aligned} S(\boldsymbol{x},t) &= \min_{j\in\{1,\dots,m\}} S_{J_j}(\boldsymbol{x},t) \geq \min_{j\in\{1,\dots,m\}} \left\{ \langle \boldsymbol{\alpha}, \boldsymbol{x}\rangle + \sum_{i=1}^{n} C_i - J_j^*(\boldsymbol{\alpha}) \right\} \\ &= \langle \boldsymbol{\alpha}, \boldsymbol{x}\rangle + \sum_{i=1}^{n} C_i - \max_{j\in\{1,\dots,m\}} J_j^*(\boldsymbol{\alpha}). \end{aligned}$$

Since $J_j^*(\boldsymbol{\alpha})$ is a finite number for each $j \in \{1,\dots,m\}$, the function $(\boldsymbol{x},t) \mapsto S(\boldsymbol{x},t) - \langle \boldsymbol{\alpha}, \boldsymbol{x}\rangle$ is bounded below by $\sum_{i=1}^{n} C_i - \max_{j\in\{1,\dots,m\}} J_j^*(\boldsymbol{\alpha}) \in \mathbb{R}$, and hence, the assumption on $S$ in Lemma 2.3.11(b) is satisfied. Therefore, all the assumptions in Lemma 2.3.11(a)-(b) are satisfied. Applying Lemma 2.3.11, we get that the value function in (2.1) is the unique viscosity solution to the HJ PDE (2.2) in the solution set $\mathcal{G}$. Moreover, by (b), which we proved earlier, the function $S$ is the value function in (2.1). Therefore, the function $S$ is the unique viscosity solution to the HJ PDE (2.2) in the solution set $\mathcal{G}$. □

### 2.3.7 Proof of Proposition 2.2.1

Let $\boldsymbol{v}^N$, $\boldsymbol{d}^N$, and $\boldsymbol{p}^N$ be the corresponding vectors in the algorithm at the $N$-th iteration. Define the function $F\colon \mathbb{R}^n \to \mathbb{R}$ by

$$F(\boldsymbol{p}) = \sum_{i=1}^{n} S(x_i, t; p_i, a_i, b_i) \quad \forall \boldsymbol{p} = (p_1,\dots,p_n) \in \mathbb{R}^n.$$

Then, the objective function in (2.22) equals $F - J^*$. Let $\boldsymbol{p}^* = (p_1^*,\dots,p_n^*)$ be the optimizer of the optimization problem in (2.22), which exists and is unique by

Lemma 2.3.9(a). By [15, Section 3.2], we have

$$\lim_{N\to\infty} \left(F(\boldsymbol{d}^N) - J^*(\boldsymbol{v}^N)\right) = F(\boldsymbol{p}^*) - J^*(\boldsymbol{p}^*) = S(\boldsymbol{x}, t).$$

According to [43, Theorem 2.2] whose assumptions are proved using [43, Remark 2.2], both $\boldsymbol{v}^N$ and $\boldsymbol{d}^N$ converge to the point $\boldsymbol{p}^*$ as $N$ approaches infinity, and hence, $\boldsymbol{p}^N$ in Algorithm 1 also converges to $\boldsymbol{p}^*$. By Lemma 2.3.1, the function $F$ is continuously differentiable, and hence, it is also Lipschitz in any compact domain. Let $L$ be its Lipschitz constant on a compact domain containing $\{\boldsymbol{d}^N\}$ and $\{\boldsymbol{v}^N\}$. Then, we have

$$\begin{aligned}
&\lim_{N\to\infty} |\hat{S}(\boldsymbol{x}, t) - S(\boldsymbol{x}, t)| \\
&\qquad \le \lim_{N\to\infty} \left|F(\boldsymbol{d}^N) - J^*(\boldsymbol{v}^N) - S(\boldsymbol{x}, t)\right| + \lim_{N\to\infty} \left|F(\boldsymbol{d}^N) - F(\boldsymbol{v}^N)\right| \\
&\qquad \le \lim_{N\to\infty} L \left\|\boldsymbol{d}^N - \boldsymbol{v}^N\right\| = 0.
\end{aligned}$$

Now, it remains to prove the second formula in (2.11). By definition of $\gamma$ in (2.9), (2.10), (2.11), (2.13), (2.14), and (2.16), the function $p \mapsto \gamma(s; x, t, p, a, b)$ is continuous. Since $\boldsymbol{p}^N$ converges to $\boldsymbol{p}^*$, $\hat{\boldsymbol{\gamma}}$ converges to $\boldsymbol{\gamma}$ pointwise for any $s \in [0, t]$. Moreover, by straightforward calculation, the function $s \mapsto \gamma(s; x, t, p, a, b)$ is continuously differentiable with respect to $s$, and its derivative is bounded by $|p| + (a+b)t$. Also, the function $s \mapsto \gamma(s; x, t, p, a, b)$ is bounded by $|x| + |p|t + (a+b)t^2$. Thus, the second formula in (2.11) holds by the Arzela-Ascoli theorem. □

## 2.4 Some computations for the numerical implementation

### 2.4.1 Proximal point of $p \mapsto -\frac{S(x,t;p,a,b)}{\lambda}$

In this section, we provide a numerical method for solving the following problem:

$$p^* = \arg\min_{p\in\mathbb{R}} \left\{ -S(x,t;p,a,b) + \frac{\lambda}{2}(p-c)^2 \right\}, \tag{2.1}$$

where $x, c \in \mathbb{R}$ and $t, a, b, \lambda > 0$ are some parameters and $S$ is the function defined in (2.6) and (2.15).

By Lemma 2.3.1, the objective function in (2.1) is strictly convex, 1-coercive, and continuously differentiable. Therefore, the minimizer exists and is unique. Moreover, $p^*$ is the minimizer if and only if the objective function has zero first-order derivative at $p^*$, i.e., if and only if $p^*$ satisfies

$$0 = \left.\frac{\partial(-S(x,t;p,a,b) + \frac{\lambda}{2}(p-c)^2)}{\partial p}\right|_{p=p^*} = \left.\left(-\frac{\partial S(x,t;p,a,b)}{\partial p} + \lambda(p-c)\right)\right|_{p=p^*}. \tag{2.2}$$

Recall that the function $V$ is defined piecewise. Thus, we compute the first-order derivative using (2.1) and solve (2.2) on each piece.

First, we consider the case where $p \geq 0$. If $(x,t,p) \in \Omega_1$, then we have

$$-\frac{\partial S(x,t;p,a,b)}{\partial p} + \lambda(p-c) = \frac{at^2}{2} + pt - x + \lambda(p-c),$$

which has the root

$$p_1(x,t,a,b,c) = \frac{-\frac{at^2}{2} + x + \lambda c}{t+\lambda}.$$

If $(x,t,p) \in \Omega_2$, then we have

$$-\frac{\partial S(x,t;p,a,b)}{\partial p} + \lambda(p-c) = -\frac{bt^2}{2} + pt - x + \lambda(p-c),$$

which has the root

$$p_2(x,t,a,b,c) = \frac{\frac{bt^2}{2} + x + \lambda c}{t+\lambda}.$$

If $(x,t,p) \in \Omega_3$, then we have

$$\begin{aligned} & -\frac{\partial S(x,t;p,a,b)}{\partial p} + \lambda(p-c) \\ = & -\frac{a+b}{(a+2b)^2}\left(-(bt-p)^2 + (p-bt)\sqrt{(bt-p)^2+2x(a+2b)}\right) \\ & -\frac{bx}{a+2b} - \frac{bt^2}{2} + pt + \lambda(p-c). \end{aligned} \tag{2.3}$$

We apply Newton's method to compute the root $p_3$ of (2.3). In Newton's method, the value is iteratively updated as

$$p_3^{k+1} = p_3^k - \frac{-\frac{\partial S(x,t;p,a,b)}{\partial p} + \lambda(p-c)}{-\frac{\partial^2 S(x,t;p,a,b)}{\partial p^2} + \lambda} = p_3^k - \frac{A_1}{A_2}, \tag{2.4}$$

where the numerator $A_1$ equals

$$\begin{aligned} A_1 = -\frac{a+b}{(a+2b)^2}\left(-(bt-p)^2 + (p-bt)\sqrt{(bt-p)^2+2x(a+2b)}\right) \\ -\frac{bx}{a+2b} - \frac{bt^2}{2} + pt + \lambda(p-c), \end{aligned}$$

and the denominator $A_2$ equals

$$A_2 = -\frac{2(a+b)}{(a+2b)^2}\left(bt - p + \frac{(bt-p)^2 + x(a+2b)}{\sqrt{(bt-p)^2+2x(a+2b)}}\right) + t + \lambda.$$

To make Newton's method more robust, we also enforce a lower bound $\underline{p}$ defined by

$$\underline{p} = \max\left\{\frac{x}{t} - \frac{at}{2}, bt - b\sqrt{\frac{2|x|}{a}}, 0\right\}$$

and an upper bound $\bar{p}$ defined by

$$\bar{p} = \max\{bt + |c| + 1, \underline{p}\}.$$

The lower bound $\underline{p}$ is given by the definition of $\Omega_3$. The upper bound $\bar{p}$ is set to be $\max\{bt+|c|+1, \underline{p}\}$ since the corresponding function value in (2.3) is positive and the function in (2.3) is increasing with respect to $p$ for $p \geq \underline{p}$, which implies that no root in $\Omega_3$ can be larger than $bt+|c|+1$. If the function value corresponding to $\underline{p}$ in (2.3) is also positive, then there is no root in $\Omega_3$, and we simply set $p_3$ to be $\underline{p}$. Here, the choice of initialization for Newton's method is not crucial. For convenience, in Sections 2.2.1 and 2.2.3, we initialize Newton's method with $p_3^0 = \underline{p} + 1$, and in Section 2.2.2, we initialize Newton's method with the value of $p_3$ found in the previous iteration of ADMM in Algorithm 1.

If $(x, t, p) \in \Omega_4 \cup \Omega_5$, we have

$$-\frac{\partial S(x, t; p, a, b)}{\partial p} + \lambda(p - c) = \frac{p^2}{2b} + \lambda(p - c),$$

whose largest root is given by

$$p_4(x, t, a, b, c) = -b\lambda + \sqrt{b^2\lambda^2 + 2b\lambda c},$$

which is non-negative if and only if $c \geq 0$. In other words, if $c$ is negative, $p_4$ is not a candidate for the minimizer.

Now, we consider the case of $p < 0$. By (2.15), we have

$$0 = -\frac{\partial S(x,t;p,a,b)}{\partial p} + \lambda(p-c) = \frac{\partial S(-x,t;q,b,a)}{\partial q} + \lambda(-q-c), \tag{2.5}$$

where we apply the change of variable $q = -p$. From (2.5), we observe that the vector $q$ is the positive solution of (2.2) where the parameters $(x,t,a,b,c)$ are replaced by $(-x,t,b,a,-c)$. Hence, the roots are in the set $\{p_i' \colon i = 1,2,3,4\}$, where each $p_i'$ is defined by

$$p_i'(x,t,a,b,c) = -p_i(-x,t,b,a,-c).$$

Therefore, we get several candidates for the minimizer $p^*$. We denote the set of candidates by $C$, which is defined by

$$C = \{p_i(x,t,a,b,c),\, p_i'(x,t,a,b,c) \colon i = 1,2,3,4\}.$$

Note that we can simplify the set $C$. By straightforward calculation, we obtain

$$p_1'(x,t,a,b,c) = p_2(x,t,a,b,c), \quad p_2'(x,t,a,b,c) = p_1(x,t,a,b,c).$$

Moreover, we have

$$p_4'(x,t,a,b,c) = a\lambda - \sqrt{a^2\lambda^2 - 2a\lambda c},$$

which is negative if and only if $c < 0$. In other words, if $c$ is non-negative, $p_4'$ is not a candidate for the minimizer. Define $\tilde{p}_4(x,t,a,b,c)$ by

$$\tilde{p}_4(x,t,a,b,c) = \begin{cases} p_4(x,t,a,b,c) & c \geq 0, \\ p_4'(x,t,a,b,c) & c < 0. \end{cases}$$

Then, $\tilde{p}_4(x,t,a,b,c)$ can replace $p_4(x,t,a,b,c)$ and $p_4'(x,t,a,b,c)$ as candidate mini-

mizers. Similarly, if $x$ is negative, then $(x, t, p)$ cannot be in the set $\Omega_3$, and hence $p_3$ is not a possible candidate. Meanwhile, if $x$ is non-negative, then $(-x, t, -p)$ cannot be in the set $\Omega_3(b, a)$ (which denotes the set $\Omega_3$ with parameters $b, a$, instead of $a, b$), and hence, $p_3'$ is not a possible candidate. Thus, we define $\tilde{p}_3(x, t, a, b, c)$ by

$$\tilde{p}_3(x, t, a, b, c) = \begin{cases} p_3(x, t, a, b, c) & x \geq 0, \\ p_3'(x, t, a, b, c) & x < 0, \end{cases}$$

and we use $\tilde{p}_3$ to replace $p_3$ and $p_3'$ as candidates. As a result, we simplify the set $C$ to

$$C = \{p_1(x, t, a, b, c),\ p_2(x, t, a, b, c),\ \tilde{p}_3(x, t, a, b, c),\ \tilde{p}_4(x, t, a, b, c)\}.$$

Finally, the minimizer $p^*$ is selected among the four candidates as follows:

$$p^* = \arg\min_{p \in C} \left\{ -S(x, t; p, a, b) + \frac{\lambda}{2}(p - c)^2 \right\}.$$

### 2.4.2 An equivalent expression for $S$

Let $S$ be the function defined in (2.6) and (2.15). In this section, we present an equivalent expression for $S$, which is used in our numerical implementation. When

$p \geq 0$, the function $S$ can be written as follows:

$$S(x,t;p,a,b) = \begin{cases} f_1 & (x,t,p) \in \Omega_1, \\ f_2 & (x,t,p) \in \Omega_2, \\ \max\{f_3, f_4\} & (x,t,p) \in \Omega_3 \cup \Omega_4, \\ f_5 & (x,t,p) \in \Omega_5, \end{cases}$$

$$= \begin{cases} f_1 & x \geq pt + \dfrac{at^2}{2}, \\ f_2 & \left(t < \dfrac{p}{b}, x < 0\right) \text{ or } \left(x < -\dfrac{b}{2}\left(t - \dfrac{p}{b}\right)^2, t \geq \dfrac{p}{b}\right), \\ \max\{f_3, f_4\} & 0 \leq x < pt + \dfrac{at^2}{2}, \\ f_5 & t - \dfrac{p}{b} \geq \sqrt{\dfrac{2|x|}{b}}, x < 0, \end{cases}$$

where $f_1, \ldots, f_5$ are the functions defined in (2.8), or equivalently in (2.6), below. Here, we use the notation $f_i$ instead of $f_i(x,t;p,a,b)$ for simplicity. By straightforward calculation, when $p < 0$, the formula reads:

$$S(x,t;p,a,b) = \begin{cases} f_2 & x \leq pt - \dfrac{bt^2}{2}, \\ f_1 & \left(t < -\dfrac{p}{a}, x > 0\right) \text{ or } \left(x > \dfrac{a}{2}\left(t + \dfrac{p}{a}\right)^2, t \geq -\dfrac{p}{a}\right), \\ \max\{f_3', f_4'\} & pt - \dfrac{bt^2}{2} < x \leq 0, \\ f_5' & t + \dfrac{p}{a} \geq \sqrt{\dfrac{2|x|}{a}}, x > 0, \end{cases}$$

where $f_i'$ denotes $f_i(-x,t;-p,b,a)$. The above equivalent expressions for $S$ are advantageous since they slightly reduce the amount of conditional branching required by our implementation as well as simplify the conditions that need to be checked. Thus, both of these differences help promote the performance of our implementation. Furthermore, the equivalent formulas for $f_i$ and $f_i'$ in our implementation are given

as follows:

$$
\begin{aligned}
f_1 &= \frac{p^3}{6a} - \frac{(at+p)^3}{6a} + x(at+p), \\
f_2 &= -\frac{p^3}{6b} - \frac{(bt-p)^3}{6b} - x(bt-p), \\
f_3 &= \frac{a+b}{3(a+2b)^2}\left((bt-p)^3 + \sqrt{\Delta}^3\right) - \frac{bx(bt-p)}{a+2b} - \frac{p^3}{6b} - \frac{(bt-p)^3}{6b}, \\
f_3' &= \frac{a+b}{3(2a+b)^2}\left((at+p)^3 + \sqrt{\Delta'}^3\right) + \frac{ax(at+p)}{2a+b} + \frac{p^3}{6a} - \frac{(at+p)^3}{6a}, \\
f_4 &= \frac{\sqrt{8a|x|^3}}{3} - \frac{p^3}{6b}, \\
f_4' &= \frac{\sqrt{8b|x|^3}}{3} + \frac{p^3}{6a}, \\
f_5 &= \frac{\sqrt{8b|x|^3}}{3} - \frac{p^3}{6b}, \\
f_5' &= \frac{\sqrt{8a|x|^3}}{3} + \frac{p^3}{6a},
\end{aligned}
\tag{2.6}
$$

where we define $\Delta = (bt-p)^2 + 2x(a+2b)$ and $\Delta' = (at+p)^2 - 2x(2a+b)$. These equivalent expressions are formulated in order to reduce extraneous arithmetic operations in the implementation as well as to avoid any potential complications with undefined square roots.

## 2.5 Summary

In this chapter, we presented analytical solutions to certain optimal control problems with quadratic running costs on the velocity and certain piecewise affine convex running costs on the trajectory. Moreover, we presented a Hopf-type representation formula for the corresponding HJ PDE with quadratic kinetic energy and piecewise affine non-positive concave potential function. We also presented efficient algorithms for solving these problems with convex initial costs and certain non-convex initial

costs. We demonstrated that our algorithms do not suffer from the curse of dimensionality and have promising speedup when implemented on FPGAs in comparison to CPUs. A possible future direction is to combine the proposed algorithms with other building blocks, such as the solver in [48] and/or the LQR solver, to handle more general optimal control problems. Additionally, it would be worthwhile to explore FPGA implementations of ADMM (or other proximal point-based methods), where we use our numerical solver for quadratic initial costs as a building block to compute the proximal point of $S$; exploring these implementations would give us a better understanding of how our solver can be combined with existing methods for real-time applications.

# Chapter Three

---

# Lax-Oleinik-type representation formulas

In this chapter, we focus on a class of optimal control problems with certain non-smooth control constraints and a running cost that is quadratic in the state variable. Constraints on the control are important for many practical applications, where the control often corresponds to some physical quantity, such as the velocity, acceleration, or jerk of a vehicle, which must be finitely bounded in practice. Typically, these control-constrained optimal control problems are not solved exactly. Instead, many algorithms [22, 21, 96, 17, 70, 110, 19, 5] use LQR as a building block to solve these control-constrained problems, even though there is no constraint on the control in the LQR problem. These algorithms then also employ other approximation and discretization methods, such as time discretizations, splitting methods, barrier methods, or spectral methods, to obtain an approximate solution. Hence, these algorithms do not directly provide the solution to control-constrained optimal control problems.

In this chapter, we present analytical solutions for a particular class of control-constrained optimal control problems and Lax-Oleinik-type representation formulas for the corresponding HJ PDEs, which solve these problems exactly without discretizations or approximations of the original problems. Note that this differs from numerical algorithms in the existing literature, which only approximate the solution. We then use these representation formulas to design efficient numerical solvers for these problems in high dimensions. Therefore, we provide both theoretical guarantees and efficient numerical methods for this class of problems. As a result, our numerical methods have the potential to complement LQR as a building block in numerical algorithms for solving control-constrained optimal control problems. We also present both low latency and high throughput FPGA implementations of our numerical solvers, which demonstrate the potential of our solvers for real-time optimal control applications. The content of this chapter is joint work with Jérôme

Darbon and Tingwei Meng [24]. My main contributions include the development of a prototype implementation of our numerical solvers in MATLAB (the results of which are not shown, but would look identical to the figures in Section 3.2), the conceptualization of the CPU implementation, and the creation of the FPGA implementations.

The organization of this chapter is as follows. In Section 3.1, we present the class of optimal control problems and HJ PDEs considered in this chapter as well as the analytical solutions of these problems. In Section 3.1.1, the one-dimensional problems are analyzed. In Section 3.1.2, we consider a class of separable high-dimensional problems. In Section 3.1.3, we provide a Lax-Oleinik-type representation formula for our most general high-dimensional case. In Section 3.2, we propose efficient numerical solvers for these problems and present some high-dimensional numerical results. More specifically, in Section 3.2.1, we present an efficient exact solver for quadratic initial costs, which will be used later as a building block for more general initial costs. In Section 3.2.2, we present an ADMM algorithm using the building block from the quadratic case to solve certain optimal control problems with more general convex initial costs. In Section 3.2.3, we generalize our proposed methods to certain nonconvex initial costs using min-plus techniques. In each of these subsections, we present corresponding high-dimensional numerical results using both a CPU and an FPGA implementation. Some proofs and technical lemmas are provided in Section 3.3, some computations for the numerical implementation are provided in Section 3.4, and some convergence results are provided in Section 3.5. Finally, in Section 3.6, we make some concluding remarks and list some possible future directions.

## 3.1 Analytical solutions

In this section, we provide the analytical solutions to certain optimal control problems, where the running cost depends on the state variable. We also provide a Lax-Oleinik-type representation formula for the corresponding HJ PDEs, where the Hamiltonian depends on the state variable.

Specifically, we consider the following problem. Let $\boldsymbol{v}_0$ be a vector in $\mathbb{R}^n$, $\{a_i\}_{i=1}^n$ and $\{b_i\}_{i=1}^n$ be positive scalars, $P$ be an invertible matrix with $n$ rows and $n$ columns, and $M := PP^T$ be a symmetric positive definite matrix which defines a norm $\|\boldsymbol{x}\|_M := \sqrt{\langle \boldsymbol{x}, M\boldsymbol{x}\rangle}$. We use $\langle\cdot,\cdot\rangle$ to denote the Euclidean inner product in $\mathbb{R}^n$ whose associated $\ell^2$-norm is denoted by $\|\cdot\|$. In this chapter, we use bold characters to denote high-dimensional vectors in $\mathbb{R}^n$, and we use $x_i$ to denote the $i$-th component of a high-dimensional vector $\boldsymbol{x} \in \mathbb{R}^n$. Our goal is to solve the following optimal control problem:

$$S(\boldsymbol{x},t) = \min\left\{ \int_0^t \frac{1}{2}\|\boldsymbol{x}(s)-\boldsymbol{v}_0\|_M^2 ds + J(\boldsymbol{x}(0))\colon \boldsymbol{x}(t)=\boldsymbol{x}, \right. \\ \left. P^T\dot{\boldsymbol{x}}(s) \in \prod_{i=1}^n [-b_i, a_i] \ \forall s \in (0,t) \right\}, \tag{3.1}$$

where $\boldsymbol{v}_0$, $\{a_i\}_{i=1}^n$, $\{b_i\}_{i=1}^n$, and $P$ satisfy the above assumptions, the time horizon is $t>0$, the terminal position is $\boldsymbol{x}\in\mathbb{R}^n$, and the initial cost is a lower semi-continuous function $J\colon \mathbb{R}^n \to \mathbb{R}$. In the optimal control problem, the function $\boldsymbol{x}(\cdot)\colon [0,t]\to\mathbb{R}^n$ is assumed to be a Lipschitz function, and $\dot{\boldsymbol{x}}(s)$ denotes its derivative at time $s$, which exists at $s\in(0,t)$ almost everywhere. Any trajectory $s\mapsto \boldsymbol{x}(s)$ is called feasible if it satisfies the constraints in the problem (3.1). To avoid the ambiguity of a trajectory and a vector, we use $\boldsymbol{x}(\cdot)$ to denote the trajectory, which is a function

of the time variable, and we use $\boldsymbol{x}$ to denote the vector.

In the literature, it is well-known that optimal control problems are highly related to HJ PDEs. Specifically, the optimal values in the optimal control problems are equal to the viscosity solutions to the corresponding HJ PDEs, and the optimal controls are related to the spatial gradient of the viscosity solutions to the HJ PDEs (see [8]). In our case, the optimal control problem (3.1) corresponds to the following HJ PDE:

$$\begin{cases} \dfrac{\partial S}{\partial t}(\boldsymbol{x},t) + K(\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)) - \dfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{v}_0\|_M^2 = 0 & \boldsymbol{x}\in\mathbb{R}^n, t\in(0,+\infty), \\ S(\boldsymbol{x},0) = J(\boldsymbol{x}) & \boldsymbol{x}\in\mathbb{R}^n, \end{cases} \tag{3.2}$$

where $K\colon \mathbb{R}^n \to [0,+\infty)$ is a piecewise affine, positively 1-homogeneous convex function related to the parameters $\{a_i\}_{i=1}^n$, $\{b_i\}_{i=1}^n$, and $P$ in the optimal control problem (3.1).

In Section 3.1.1, we provide the analytical solution for the one-dimensional case. In Section 3.1.2, we solve a simple separable high-dimensional case where $\boldsymbol{v}_0$ is the zero vector and $P$ is the identity matrix. In Section 3.1.3, we present the analytical solution to the high-dimensional optimal control problem (3.1) and the HJ PDE (3.2).

### 3.1.1 One-dimensional case

In this section, we consider the one-dimensional optimal control problem, which reads:

$$S(x,t) = \min\left\{ \int_0^t \frac{x(s)^2}{2} ds + J(x(0))\colon \dot{x}(s)\in[-b,a]\ \forall s\in(0,t),\ x(t)=x \right\}, \tag{3.3}$$

where $x \in \mathbb{R}$ and $t > 0$ denote the terminal position and time horizon, respectively, and $a, b > 0$ are positive scalars which give the restrictions on the velocity $\dot{x}(s)$. The corresponding HJ PDE reads:

$$\begin{cases} \dfrac{\partial S}{\partial t}(x,t) + K(\nabla_x S(x,t)) - \dfrac{x^2}{2} = 0 & x \in \mathbb{R}, t \in (0,+\infty), \\ S(x,0) = J(x) & x \in \mathbb{R}, \end{cases} \tag{3.4}$$

where $K \colon \mathbb{R} \to [0,+\infty)$ is the positively 1-homogeneous convex function defined by:

$$K(x) = \begin{cases} ax & x \geq 0, \\ -bx & x < 0, \end{cases} \tag{3.5}$$

where $a$ and $b$ are the positive parameters in (3.3).

In what follows, we will present the analytical solutions to the optimal control problem (3.3) and the HJ PDE (3.4). First, we start with the case when $J(x) = I_{\{u\}}(x)$ for some $u \in \mathbb{R}$, where $I_{\{u\}}$ denotes the indicator function of the set $\{u\}$. Recall that the indicator function $I_C \colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ of a set $C \subseteq \mathbb{R}^n$ is defined by:

$$I_C(x) := \begin{cases} 0 & x \in C, \\ +\infty & x \notin C. \end{cases}$$

Under this initial cost, the initial position in the optimal control problem (3.3) is fixed to be the point $u$. Thus, the optimal control problem (3.3) becomes

$$\min \left\{ \int_0^t \frac{x(s)^2}{2} ds \colon \dot{x}(s) \in [-b, a] \ \forall s \in (0,t),\ x(0) = u,\ x(t) = x \right\}. \tag{3.6}$$

We denote by $S(x,t;u,a,b)$ the optimal value in the minimization problem (3.6), which is a function of $(x,t) \in \mathbb{R} \times [0,+\infty)$ with parameters $u \in \mathbb{R}$, $a > 0$, and $b > 0$. The variables $x$ and $t$ are the terminal position and time horizon, respectively,

the parameter $u$ denotes the parameter in the initial cost $I_{\{u\}}$, and the parameters $a$ and $b$ are the positive parameters in (3.6). The optimal trajectory is denoted by $s \mapsto \gamma(s; x, t, u, a, b)$. In the notation of the optimal trajectory, there are five parameters: $x$, $t$, $u$, $a$, and $b$, which have the same meaning as the corresponding variables or parameters in the notation of the optimal value $S(x, t; u, a, b)$.

Now, we define the functions $S$ and $\gamma$. In Lemma 3.1.1, we will prove that the following definitions are indeed the optimal value and optimal trajectory in the corresponding optimal control problem (3.6). When $u \geq 0$, we define the function $\mathbb{R} \times [0, +\infty) \ni (x, t) \mapsto S(x, t; u, a, b) \in \mathbb{R} \cup \{+\infty\}$ by

$$S(x,t;u,a,b) := \begin{cases} \dfrac{u^3}{6b} + \dfrac{x^3}{6a} - \left(\dfrac{1}{6a} + \dfrac{1}{6b}\right)\left(\dfrac{au + bx - abt}{a+b}\right)^3 & \text{if } (x,t,u) \in \Omega_1, \\ \dfrac{u^3}{6b} + \dfrac{x^3}{6a} & \text{if } (x,t,u) \in \Omega_2, \\ \dfrac{u^3}{6b} - \dfrac{x^3}{6b} & \text{if } (x,t,u) \in \Omega_3, \\ +\infty & \text{otherwise,} \end{cases} \tag{3.7}$$

where the three sets $\Omega_i \subset \mathbb{R} \times [0, +\infty) \times [0, +\infty)$ with $i = 1, 2, 3$ are defined by:

$$\begin{aligned} \Omega_1 &:= \left\{(x,t,u) \colon 0 \leq t < \frac{u}{b},\ u - bt \leq x \leq u + at\right\} \\ &\qquad \bigcup \left\{(x,t,u) \colon t \geq \frac{u}{b},\ at - \frac{au}{b} \leq x \leq u + at\right\}, \\ \Omega_2 &:= \left\{(x,t,u) \in \mathbb{R} \times [0,+\infty) \times [0,+\infty) \colon t \geq \frac{u}{b},\ 0 \leq x < at - \frac{au}{b}\right\}, \\ \Omega_3 &:= \left\{(x,t,u) \in \mathbb{R} \times [0,+\infty) \times [0,+\infty) \colon t \geq \frac{u}{b},\ u - bt \leq x < 0\right\}. \end{aligned} \tag{3.8}$$

An illustration of a two-dimensional slice of these three sets for a fixed $u > 0$ is shown in Figure 3.1. After some computation, we conclude that the domain of the

function $(x,t) \mapsto S(x,t;u,a,b)$ is

$$\text{dom } ((x,t) \mapsto S(x,t;u,a,b)) = \{(x,t) \in \mathbb{R}^n \times [0,+\infty) \colon u - bt \le x \le u + at\}\,. \tag{3.9}$$

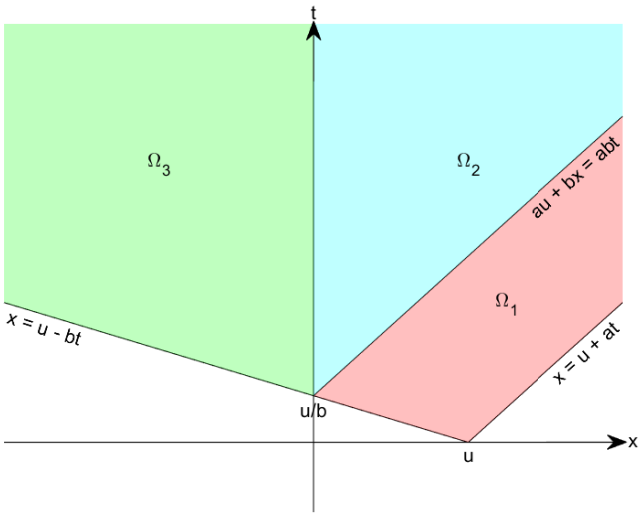



Figure 3.1: An illustration of a two-dimensional slice of the three sets $\Omega_1, \Omega_2, \Omega_3$ on the $xt$-plane.

To define the trajectory $s \mapsto \gamma(s;x,t,u,a,b)$, we consider three cases, which correspond to the first three lines in (3.7), respectively. We define the trajectory $[0,t] \ni s \mapsto \gamma(s;x,t,u,a,b) \in \mathbb{R}$ in these three cases as follows:

1. When $(x,t,u) \in \Omega_1$ holds, which corresponds to the first line in (3.7), we define $\gamma(s;x,t,u,a,b)$ by:

$$\gamma(s;x,t,u,a,b) := \begin{cases} u - bs & s \in \left[0, \dfrac{-x+u+at}{a+b}\right), \\ a(s-t) + x & s \in \left[\dfrac{-x+u+at}{a+b}, t\right]. \end{cases} \tag{3.10}$$

   The trajectory $\gamma(s;x,t,u,a,b)$ is non-negative for all $s \in [0,t]$. The velocity $\frac{d}{ds}\gamma(s;x,t,u,a,b)$ is $-b$ in the first line and $a$ in the second line. In other words,

the controlled object goes left in the first time period, and it goes right in the second time period.

2. When $(x,t,u) \in \Omega_2$ holds, which corresponds to the second line in (3.7), we define $\gamma(s;x,t,u,a,b)$ by:

$$\gamma(s;x,t,u,a,b) := \begin{cases} u - bs & s \in \left[0, \dfrac{u}{b}\right), \\ 0 & s \in \left[\dfrac{u}{b}, t - \dfrac{x}{a}\right), \\ a\,(s-t) + x & s \in \left[t - \dfrac{x}{a}, t\right]. \end{cases} \tag{3.11}$$

The trajectory $\gamma(s;x,t,u,a,b)$ is non-negative for all $s \in [0,t]$. The velocity $\frac{d}{ds}\gamma(s;x,t,u,a,b)$ is $-b$ in the first line, 0 in the second line, and $a$ in the third line. In other words, the controlled object first goes to the left, then stays at zero, and then goes to the right.

3. When $(x,t,u) \in \Omega_3$ holds, which corresponds to the third line in (3.7), we define $\gamma(s;x,t,u,a,b)$ by:

$$\gamma(s;x,t,u,a,b) := \begin{cases} u - bs & s \in \left[0, \dfrac{u}{b}\right), \\ 0 & s \in \left[\dfrac{u}{b}, t - \dfrac{|x|}{b}\right), \\ -b\,(s-t) + x & s \in \left[t - \dfrac{|x|}{b}, t\right]. \end{cases} \tag{3.12}$$

In the first line, the trajectory is positive, and the velocity is $-b$. In the second line, the trajectory and the velocity are both zero. In the third line, the trajectory is negative, and the velocity is $-b$. In other words, the controlled object first goes to the left, then stays at zero, and then goes to the left again.

So far, we have presented the representation formulas for the optimal values and the optimal trajectories of the one-dimensional optimal control problem (3.6), where

the initial position $u$ is non-negative. Note that the domain of the value function $S$ is given in (3.9), and the optimal trajectory $\gamma$ is well-defined if and only if $(x,t)$ is in the domain of $S$.

Now, we consider the case when $u$ is negative. In this case, we define the optimal values and the optimal trajectories by symmetry. To be specific, for any $u<0$, the function $\mathbb{R}\times[0,+\infty)\ni(x,t)\mapsto S(x,t;u,a,b)\in\mathbb{R}\cup\{+\infty\}$ is defined by:

$$S(x,t;u,a,b):=S(-x,t;-u,b,a)\quad\forall x\in\mathbb{R},\ t\geq 0, \tag{3.13}$$

where the right-hand side is defined in (3.7). After some computation, we conclude that the domain of the function $(x,t)\mapsto S(x,t;u,a,b)$ also satisfies (3.9) in this case. Similarly, for any $x\in\mathbb{R}$, $t\geq 0$, and $u<0$, the optimal trajectory $[0,t]\ni s\mapsto\gamma(s;x,t,u,a,b)\in\mathbb{R}$ is defined by:

$$\gamma(s;x,t,u,a,b):=-\gamma(s;-x,t,-u,b,a)\quad\forall\, s\in[0,t], \tag{3.14}$$

where the right-hand side is defined in (3.10), (3.11), and (3.12) for different cases. Note that $\gamma(s;-x,t,-u,b,a)$ on the right-hand side of (3.14) is well-defined if and only if the corresponding optimal value $S(-x,t;-u,b,a)$ is finite. Hence, $\gamma(s;x,t,u,a,b)$ on the left-hand side of (3.14) is well-defined if and only if $(x,t)$ is in the domain of the function $(x,t)\mapsto S(x,t;u,a,b)$, which equals the set in (3.9).

In the following lemma, we prove that the function $S$ and the trajectory $\gamma$, whenever they are well-defined, are indeed the unique value function and optimal trajectory for the optimal control problem (3.6).

**Lemma 3.1.1.** *Let $a,b,t$ be positive scalars and $x,u$ be real numbers satisfying $u-bt\leq x\leq u+at$. Define the function $[0,t]\ni s\mapsto\gamma(s;x,t,u,a,b)\in\mathbb{R}$*

*by* (3.10), (3.11), (3.12), *and* (3.14) *for different cases. Then,* $s \mapsto \gamma(s; x, t, u, a, b)$ *is the unique optimal trajectory of the optimal control problem* (3.6)*. Moreover, the optimal value equals* $S(x, t; u, a, b)$*, as defined in* (3.7) *and* (3.13)*.*

*Proof.* The proof is provided in Section 3.3.2. □

Now, we consider a general lower semi-continuous initial cost $J\colon \mathbb{R} \to \mathbb{R}$. We define the function $\mathbb{R} \times [0, +\infty) \ni (x, t) \mapsto S(x, t) \in \mathbb{R}$ as follows:

$$S(x,t) := \inf_{u \in [x-at, x+bt]} \{S(x, t; u, a, b) + J(u)\} \quad \forall x \in \mathbb{R}, t \geq 0, \tag{3.15}$$

where the term $S(x, t; u, a, b)$ is defined in (3.7) and (3.13). This is a Lax-Oleinik-type representation formula. In Propositions 3.1.2 and 3.1.3, we show that $S(x, t)$ is the optimal value in the optimal control problem (3.3) and that the function $S$ is the viscosity solution to the HJ PDE (3.4).

Let $u^*$ be a minimizer of the minimization problem in (3.15). Note that the minimizer $u^*$ exists since the objective function in the minimization problem in (3.15) is a lower semi-continuous function with compact domain $[x - at, x + bt]$ (see [119, Theorem 1.9]). However, the minimizer may be not unique. When there are multiple minimizers, let $u^*$ be one such minimizer. We define the function $\gamma(s; x, t)$ using $u^*$ as follows:

$$\gamma(s; x, t) := \gamma(s; x, t, u^*, a, b) \quad \forall s \in [0, t], \tag{3.16}$$

where the term $\gamma(s; x, t, u^*, a, b)$ on the right-hand side is defined in (3.10), (3.11), (3.12), and (3.14) for different cases. Note that the minimizer $u^*$ satisfies $u^* \in [x - at, x + bt]$. In other words, $(x, t)$ is in the domain in (3.9) with $u = u^*$, and hence

the right-hand side in (3.16) is well-defined. We prove in Proposition 3.1.2 that this trajectory $s \mapsto \gamma(s; x, t)$ is indeed an optimal trajectory of the problem (3.3). When there are multiple minimizers in (3.15), by Proposition 3.1.2, for any minimizer $u^*$, the corresponding function $s \mapsto \gamma(s; x, t)$ defined in (3.16) with $u^*$ is one optimal trajectory.

Now, we present the main results for the one-dimensional problems (3.3) and (3.4).

**Proposition 3.1.2.** *Let $J\colon \mathbb{R} \to \mathbb{R}$ be a lower semi-continuous function and $a, b$ be some positive scalars. Then, for all $x \in \mathbb{R}$ and $t > 0$, the function $[0, t] \ni s \mapsto \gamma(s; x, t) \in \mathbb{R}$ defined in (3.16) is an optimal trajectory for the optimal control problem (3.3), whose optimal value equals $S(x, t)$ as defined in (3.15). Moreover, if $J$ is convex, then the trajectory $s \mapsto \gamma(s; x, t)$ is the unique optimal trajectory.*

*Proof.* This is a corollary of Proposition 3.1.4. □

**Proposition 3.1.3.** *Let $J\colon \mathbb{R} \to \mathbb{R}$ be a continuous function and $K$ be the function defined in (3.5) with parameters $a, b > 0$. Let $S$ be the function defined in (3.15). Then, the function $S$ is the unique viscosity solution to the HJ PDE (3.4) in the solution set $C(\mathbb{R} \times [0, +\infty))$.*

*Proof.* This is a corollary of Proposition 3.1.7. □

### 3.1.2 Separable high-dimensional case

In this section, we consider the following high-dimensional optimal control problem with separable running cost:

$$S(\boldsymbol{x},t) = \min\left\{ \int_0^t \frac{\|\boldsymbol{x}(s)\|^2}{2} ds + J(\boldsymbol{x}(0)) \colon \boldsymbol{x}(t) = \boldsymbol{x}, \right. \\ \left. \dot{\boldsymbol{x}}(s) \in \prod_{i=1}^n [-b_i, a_i] \ \forall s \in (0,t) \right\}, \tag{3.17}$$

where $\boldsymbol{x}$ and $t$ are the terminal position and time horizon, respectively, $\{a_i\}$ and $\{b_i\}$ are positive scalars which provide the restrictions on the velocity $\dot{\boldsymbol{x}}$, and the initial cost $J \colon \mathbb{R}^n \to \mathbb{R}$ is a lower semi-continuous function. Note that we call the running cost separable since it is the sum of $n$ functions, the $j$-th of which only depends on the $j$-th component of $\boldsymbol{x}(s)$. The constraint on the control is also separable since it can be written as $n$ constraints, the $j$-th of which only depends on the derivative of the $j$-th component of the trajectory $\boldsymbol{x}(\cdot)$. The corresponding HJ PDE reads:

$$\begin{cases} \dfrac{\partial S}{\partial t}(\boldsymbol{x},t) + \displaystyle\sum_{i=1}^n K_i\left(\frac{\partial S(\boldsymbol{x},t)}{\partial x_i}\right) - \frac{1}{2}\|\boldsymbol{x}\|^2 = 0 & \boldsymbol{x} \in \mathbb{R}^n, t \in (0,+\infty), \\ S(\boldsymbol{x},0) = J(\boldsymbol{x}) & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{3.18}$$

where each function $K_i \colon \mathbb{R} \to \mathbb{R}$ is the positively 1-homogeneous convex function defined in (3.5) with the positive constants $a = a_i$ and $b = b_i$ from (3.17) and the initial condition is given by the initial cost $J$ in (3.17).

For this problem, we denote the optimal value by $S(\boldsymbol{x},t)$ and the optimal trajectory by $s \mapsto \boldsymbol{\gamma}(s;\boldsymbol{x},t)$, where $\boldsymbol{x}, t$ are parameters denoting the terminal position and the time horizon, respectively. Define the function $S \colon \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ by the following

Lax-Oleinik-type representation formula:

$$S(\boldsymbol{x},t) := \inf_{\boldsymbol{u}\in\prod_{i=1}^n [x_i-a_it,x_i+b_it]} \left\{ \sum_{i=1}^n S(x_i,t;u_i,a_i,b_i) + J(\boldsymbol{u}) \right\}, \tag{3.19}$$

for all $\boldsymbol{x} \in \mathbb{R}^n, t \geq 0$ and where each function $(x_i,t) \mapsto S(x_i,t;u_i,a_i,b_i)$ on the right-hand side is the function defined by (3.7) and (3.13). Let $\boldsymbol{u}^* = (u_1^*,\ldots,u_n^*) \in \mathbb{R}^n$ be a minimizer in (3.19). Note that the minimizer $\boldsymbol{u}^*$ exists since the objective function in the minimization problem in (3.19) is a lower semi-continuous function with compact domain $\prod_{i=1}^n [x_i - a_it, x_i + b_it]$ (see [119, Theorem 1.9]). However, the minimizer may be not unique. When there are multiple minimizers, let $\boldsymbol{u}^*$ be one such minimizer. Define the trajectory $s \mapsto \boldsymbol{\gamma}(s;\boldsymbol{x},t)$ using $\boldsymbol{u}^*$ as

$$\boldsymbol{\gamma}(s;\boldsymbol{x},t) := (\gamma(s;x_1,t,u_1^*,a_1,b_1),\ldots,\gamma(s;x_n,t,u_n^*,a_n,b_n)) \quad \forall s \in [0,t], \tag{3.20}$$

where the function $\gamma$ in the $i$-th component $\gamma(s;x_i,t,u_i^*,a_i,b_i)$ on the right-hand side is defined by (3.10), (3.11), (3.12), and (3.14) for different cases. Note that the $i$-th component of the minimizer $\boldsymbol{u}^*$ satisfies $u_i^* \in [x_i - a_it, x_i + b_it]$. In other words, $(x_i,t)$ is in the domain (3.9) with $u = u_i^*$, and hence, the $i$-th component on the right-hand side of (3.20) is well-defined for each $i \in \{1,\ldots,n\}$.

Next, we present two propositions. In Proposition 3.1.4, we show that the trajectory $s \mapsto \boldsymbol{\gamma}(s;\boldsymbol{x},t)$ is an optimal trajectory of the problem (3.17), where the optimal value equals $S(\boldsymbol{x},t)$. By Proposition 3.1.4, if there are multiple minimizers in (3.19), for any minimizer $\boldsymbol{u}^*$, the corresponding function $s \mapsto \boldsymbol{\gamma}(s;\boldsymbol{x},t)$ defined in (3.20) with $\boldsymbol{u}^*$ is an optimal trajectory. In Proposition 3.1.5, we show that $S$ is the viscosity solution to the high-dimensional HJ PDE (3.18).

**Proposition 3.1.4.** *Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a lower semi-continuous function. Let $\boldsymbol{a} = (a_1,\ldots,a_n)$ and $\boldsymbol{b} = (b_1,\ldots,b_n)$ be two vectors in $(0,+\infty)^n$. For any $\boldsymbol{x} \in \mathbb{R}^n$ and $t >$*

0, *the function* $[0,t] \ni s \mapsto \boldsymbol{\gamma}(s;\boldsymbol{x},t) \in \mathbb{R}$ *defined in* (3.20) *is an optimal trajectory of the optimal control problem* (3.17)*, where the optimal value equals* $S(\boldsymbol{x},t)$*, as defined in* (3.19)*. Moreover, if* $J$ *is convex, then the trajectory* $s \mapsto \gamma(s;\boldsymbol{x},t)$ *is the unique optimal trajectory.*

*Proof.* The proof is provided in Section 3.3.3. □

**Proposition 3.1.5.** *Let* $J\colon \mathbb{R}^n \to \mathbb{R}$ *be a continuous function. Let* $\boldsymbol{a} = (a_1,\dots,a_n)$ *and* $\boldsymbol{b} = (b_1,\dots,b_n)$ *be two vectors in* $(0,+\infty)^n$ *and* $K_i\colon \mathbb{R} \to [0,+\infty)$ *be defined by* (3.5) *with constants* $a = a_i$ *and* $b = b_i$ *for each* $i \in \{1,\dots,n\}$*. Then, the function* $S$ *defined in* (3.19) *is the unique viscosity solution to the HJ PDE* (3.18) *in the solution set* $C(\mathbb{R}^n \times [0,+\infty))$*.*

*Proof.* This is a corollary of Proposition 3.1.7. □

### 3.1.3 General high-dimensional case

In this section, we provide the analytical solutions to the high-dimensional optimal control problem (3.1) and the corresponding HJ PDE (3.2). Denote the optimal value by $S(\boldsymbol{x},t)$ and the optimal trajectory by $\boldsymbol{\gamma}(s;\boldsymbol{x},t)$. We define the function $S\colon \mathbb{R}^n \times [0,+\infty) \to \mathbb{R}$ by the following Lax-Oleinik-type representation formula:

$$S(\boldsymbol{x},t) := \inf_{\boldsymbol{u}\in\prod_{i=1}^n [y_i - a_i t, y_i + b_i t]} \left\{ \sum_{i=1}^n S(y_i,t;u_i,a_i,b_i) + J\left((P^T)^{-1}\boldsymbol{u} + \boldsymbol{v}_0\right) \right\}, \quad (3.21)$$

for all $\boldsymbol{x} \in \mathbb{R}^n, t \geq 0$ and where the vector $\boldsymbol{y} \in \mathbb{R}^n$ is defined by:

$$\boldsymbol{y} = (y_1, \ldots, y_n) := P^T \boldsymbol{x} - P^T \boldsymbol{v}_0. \tag{3.22}$$

In (3.21), each function $(y_i, t) \mapsto S(y_i, t; u_i, a_i, b_i)$ on the right-hand side is the function defined in (3.7) and (3.13), and $P$, $\boldsymbol{v}_0$, $\{a_i\}_{i=1}^n$, $\{b_i\}_{i=1}^n$ are the parameters in (3.1).

Let $\boldsymbol{u}^* = (u_1^*, \ldots, u_n^*)$ be a minimizer in the minimization problem (3.21). With a similar argument as in Section 3.1.2, we conclude that the minimizer $\boldsymbol{u}^*$ exists but may be not unique. If it is not unique, let $\boldsymbol{u}^*$ be one such minimizer. Define the trajectory $[0, t] \ni s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t) \in \mathbb{R}^n$ by

$$\boldsymbol{\gamma}(s; \boldsymbol{x}, t) := (P^T)^{-1} \left(\gamma(s; y_1, t, u_1^*, a_1, b_1), \ldots, \gamma(s; y_n, t, u_n^*, a_n, b_n)\right) + \boldsymbol{v}_0, \tag{3.23}$$

for all $s \in [0, t]$, where $y_1, \ldots, y_n$ are the components of the vector $\boldsymbol{y}$ defined in (3.22) and the $i$-th element $\gamma(s; y_i, t, u_i^*, a_i, b_i)$ on the right-hand side is the one-dimensional trajectory defined by (3.10), (3.11), (3.12), and (3.14) for different cases of $y_i$, $t$, and $u_i^*$. Note that the $i$-th component of the minimizer $\boldsymbol{u}^*$ satisfies $u_i^* \in [y_i - a_i t, y_i + b_i t]$. In other words, $(y_i, t)$ is in the domain in (3.9) with $u = u_i^*$, and hence, the term $\gamma(s; y_i, t, u_i^*, a_i, b_i)$ on the right-hand side of (3.23) is well-defined for each $i \in \{1, \ldots, n\}$.

Next, we present two propositions showing that the functions $S$ and $\boldsymbol{\gamma}$ defined above do indeed solve the optimal control problem (3.1) and the corresponding HJ PDE (3.2). More specifically, Proposition 3.1.6 shows that the trajectory $s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t)$ is an optimal trajectory of the problem (3.1), whose optimal value equals $S(\boldsymbol{x}, t)$. If there are multiple minimizers in (3.21), Proposition 3.1.6 shows

that any minimizer $\boldsymbol{u}^*$ defines an optimal trajectory $s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t)$ by (3.23). Proposition 3.1.7 shows that the function $S$ defined in (3.21) is the viscosity solution to the HJ PDE (3.2).

**Proposition 3.1.6.** *Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a lower semi-continuous function. Let $\boldsymbol{a} = (a_1, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, \ldots, b_n)$ be two vectors in $(0, +\infty)^n$, $\boldsymbol{v}_0$ be a vector in $\mathbb{R}^n$, and $P$ be an invertible matrix with $n$ rows and $n$ columns. Then, for any $\boldsymbol{x} \in \mathbb{R}^n$ and $t > 0$, the function $[0, t] \ni s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t) \in \mathbb{R}$ defined in (3.23) is an optimal trajectory for the optimal control problem (3.1), where the matrix $M$ satisfies $M = PP^T$. The optimal value of the problem (3.1) equals $S(\boldsymbol{x}, t)$, as defined in (3.21). Moreover, if $J$ is convex, then the trajectory $s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t)$ is the unique optimal trajectory.*

*Proof.* The proof is provided in Section 3.3.4. □

**Proposition 3.1.7.** *Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a continuous function. Let $K\colon \mathbb{R}^n \to [0, +\infty)$ be a piecewise affine 1-homogeneous convex function. Assume that there exist linearly independent vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in \mathbb{R}^n$ and positive scalars $a_i, b_i > 0$ for each $i \in \{1, \ldots, n\}$, such that the sublevel set of $K$ satisfies*

$$\{\boldsymbol{x} \in \mathbb{R}^n\colon K(\boldsymbol{x}) \leq 1\} = \operatorname{co}\left(\bigcup_{j=1}^{n} \left\{\frac{1}{a_j}\boldsymbol{u}_j, -\frac{1}{b_j}\boldsymbol{u}_j\right\}\right),$$

*where* co *$E$ denotes the convex hull of a set $E$. Define the matrix $P$ to be the matrix whose columns are $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$, and define the matrix $M$ by $M := PP^T$. Then, the function $S\colon \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$ defined by (3.21) is the unique viscosity solution to the HJ PDE (3.2) in the solution set $C(\mathbb{R}^n \times [0, +\infty))$.*

*Proof.* The proof is provided in Section 3.3.5. □

## 3.2 Efficient algorithms and FPGA implementations

In this section, we present efficient algorithms for evaluating the optimal trajectory of the high-dimensional optimal control problem (3.17) as well as the solution of the corresponding high-dimensional HJ PDE (3.18). We note that our algorithms may be easily adjusted to solve the more general problems given in (3.1) and (3.2), but for simplicity of notation, we define our algorithms for (3.17) and (3.18), instead. To solve the more general problems in (3.1) and (3.2), one can first compute the minimizer and the minimal value of the optimization problem in (3.21) by applying our proposed algorithms to the new initial cost $\tilde{J}$ defined in (3.12) and the new terminal position $\boldsymbol{y}$ defined in (3.22) and then compute the optimal values and optimal trajectories using (3.21) and (3.23).

Recall that the representation formulas for problems (3.17) and (3.18) are provided in Section 3.1. Note that these problems are numerically solvable using the representation formulas if the optimization problem in (3.19) is numerically solvable. Therefore, in this section, we provide different methods to solve (3.19) for different classes of initial costs $J$. In Section 3.2.1, we consider quadratic initial costs and propose explicit formulas for solving (3.19) exactly. In Section 3.2.2, we consider convex initial costs and apply ADMM to solve (3.19). Note that ADMM can be replaced by any other appropriate convex optimization algorithm, and we only apply ADMM in this chapter for illustrative purposes. Furthermore, by applying optimization algorithms to the representation formula directly, we solve the

optimal control problem exactly, without discretizations or approximations of the original problem. In Section 3.2.3, we extend our methods in the previous sections to address a class of nonconvex initial costs using a min-plus technique. In each of these three sections, we provide numerical results using both a CPU implementation and an FPGA implementation to demonstrate the efficiency of our numerical solvers in various dimensions $n$ as well as their potential for real-time applications. All of our CPU results are run using an 11th Gen Intel Laptop Core i7-1165G7 with a 2.80GHz processor. All of our FPGA results are run using a Xilinx Alveo U280 board with a target design running at 300 MHz and double floating point precision. For a general overview of FPGAs, we refer the reader to [81].

To avoid confusion, we use $S$ and $\boldsymbol{\gamma}$ to denote the analytical solutions to the HJ PDEs and optimal control problems, while we use $\hat{S}$ and $\hat{\boldsymbol{\gamma}}$ to denote their numerical approximations obtained by our proposed methods.

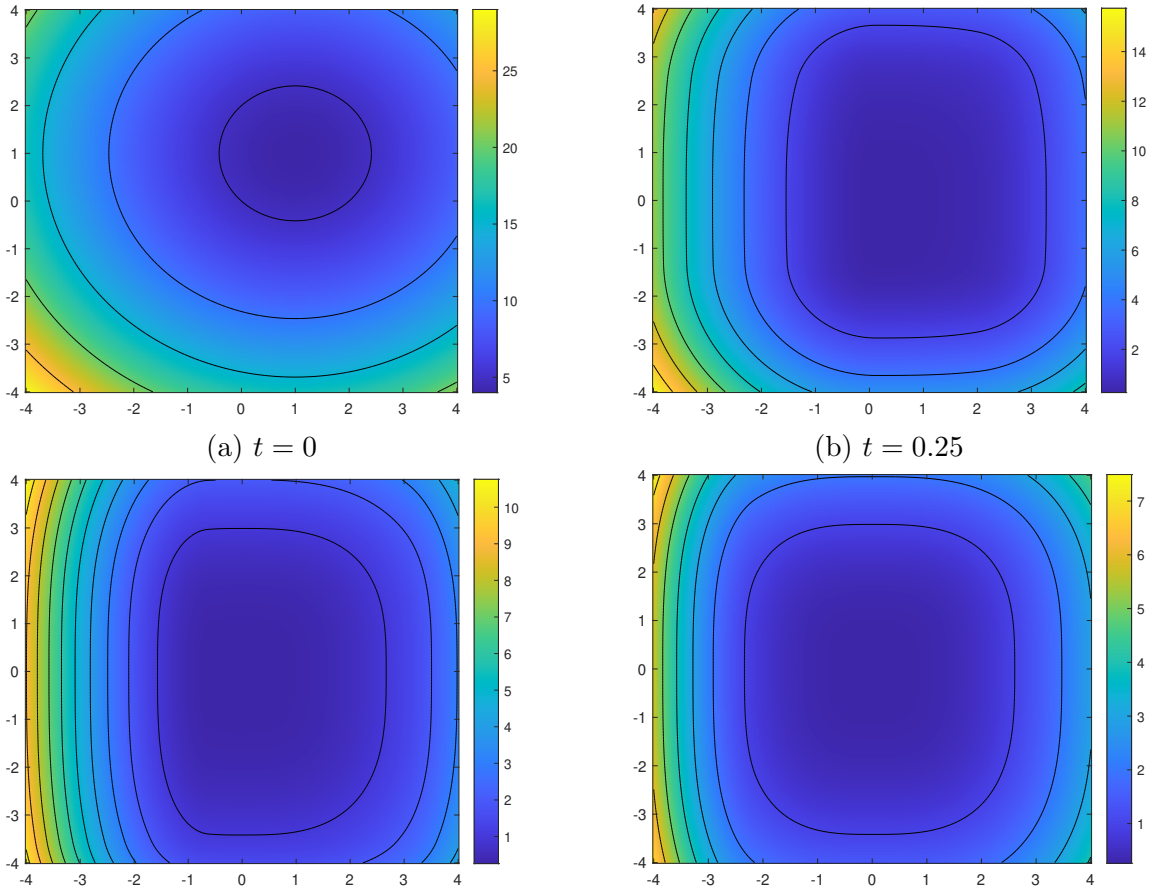
### 3.2.1 Quadratic initial costs

In this section, we present an exact numerical solver for solving the optimal control problem (3.17) and the corresponding HJ PDE (3.18) with quadratic initial costs $J$. Assume that the function $J$ is defined by:

$$J(\boldsymbol{x}) = \frac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 + \alpha \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \tag{3.1}$$

where $\boldsymbol{y} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$, and $\lambda > 0$ are some parameters. Recall that $\|\cdot\|$ denotes the $\ell^2$-norm in $\mathbb{R}^n$.

For this quadratic initial cost, the optimization problem (3.19) is equivalent to



(a) $t = 0$ (b) $t = 0.25$

(c) $t = 0.5$ (d) $t = 1$

Figure 3.2: Evaluation of the solution $\hat{S}(\boldsymbol{x}, t)$ of the 10-dimensional HJ PDE (3.18) with $\boldsymbol{a} = (4, 6, 5, \ldots, 5)$, $\boldsymbol{b} = (3, 9, 6, \ldots, 6)$, and initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|^2$ for $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$, where $(x_1, x_2) \in [-4, 4]^2$, and different times $t$. Plots for times $t = 0$, 0.25, 0.5, and 1 are depicted in (a)-(d), respectively. Level lines are superimposed on the plots.

computing the proximal point of the function $\boldsymbol{u} \mapsto \frac{1}{\lambda}\sum_{i=1}^{n} S(x_i, t; u_i, a_i, b_i)$, which can be split into $n$ one-dimensional subproblems where the $i$-th subproblem reads:

$$u_i^* = \underset{u \in [x_i - a_i t, x_i + b_i t]}{\arg\min} \left\{ S(x_i, t; u, a_i, b_i) + \frac{\lambda}{2}(u - y_i)^2 \right\}. \tag{3.2}$$

Hence, solving (3.19) in this case is embarrassingly parallel; these $n$ subproblems can be solved independently in parallel using the analytical solution (3.6) in Section 3.4.1, and the solution to the problems (3.17) and (3.18) can be obtained directly from (3.19) and (3.20) using the minimizer computed in (3.6). We use our numerical solver for this case as a building block for our numerical methods in Sections 3.2.2 and 3.2.3.

Since we implement our representation formula explicitly, the result of this numerical solver is exact up to machine precision. Furthermore, since the complexity for solving each subproblem (3.2) is $\Theta(1)$ (see the discussion in Section 3.4.1), the complexity for solving HJ PDE (3.18) with quadratic initial cost is $\Theta(n)$ and the curse of dimensionality is avoided in this case.

Now, we apply our proposed method to solve the high-dimensional HJ PDE (3.18) with quadratic initial cost $J$ defined by:

$$J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|^2 \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \tag{3.3}$$

i.e., we set $\boldsymbol{y} = \mathbf{1} = (1, 1, \ldots, 1) \in \mathbb{R}^n$, $\alpha = 0$, and $\lambda = 1$ in (3.1). We define the

parameters $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{R}^n$ and $\boldsymbol{b} = (b_1, \ldots, b_n) \in \mathbb{R}^n$ by:

$$a_i = \begin{cases} 4 & \text{if } i = 1, \\ 6 & \text{if } i = 2, \\ 5 & \text{if } i > 2, \end{cases} \qquad \text{and} \qquad b_i = \begin{cases} 3 & \text{if } i = 1, \\ 9 & \text{if } i = 2, \\ 6 & \text{if } i > 2. \end{cases} \tag{3.4}$$

Figure 3.2 depicts two-dimensional contour plots of the numerical solution $\hat{S}(\boldsymbol{x}, t)$ to this 10-dimensional HJ PDE (i.e., $n = 10$) with quadratic initial cost (3.3) at different positions $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$ and different times $t \in \{0, 0.25, 0.5, 1\}$.

| n | CPU time (s) | FPGA time (s) | Speedup |
|---|---|---|---|
| 4 | 6.4665e-08 | 1.334e-08 | 4.8475 |
| 8 | 1.6845e-07 | 2.667e-08 | 6.3161 |
| 12 | 4.6512e-07 | 4.000e-08 | 11.6280 |
| 16 | 7.4280e-07 | 5.334e-08 | 13.9258 |

Table 3.1: Comparison of the average time per call over $100,000$ runs for evaluating the solution of the HJ PDE (3.18) with quadratic initial condition (3.3) for various dimensions $n$ using a CPU implementation on a single Intel Core i7-1165G7 versus an FPGA implementation on a Xilinx Alveo U280 board with a frequency of 300 MHz.

| n | Latency (ns) | BRAM | DSPs | FFs | LUTs |
|---|---|---|---|---|---|
| 4 | 400,224 (1.334e06) | 0 (0%) | 847 (9%) | 91,716 (3%) | 55,345 (4%) |
| 8 | 800,238 (2.667e06) | 0 (0%) | 847 (9%) | 92,105 (3%) | 55,444 (4%) |
| 12 | 1,200,246 (4.000e06) | 0 (0%) | 847 (9%) | 92,301 (3%) | 55,467 (4%) |
| 16 | 1,600,258 (5.334e06) | 0 (0%) | 847 (9%) | 92,626 (3%) | 55,517 (4%) |

Table 3.2: FPGA resources and latencies in cycles and nanoseconds (ns) for evaluating the solution of the HJ PDE (3.18) with quadratic initial condition (3.3) at $100,000$ points $(\boldsymbol{x}, t, \boldsymbol{y}) \in \mathbb{R}^n \times [0, \infty) \times \mathbb{R}^n$ for various dimensions $n$ using double precision floating points on a Xilinx Alveo U280 board with a frequency of 300 MHz.

The running time using either a CPU or an FPGA implementation of our numerical solver in different dimensions is shown in Table 3.1. To compute the running time, we first compute the overall running time for computing the solution at $100,000$ random points $(\boldsymbol{x}, t) \in [-4, 4]^n \times [0, 0.5]$ and then report the average running time for computing the solution $\hat{S}(\boldsymbol{x}, t)$ at one point $(\boldsymbol{x}, t)$ over these $100,000$ trials. From

Table 3.1, we see that, using a CPU implementation, it takes less than $8 \times 10^{-7}$ seconds to compute the solution at one point in a 16-dimensional problem, which demonstrates the efficiency of our proposed solver even in high dimensions. However, using our FPGA implementation, it takes less than $6 \times 10^{-8}$ seconds to compute the solution at one point in a 16-dimensional problem, for approximately a 14× speed up over the CPU implementation in dimension 16.

We achieve this speedup by designing our FPGA implementation to have high throughput, where throughput refers to the amount of data that can be processed in a given amount of time. Specifically, we design our FPGA implementation to have an iteration interval (II) of 1, which means that we can begin processing a new input at every FPGA clock cycle (e.g., for our implementation, every 3.3333 nanoseconds). The inputs of our FPGA kernel are the points $(x_i, t, y_i) \in \mathbb{R} \times [0, \infty) \times \mathbb{R}$ as defined in (3.2). In other words, our FPGA implementation streams the points $(\boldsymbol{x}, t, \boldsymbol{y}) \in \mathbb{R}^n \times [0, \infty) \times \mathbb{R}^n$ elementwise. In contrast, the CPU implementation achieves its performance by relying on both elementwise (i.e., solving the one-dimensional subproblems (3.2) $n$ times) and pointwise (i.e., solving the $n$-dimensional problem (3.19) for multiple points $(\boldsymbol{x}, t, \boldsymbol{y}) \in \mathbb{R}^n \times [0, \infty) \times \mathbb{R}^n$) parallelism, but must execute these parallelized tasks sequentially. Thus, as the dimension $n$ increases, the CPU is able to parallelize fewer points $(\boldsymbol{x}, t, \boldsymbol{y}) \in \mathbb{R}^n \times [0, \infty) \times \mathbb{R}^n$ at a time, and its performance degrades by some multiplicative factor as $n$ increases. However, due to its elementwise streaming and II of 1, our FPGA implementation achieves average runtimes that only increase as $n$ times the length of one FPGA clock cycle, or, in other words, as the dimension $n$ increases, the performance of the FPGA implementation degrades only by some small additive amount. As a result, not only does our FPGA implementation achieve a speedup over the CPU implementation in lower dimensions (e.g., a speedup of about 5 in dimension 4), but this speedup becomes

more pronounced as the dimension increases.

In Table 3.2, we present the amount of FPGA resources and latencies used to implement and run the FPGA implementation of our numerical solver for various dimemsions $n$ and 100,000 points $(\boldsymbol{x}, t, \boldsymbol{y}) \in \mathbb{R}^n \times [0, \infty) \times \mathbb{R}^n$. We observe that since our design streams the points $(\boldsymbol{x}, t, \boldsymbol{y}) \in \mathbb{R}^n \times [0, \infty) \times \mathbb{R}^n$ elementwise (i.e., our FPGA kernel takes the inputs $(x_i, t, y_i) \in \mathbb{R} \times [0, \infty) \times \mathbb{R}$) the latency of our FPGA implementation scales linearly in the dimension $n$ and the amount of FPGA resources used remains essentially constant in $n$.

Recall that the Alveo U280 board consists of three "chiplets." Since routing resources between chiplets are limited, crossing chiplets can severely degrade performance [123, 114]. As such, we design our FPGA implementation to use less than 30% of any given type of FPGA resource (e.g., flip flops (FFs), lookup tables (LUTs), digital signal processing units (DSPs), block random access memory (BRAM), etc.) to ensure that no chiplet is crossed. Since our design uses less than 30% of the FPGA resources available on the Xilinx Alveo U280 board, we could either use a smaller (i.e., cheaper) FPGA to implement our numerical solver with similar performance as we report here or we could parallelize by simply implementing multiple, independent copies of our FPGA kernel to maximize usage of the FPGA board. In the latter case, we could achieve a further speedup of $\times 9$ (i.e., 3 copies of our FPGA kernel per each of the 3 chiplets, ensuring that no kernel requires crossing chiplets) for a total speedup of about 44 to 125 over the CPU depending on the dimension $n$.

**Algorithm 3:** An ADMM algorithm for solving the optimal control problem (3.17) and the corresponding HJ PDE (3.18) with convex initial cost.

**Inputs :** Parameters $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, terminal position $\boldsymbol{x} \in \mathbb{R}^n$, time horizon $t > 0$, running time $s > 0$ of the trajectory, and convex initial cost $J \colon \mathbb{R}^n \to \mathbb{R}$ in the problem (3.17) and the PDE (3.18). Parameter $\lambda > 0$, initialization $\boldsymbol{d}^0 \in \mathbb{R}^n$, $\boldsymbol{w}^0 \in \mathbb{R}^n$, and error tolerance $\epsilon > 0$ for the ADMM scheme.

**Outputs:** The optimal trajectory $\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t)$ in the optimal control problem (3.17) and the solution value $\hat{S}(\boldsymbol{x}, t)$ to the corresponding HJ PDE (3.18).

**for** $k = 1, 2, \ldots$ **do**

  Update $\boldsymbol{v}^{k+1} \in \mathbb{R}^n$ by:

$$\boldsymbol{v}^{k+1} = \arg\min_{\boldsymbol{v} \in \mathbb{R}^n} \left\{ J(\boldsymbol{v}) + \frac{\lambda}{2} \left\| \boldsymbol{v} - \boldsymbol{d}^k + \boldsymbol{w}^k \right\|^2 \right\}. \tag{3.5}$$

  Update $\boldsymbol{d}^{k+1} \in \mathbb{R}^n$, where the $i$-th element $d_i^{k+1}$ is updated by:

$$d_i^{k+1} = \arg\min_{d_i \in \mathbb{R}} \left\{ S(x_i, t; d_i, a_i, b_i) + \frac{\lambda}{2} (v_i^{k+1} - d_i + w_i^k)^2 \right\}. \tag{3.6}$$

  Update $\boldsymbol{w}^{k+1} \in \mathbb{R}^n$ by:

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \boldsymbol{v}^{k+1} - \boldsymbol{d}^{k+1}.$$

  **if** $\|\boldsymbol{v}^{k+1} - \boldsymbol{v}^k\|^2 \le \epsilon$, $\|\boldsymbol{d}^{k+1} - \boldsymbol{d}^k\|^2 \le \epsilon$, *and* $\|\boldsymbol{v}^{k+1} - \boldsymbol{d}^{k+1}\|^2 \le \epsilon$ **then**

    set $N = k + 1$ and $\boldsymbol{u}^N = \boldsymbol{d}^N$;

    break;

  **end**

**end**

Output the optimal trajectory by:

$$\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t) = (\gamma(s; x_1, t, u_1^N, a_1, b_1), \ldots, \gamma(s; x_n, t, u_n^N, a_n, b_n)),$$

where the $i$-th component $\gamma(s; x_i, t, u_i^N, a_i, b_i)$ is defined by (3.10), (3.11), (3.12), and (3.14). Also, output the solution to the HJ PDE by:

$$\hat{S}(\boldsymbol{x}, t) = \sum_{i=1}^{n} S\left(x_i, t; u_i^N, a_i, b_i\right) + J\left(\boldsymbol{u}^N\right),$$

where the $i$-th component $S\left(x_i, t; u_i^N, a_i, b_i\right)$ in the summation is defined by (3.7) and (3.13).

### 3.2.2 Convex initial costs

In this section, we solve (3.17) and (3.18) with convex initial cost $J$. To solve these problems, we need to solve the convex optimization problem in the representation formula (3.19), which can be solved using many possible convex optimization algorithms. Based on the discussion in Section 3.2.1, proximal point-based methods would be a reasonable approach.

For illustrative purposes, in this section, we apply ADMM (see [59, 15]) to (3.19) with certain convex initial costs $J$ whose proximal points are numerically computable. The details of applying ADMM to this problem are given in Algorithm 3. We emphasize that ADMM is not the only possible optimization algorithm that can be applied here. Rather, any appropriate optimization algorithm can be applied to (3.19), the choice of which depends on the properties of the function $J$ and among which the use of ADMM in Algorithm 3 is simply one such possible choice.

In each iteration of ADMM in Algorithm 3, we first update $\boldsymbol{v}^{k+1}$ using (3.5). The vector $\boldsymbol{v}^{k+1}$ is the proximal point of $\frac{J}{\lambda}$ at $\boldsymbol{d}^k - \boldsymbol{w}^k$, which is assumed to be numerically computable. Then, we compute $\boldsymbol{d}^{k+1}$ componentwise using (3.6), which can be solved in parallel. More specifically, we apply the solver in Section 3.4.1 to solve (3.6), and hence, the complexity for computing $\boldsymbol{d}^{k+1}$ is $\Theta(n)$. Note that the update step for $\boldsymbol{d}^{k+1}$ in Algorithm 3 has the same form as solving (3.19) with quadratic initial cost $\boldsymbol{x} \mapsto \frac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{v}^{k+1} - \boldsymbol{w}^k\|^2$. As a result, the solver proposed in Section 3.2.1 serves as a building block in our ADMM algorithm (Algorithm 3), and the running time in Table 3.1 underpins the running time for updating $\boldsymbol{d}^{k+1}$ in each iteration of ADMM. Additionally, since we apply ADMM to the representation formula directly, we do not rely on discretizations or approximations of the optimal

control problem. Instead, we solve the problem exactly.

In the following proposition, we prove that the optimal trajectory $\hat{\boldsymbol{\gamma}}$ and the solution value $\hat{S}$ as computed by Algorithm 3 do indeed converge to their analytical counterparts as the number of ADMM iterates $N$ approaches infinity.

**Proposition 3.2.1.** *Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function and $\boldsymbol{a}, \boldsymbol{b}$ be two vectors in $(0, +\infty)^n$. Let $\boldsymbol{x}$ be any vector in $\mathbb{R}^n$ and $t > 0$ be any scalar. Let $S$ and $\boldsymbol{\gamma}$ be the functions defined in (3.19) and (3.20), respectively. Let $\lambda > 0$ and the initialization $\boldsymbol{d}^0, \boldsymbol{w}^0 \in \mathbb{R}^n$ be arbitrary parameters for Algorithm 3. Let $\hat{S}^N$ and $\hat{\boldsymbol{\gamma}}^N$ be the output solution and trajectory, respectively, from Algorithm 3 with iteration number $N$. Then, we have*

$$\lim_{N\to\infty} \hat{S}^N(\boldsymbol{x}, t) = S(\boldsymbol{x}, t) \quad \text{and} \quad \lim_{N\to\infty} \sup_{s\in[0,t]} \|\hat{\boldsymbol{\gamma}}^N(s; \boldsymbol{x}, t) - \boldsymbol{\gamma}(s; \boldsymbol{x}, t)\| = 0. \tag{3.7}$$

*Proof.* The proof is provided in Section 3.5.1. □

The convergence of the output optimal trajectory $\hat{\boldsymbol{\gamma}}^N$ and solution $\hat{S}^N$ from Algorithm 3 are proved in the proposition above. For a general convex function $J$, the convergence rate of the output solution $\hat{S}^N(\boldsymbol{x}, t)$ is $o(\frac{1}{\sqrt{N}})$ if the best iteration (in terms of having the smallest objective function value among the first $N$ iterations) is selected as the output. This convergence rate can be improved to $O(\frac{1}{N})$ if the output $\boldsymbol{u}^N$ is chosen in an ergodic manner, i.e., by setting the output $\boldsymbol{u}^N$ to be $\frac{1}{N}\sum_{k=1}^{N} \boldsymbol{d}^k$. Moreover, when the initial condition $J$ satisfies stronger assumptions (for instance, if $J$ is strongly convex and differentiable with Lipschitz gradient), we obtain linear convergence for $\boldsymbol{u}^N$, the output solution $\hat{S}^N$, and the output trajectory $\hat{\boldsymbol{\gamma}}^N$. For more details on the convergence rates, we refer readers to [43, 39].

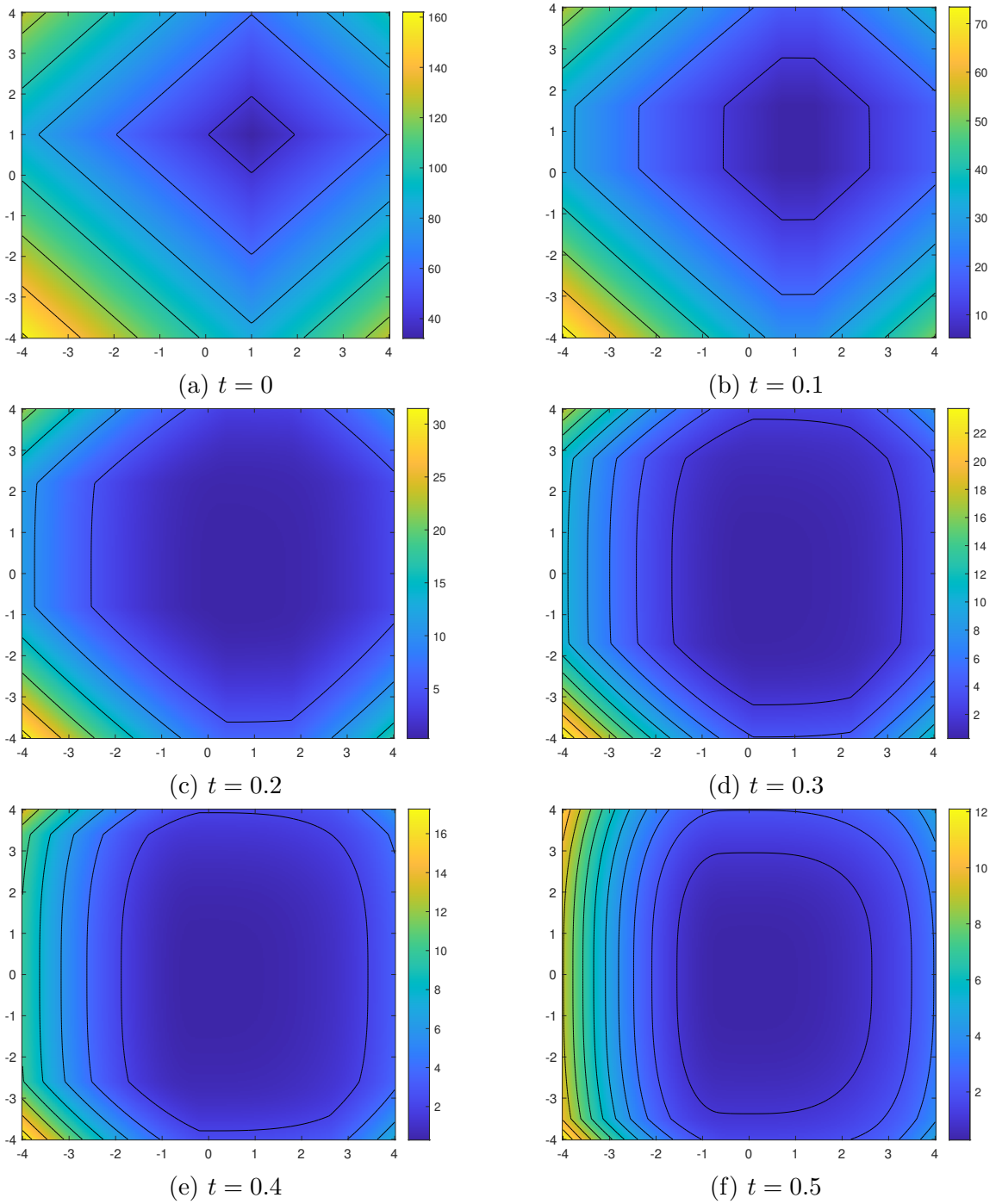
#### 3.2.2.1 Numerical example

Now, we show a numerical example solved using Algorithm 3 for the optimal control problem (3.17) and the HJ PDE (3.18) with convex initial cost $J$ defined by:

$$J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2 \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \tag{3.8}$$

where $\|\cdot\|_1$ denotes the $\ell^1$-norm in $\mathbb{R}^n$ and $\mathbf{1}$ is the $n$-dimensional vector whose components are all 1's. We set the parameters $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ to be the vectors defined in (3.4), i.e., $\boldsymbol{a} = (4, 6, 5, \ldots, 5)$ and $\boldsymbol{b} = (3, 9, 6, \ldots, 6)$. With these parameters and initial cost, we apply the ADMM algorithm in Algorithm 3 to solve (3.17) and (3.18). We set the parameters in Algorithm 3 to be $\lambda = 1$, $\boldsymbol{d}^0 = \boldsymbol{x}$, $\boldsymbol{w}^0 = \mathbf{0}$, and $\epsilon = 10^{-8}$. In order to solve (3.6), we apply the efficient method described in [38, Section 4.4], which has complexity $\Theta(n)$. Therefore, the complexity for each ADMM iteration in Algorithm 3 is also $\Theta(n)$. In other words, if the number of iterations is fixed, the curse of dimensionality is avoided in this example.

We solve the problem in 10 dimensions (i.e., we set $n = 10$) and plot the solution $\hat{S}$ and the optimal trajectories $\hat{\boldsymbol{\gamma}}$ in Figure 3.3 and Figure 3.4, respectively. Figure 3.3 depicts two-dimensional slices of the solution $\hat{S}(\boldsymbol{x}, t)$, as computed using Algorithm 3, of the HJ PDE (3.18) at different positions $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$ and at different times $t$. As expected, in Figure 3.3(a), we see that the initial condition $J$ is not smooth, e.g., we see kinks in the contour plots near $(x_1, x_2) = (1, -1), (-2, 1), (4, 1)$, and $(1, 4)$. In Figures 3.3(b)-(f), we see that the solution continues to evolve with several kinks as well. These kinks help numerically verify that our algorithm does indeed compute the non-smooth viscosity solution to the corresponding HJ PDE. Overall, the solution appears to be continuous in $(x_1, x_2)$ at all times $t$, which is consistent



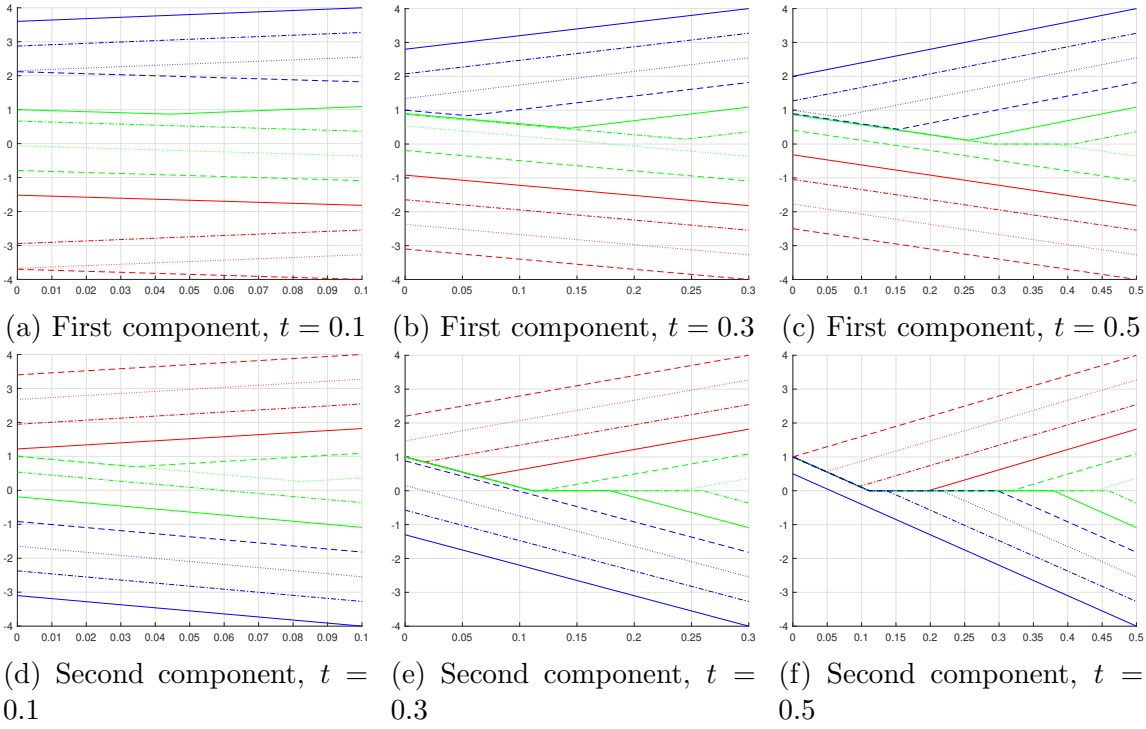
(a) $t = 0$ (b) $t = 0.1$

(c) $t = 0.2$ (d) $t = 0.3$

(e) $t = 0.4$ (f) $t = 0.5$

Figure 3.3: Evaluation of the solution $\hat{S}(\boldsymbol{x}, t)$ of the 10-dimensional HJ PDE (3.18) with $\boldsymbol{a} = (4, 6, 5, \ldots, 5)$, $\boldsymbol{b} = (3, 9, 6, \ldots, 6)$, and initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$ for $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$, where $(x_1, x_2) \in [-4, 4]^2$, and different times $t$. Plots for $t = 0$, 0.1, 0.2, 0.3, 0.4, and 0.5 are depicted in (a)-(f), respectively. Level lines are superimposed on the plots.

with the results of Proposition 3.1.5.



(a) First component, $t = 0.1$ (b) First component, $t = 0.3$ (c) First component, $t = 0.5$

(d) Second component, $t = 0.1$ (e) Second component, $t = 0.3$ (f) Second component, $t = 0.5$

Figure 3.4: Evaluation of the optimal trajectory $\hat{\boldsymbol{\gamma}}(s; (x, -x, 0, \ldots, 0), t)$ of the 10-dimensional optimal control problem (3.17) with $\boldsymbol{a} = (4, 6, 5, \ldots, 5)$, $\boldsymbol{b} = (3, 9, 6, \ldots, 6)$, and initial cost $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$ versus $s \in [0, t]$ for different terminal positions $(x, -x, 0, \ldots, 0)$ ($x \in [-4, 4]$) and different time horizons $t$. The different colors and line markers simply help differentiate between the different trajectories. Figures (a)-(c) depict the first component of the trajectory versus $s \in [0, t]$ with different time horizons $t$, while Figures (d)-(f) depict the second component of the trajectory versus $s \in [0, t]$ with different time horizons $t$. Plots for time horizons $t = 0.1$, 0.3, and 0.5 are depicted in Figures (a)/(d), (b)/(e), and (c)/(f), respectively. We note that because of our choice of $\boldsymbol{a}$ and $\boldsymbol{b}$, the piecewise slopes of our trajectories are not symmetric about 0.

Figure 3.4 depicts one-dimensional slices of the optimal trajectory $\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t)$ of the corresponding 10-dimensional optimal control problem (3.17), using different terminal positions $\boldsymbol{x} = (x, -x, 0, \ldots, 0)$ for $x \in [-4, 4]$ and different time horizons $t$. We observe that the one-dimensional slices are piecewise linear and continuous in $s$, which is consistent with the properties of the formulas in (3.10), (3.11), (3.12), and (3.14). We note that in each subplot, all line segments with positive slope are parallel with slope $a_i$ (i.e., the $i$-th component of the trajectory has velocity $a_i$),

while all lines segments with negative slope are parallel with slope $-b_i$ (i.e., the $i$-th component of the trajectory has velocity $-b_i$). As such, in any given subplot, the piecewise slopes of the trajectories are not symmetric about 0 due to our choice of $\boldsymbol{a}$ and $\boldsymbol{b}$.

| n | CPU time (s) | FPGA time (s) | Speedup |
|---|---|---|---|
| 4 | 5.3711e-08 | 9.1888e-09 | 5.8453 |
| 8 | 1.1719e-07 | 3.6775e-08 | 3.1867 |
| 12 | 1.8880e-07 | 6.9133e-08 | 2.7310 |
| 16 | 2.7344e-07 | 1.1555e-07 | 2.3664 |

(a) Comparison of the average time per 1 iteration of ADMM over 100,000 runs for various dimensions $n$.

| n | $N$ | Latency (ns) | Interval (ns) | BRAM | DSPs | FFs | LUTs |
|---|---|---|---|---|---|---|---|
| 4 | 8 | 2,205,620 (7.351e06) | 2,200,087 (7.334e06) | 0 (0%) | 5,840 (64%) | 2,606,704 (99%) | 1,160,176 (88%) |
| 8 | 4 | 4,413,012 (1.471e07) | 4,400,085 (1.467e07) | 0 (0%) | 4,955 (54%) | 2,630,997 (100%) | 870,327 (66%) |
| 12 | 3 | 6,222,468 (2.074e07) | 6,200,127 (2.067e07) | 0 (0%) | 4,949 (54%) | 2,515,392 (96%) | 948,691 (72%) |
| 16 | 2 | 6,933,965 (2.311e07) | 6,900,222 (2.300e07) | 0 (0%) | 3,341 (37%) | 2,149,905 (82%) | 740,551 (56%) |

(b) Latency and interval in cycles and nanoseconds (ns) and FPGA resources used to implement $N$ ADMM iterations for evaluating the solution at 100,000 points $(\boldsymbol{x}, t) \in \mathbb{R}^n \times [0, \infty)$.

Table 3.3: Results for a high throughput FPGA implementation of ADMM for evaluating the solution of the HJ PDE (3.18) with initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$. The FPGA implementation uses a Xilinx Alveo U280 board with a frequency of 300 MHz. In (a), the CPU implementation uses a single Intel Core i7-1165G7, and the number of ADMM iterates used per run is the same as the number of iterates $N$ used in (b) for each dimension $n$.

Next, we describe two different FPGA implementations of ADMM for this example. Specifically, we present a high throughput implementation and a low latency implementation, the results for which are shown in Tables 3.3 and 3.4, respectively. Latency refers to the amount of time that it takes for an input to finish being processed. The "optimality" of a given FPGA implementation is usually determined by its latency (where lower latency is more optimal), its throughput (where higher throughput is more optimal), or the amount of resources used (where fewer resources is more optimal). However, optimizing for one of these criteria usually competes with the optimization of another criteria, and thus, one can at best only expect a Pareto

| n | CPU time (s) | FPGA time (s) | Speedup |
|---|---|---|---|
| 4 | 5.4254e-08 | 8.4816e-08 | 0.6397 |
| 8 | 1.1161e-07 | 1.1857e-07 | 0.9413 |
| 12 | 1.8359e-07 | 1.8067e-07 | 1.0162 |
| 16 | 2.5879e-07 | 2.3167e-07 | 1.1171 |

(a) Comparison of the average time per 1 iteration of ADMM over 100,000 runs for various dimensions $n$.

| n | $N$ | Latency (ns) | Interval (ns) | BRAM | DSPs | FFs | LUTs |
|---|---|---|---|---|---|---|---|
| 4 | 9 | 433 (1.443e03) | 229 (7.633e02) | 0 (0%) | 2,608 (28%) | 612,037 (23%) | 390,421 (29%) |
| 8 | 7 | 405 (1.350e03) | 249 (8.300e02) | 0 (0%) | 2,211 (24%) | 616,758 (23%) | 377,262 (28%) |
| 12 | 5 | 377 (1.257e03) | 271 (9.033e02) | 0 (0%) | 1,949 (21%) | 629,957 (24%) | 387,615 (29%) |
| 16 | 4 | 371 (1.237e03) | 278 (9.267e02) | 0 (0%) | 1,898 (21%) | 620,606 (23%) | 382,723 (29%) |

(b) Latency and interval in cycles and nanoseconds (ns) and FPGA resources used to implement $N$ ADMM iterations for evaluating the solution at 1 point $(\boldsymbol{x}, t) \in \mathbb{R}^n \times [0, \infty)$.

Table 3.4: Results for a low latency FPGA implementation of ADMM for evaluating the solution of the HJ PDE (3.18) with initial condition $J(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \mathbf{1}\|_1^2$. The FPGA implementation uses a Xilinx Alveo U280 board with a frequency of 300 MHz. In (a), the CPU implementation uses a single Intel Core i7-1165G7, and the number of ADMM iterates used per run is the same as the number of iterates $N$ used in (b) for each dimension $n$.

optimal implementation. For example, a high throughput implementation typically has a relatively high latency and vice versa. In fact, we observe that this trend holds for our FPGA implementations. For instance, our high throughput implementation has latencies (in cycles) per ADMM iteration of approximately 694.4, 3,242.8, 7,467.7, and 16,906.0 for dimensions $n = 4, 8, 12$, and 16, respectively. Meanwhile, our low latency implementation has latencies (in cycles) per ADMM iteration of approximately 48.1, 57.9, 75.4, and 92.8 for dimensions $n = 4, 8, 12$, and 16, respectively. Note that the latency and interval listed in Tables 3.3b and 3.4b correspond to the latency and interval for processing $N$ ADMM iterations for all 100,000 points (in Table 3.3b) or 1 point (in Table 3.4b), respectively. Thus, we can compute the latency per ADMM iteration using the values in Tables 3.3b and 3.4b as follows:

$$\frac{\text{latency} - \text{interval} + \text{interval}/\#\text{ points}}{N},$$

where # points = 100,000 in Table 3.3b, # points = 1 in Table 3.4b, and the numerator in the above formula corresponds to the latency (for $N$ ADMM iterations) per point.

The benefit of a high throughput implementation is that it achieves low average runtime. Hence, high throughput implementations are best suited for offline computations, where a computational experiment needs to be run many times for many different inputs. We can expect a high throughput implementation to achieve the best speedup compared to a CPU implementation, whose performance is typically also measured by average runtime. For example, in Table 3.3a, we see that our high throughput implementation achieves a speedup of about 2-6 times the average runtime of the CPU depending on the dimension $n$, whereas in Table 3.4a, we see that the average runtime of our low latency implementation is approximately the same as that of the CPU for each $n$. Here, we compute the average runtime using the procedure described in Section 3.2.1, except we average over the number of ADMM iterations in addition to the number of runs.

In contrast, low latencies are best suited for online computations, where results are required to be available within some fixed short period of time after an input is provided. Such online computations are critical for real-time optimal control applications. While it is impossible to measure the latency of a CPU implementation, FPGA implementations have guaranteed latencies. For example, as computed above, our low latency implementation computes one iteration of ADMM in 92.8 clock cycles (or approximately $3.0933 \times 10^{-7}$ seconds) for the 16-dimensional problem, but this quantity would not be able to be measured on a CPU. Hence, our low latency FPGA implementation achieves similar performance (in terms of throughput; e.g., see Table 3.4a) as the CPU but with a guaranteed (low) latency.

In both FPGA implementations, we stream both the points $(\boldsymbol{x}, t) \in \mathbb{R}^n \times [0, \infty)$ and the ADMM iterates $\boldsymbol{v}^k, \boldsymbol{d}^k, \boldsymbol{w}^k \in \mathbb{R}^n$ between consecutive ADMM iterations, where the number of ADMM iterations per implementation is determined by the the amount of resources available. For the high throughput implementation, the amount of resources used per ADMM iteration is high (and increases with the dimension $n$), and we aim to use as many of the resources as possible in order to maximize the throughput. Thus, our high throughput FPGA kernel must cross chiplets. In order to ensure that the performance does not degrade due to the crossing of chiplets, we stream $(\boldsymbol{x}, t, \boldsymbol{v}^k, \boldsymbol{d}^k, \boldsymbol{w}^k)$ in a single stream of doubles. As a result, (in contrast with the high throughput FPGA implementation in Section 3.2.1) our high throughput implementation for ADMM cannot achieve an II of 1. Instead, the II of our high throughput FPGA kernel (per point) is lower bounded by $4n + 1$, the dimension of the concatenated vector $(\boldsymbol{x}, t, \boldsymbol{v}^k, \boldsymbol{d}^k, \boldsymbol{w}^k)$, which means that its II per ADMM iteration is lower bounded by $(4n + 1)/N$, where $N$ is the total number of ADMM iterations implemented. For example, for our high throughput implementation, the II per ADMM iteration is about 2.8, 11.0, 20.7, and 34.5 cycles for dimensions $n = 4, 8, 12$, and 16, respectively. Note that the II per ADMM iteration can be computed using the quantities in Table 3.3b as $\frac{\text{interval}}{\#\text{ points}\times\text{N}}$. We also note that the average runtime of our high throughput FPGA implementation would be improved if we could implement more ADMM iterations on the FPGA. However, as we observe in Table 3.3b, our current implementation is heavily limited by the number of flip flops used. Theoretically, we should be able to reduce the number of flip flops using other FPGA resources, such as BRAM or URAM, instead, but we leave this for future research.

In contrast, for the low latency FPGA implementation, we stream each of the quantities $\boldsymbol{x}, t, \boldsymbol{v}^k, \boldsymbol{d}^k$, and $\boldsymbol{w}^k$ separately. Using separate streams for these quanti-

ties ensures that these quantities are available more immediately for computations, which is critical for achieving a low latency. However, using multiple streams also means that crossing chiplets is likely to cause a significant decrease in performance. Hence, we aim for designs that use less than 30% of any given FPGA resource to ensure that no chiplet is crossed. In Table 3.4b, we see that our low latency FPGA implementation meets this constraint.

#### 3.2.2.2 A note on negative slack

As discussed in the previous section, our high throughput FPGA implementation requires us to use a relatively high II ($> 4n + 1$). However, (using Xilinx Vitis HLS 2022.2) having an II greater than 1 often leads to negative slack warnings. Negative slack indicates that the timing results provided by the software may not be possible, potentially due to routing issues or other physical constraints on resource placement on the FPGA board. In the literature, there is almost no advice on how to avoid these slack issues, even though many Vitis HLS users seem to experience this issue.

To our understanding, increasing the II increases the likelihood of negative slack because Vitis HLS seems to prioritize reusing computational resources when the II is high. Specifically, we observed that with an II of 1, the number of operations (e.g., additions, multiplications, divisions, etc.) implemented by the software can usually be predicted by directly counting the number of those operations (e.g., +, *, /, etc.) that are written in the C/C++ code. Then, when we increase the II, the number of each type of operation implemented by the software can typically be predicted as $\lceil \frac{\text{\# of occurrences of the operation in the code}}{\text{II}} \rceil$. Thus, in the extreme case where the II is very high, Vitis HLS may attempt to implement at most one of each operation type used, which we infer causes routing issues related to negative slack.

```
void foo (stream_t &in, stream_t &out) {
    // some computations a
    // some computations b
}
```

Listing 1: Example of some pseudocode that may cause negative slack.

```
void foo_a (stream_t &in, stream_t &tmp) {
    // some computations a
}

void foo_b (stream_t &tmp, stream_t &out) {
    // some computations b
}

void foo (stream_t &in, stream_t &out) {
    stream_t tmp;
    foo_a(in, tmp);
    foo_b(tmp, out);
}
```

Listing 2: Example of splitting the pseudocode in Listing 1 to (possibly) get rid of negative slack.

In our implementation, we overcome the negative slack issue by "splitting." Specifically, consider a function that yields a negative slack warning (e.g., the function in Listing 1). Then, we split this function into two (or more) sub-functions, such that we stream between the sub-functions, the computations of the original function are split between the sub-functions, and the result of chaining these sub-functions together is the same as the result of the original function (e.g., as in Listing 2). By splitting the problematic function, we effectively force the software to use more computational resources by creating more intermediate streams. While splitting generally seems effective in overcoming negative slack, this methodology has several drawbacks. In particular, splitting makes the HLS code less readable and more difficult to write, increases the amount of resources used (specifically, it seems to significantly increase the amount of flip flops used), and may require the II to be

increased further in order to relay intermediate values between the resulting sub-functions (if we do not allow the width of the streams to increase). Furthermore, it is not always predictable how many times a function needs to be split or how the computations should be divided between sub-functions in order for the negative slack to disappear.

Alternatively, although we do not not use it in our implementation, the ALLOCATION pragma is also sometimes useful in relieving negative slack. The ALLOCATION pragma allows us to specify how many of a particular operation (e.g., multiplication, addition, etc.) we want the software to implement. While using this pragma is easier to code than the splitting method described above, picking the correct amount of each operation to allocate is not easy to determine a priori. For example, in some cases, we found that we had to allocate an excessive amount of each operation to get rid of the negative slack, while in other cases, seemingly no choice mitigated the negative slack.

Thus, although we are able to provide some possible solutions for removing negative slack, our solutions are clearly not robust and can even pose additional challenges. Additionally, this slack issue is not unique to our particular implementation of ADMM and seems to arise in a wide range of FPGA designs. As such, further research into this issue is needed in order to establish reliable methodology for creating FPGA implementations of general scientific computing methods.

### 3.2.3 Certain nonconvex initial costs

In this section, we use min-plus techniques to extend our Lax-Oleinik-type representation formulas (3.15), (3.19), and (3.21) to handle a certain class of nonconvex

initial costs. Moreover, we propose an algorithm based on the resulting extended representation formulas, which uses the numerical methods in Sections 3.2.1 and 3.2.2 (or any possible algorithm for solving (3.15), (3.19), and (3.21)) as building blocks. From the numerical results and resulting running times, our proposed algorithm is shown to be able to solve the optimal control problems and corresponding HJ PDEs with these nonconvex initial costs efficiently.

We have already shown in Sections 3.2.1 and 3.2.2 that the solutions in (3.15), (3.19), and (3.21) are computable using convex optimization methods, such as Algorithm 3 if the initial cost $J\colon \mathbb{R}^n \to \mathbb{R}$ is a convex function. However, these representation formulas are also computable for a broader class of initial costs $J$. Consider the following nonconvex initial condition:

$$J(\boldsymbol{x}) = \min_{j\in\{1,\dots,m\}} J_j(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \tag{3.9}$$

where $J_j\colon \mathbb{R}^n \to \mathbb{R}$ is a convex function for each $j \in \{1,\dots,m\}$. In this case, the min-plus technique (see [84, 98]) is applied, and the optimization problem in (3.21) can be written as

$$\begin{aligned}
&S(\boldsymbol{x},t)\\
&= \inf_{\boldsymbol{u}\in\prod_{i=1}^n [y_i - a_i t, y_i + b_i t]} \left\{ \sum_{i=1}^n S(y_i,t;u_i,a_i,b_i) + \min_{j\in\{1,\dots,m\}} J_j\left((P^T)^{-1}\boldsymbol{u} + \boldsymbol{v}_0\right) \right\}\\
&= \min_{j\in\{1,\dots,m\}} \left\{ \inf_{\boldsymbol{u}\in\prod_{i=1}^n [y_i - a_i t, y_i + b_i t]} \left\{ \sum_{i=1}^n S(y_i,t;u_i,a_i,b_i) + J_j\left((P^T)^{-1}\boldsymbol{u} + \boldsymbol{v}_0\right) \right\} \right\}\\
&=: \min_{j\in\{1,\dots,m\}} S_j(\boldsymbol{x},t),
\end{aligned} \tag{3.10}$$

where $\boldsymbol{y} = (y_1,\dots,y_n)$ is the vector defined in (3.22), and in the last line, the function

$S_j \colon \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}$ for each $j \in \{1, \ldots, m\}$ is defined by:

$$S_j(\boldsymbol{x}, t) := \inf_{\boldsymbol{u} \in \prod_{i=1}^n [y_i - a_i t, y_i + b_i t]} \left\{ \sum_{i=1}^n S(y_i, t; u_i, a_i, b_i) + J_j\left((P^T)^{-1}\boldsymbol{u} + \boldsymbol{v}_0\right) \right\}, \quad (3.11)$$

which is the solution to the corresponding HJ PDE with convex initial cost $J_j$. Therefore, to compute $S(\boldsymbol{x}, t)$, this problem is divided into $m$ subproblems. In the $j$-th subproblem, convex optimization methods (e.g., the solver in Section 3.2.1 or Algorithm 3 in Section 3.2.2) are applied to solve (3.11) and to compute the optimal value $S_j(\boldsymbol{x}, t)$. Then, by (3.10), the solution $S(\boldsymbol{x}, t)$ is the minimum among these $m$ optimal values.

In addition, a minimizer $\boldsymbol{u}^*$ of (3.21) can be computed as

$$\boldsymbol{u}^* \in \bigcup_{r \in \mathcal{J}} \operatorname*{arg\,min}_{\boldsymbol{u} \in \prod_{i=1}^n [y_i - a_i t, y_i + b_i t]} \left\{ \sum_{i=1}^n S(y_i, t; u_i, a_i, b_i) + J_r\left((P^T)^{-1}\boldsymbol{u} + \boldsymbol{v}_0\right) \right\}, \quad (3.12)$$

where the set $\mathcal{J}$ is defined by:

$$\mathcal{J} := \operatorname*{arg\,min}_{j \in \{1, \ldots, m\}} S_j(\boldsymbol{x}, t),$$

with $S_j$ defined by (3.11). In other words, $\boldsymbol{u}^*$ can be computed using (3.12) by applying convex optimization methods to solve each of the $m$ subproblems, where the $j$-th subproblem is defined in (3.11). Finally, an optimal trajectory $\boldsymbol{\gamma}$ is computed using (3.23) as long as the minimizer $\boldsymbol{u}^*$ is obtained. Note that $\boldsymbol{\gamma}$ may not be unique since we are solving a nonconvex optimization problem and hence $\boldsymbol{u}^*$ is no longer necessarily unique.

Similarly, the min-plus technique can also be applied to the representation formulas (3.15) and (3.19) to solve the corresponding optimal control problems and HJ

PDEs with nonconvex initial condition $J$ of the form (3.9). To be specific, (3.15) and (3.19) are the one-dimensional and high-dimensional cases, respectively, of (3.21) when $P$ is the identity matrix and $\boldsymbol{v}_0$ is the zero vector. Therefore, in these cases, the $j$-th subproblem becomes

$$S_j(\boldsymbol{x},t) := \inf_{\boldsymbol{u}\in\prod_{i=1}^n [x_i-a_it,x_i+b_it]} \left\{ \sum_{i=1}^n S(x_i,t;u_i,a_i,b_i) + J_j\left(\boldsymbol{u}\right) \right\}, \tag{3.13}$$

and an optimal trajectory $\boldsymbol{\gamma}$ and the optimal value $S$ are computed using the minimizers and the minimal values of these subproblems, similarly to before. The details of the proposed algorithm for solving the optimal control problem (3.17) and the corresponding HJ PDE (3.18) are given in Algorithm 4. The more general problems in (3.1) and (3.2) can be solved by replacing the $j$-th subproblem in Algorithm 4 with (3.11) for each $j \in \{1,\dots,m\}$. Note that each subproblem is solved independently from each other, and hence, each subproblem can be solved in parallel and/or using different numerical methods, when convenient. The complexity of Algorithm 4 is the sum of the complexity of solving all $m$ subproblems.

In the following proposition, we show some error analysis for Algorithm 4, given the error of each subproblem. To be specific, if each subproblem converges, then the value function $\hat{S}$ converges pointwise and any cluster point of the numerical optimal trajectory $\hat{\boldsymbol{\gamma}}$ is an optimal trajectory in (3.17). Note that the convergence of the numerical optimal trajectories is not guaranteed, due to the non-uniqueness of the optimal trajectory $\boldsymbol{\gamma}$. Moreover, we prove error bounds for the output solution and output trajectory of Algorithm 4. With this error analysis, the convergence rate of any convergent subsequence is determined by the convergence rate of each of the $m$ subproblems. For the special case when each subproblem is solved using Algorithm 3, the convergence rate is given by the slowest convergence rate of each subproblem, which is discussed in Section 3.2.2.

**Algorithm 4:** An optimization algorithm for solving the optimal control problem (3.17) and the corresponding HJ PDE (3.18) with nonconvex initial cost $J$ of the form (3.9).

**Inputs :** Parameters $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, terminal position $\boldsymbol{x} \in \mathbb{R}^n$, time horizon $t > 0$, running time $s > 0$ of the trajectory, and the convex functions $J_1, \ldots, J_m$ in (3.9).

**Outputs:** An optimal trajectory $\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t)$ in the optimal control problem (3.17) and the solution value $\hat{S}(\boldsymbol{x}, t)$ to the corresponding HJ PDE (3.18) with nonconvex initial cost $J$ of the form (3.9).

**for** $j = 1, 2, \ldots, m$ **do**

  Numerically solve the $j$-th subproblem (3.13) using any appropriate method (e.g., the solver in Section 3.2.1 or Algorithm 3 in Section 3.2.2), and get the optimal trajectory $\hat{\boldsymbol{\gamma}}_j(s; \boldsymbol{x}, t)$ and the solution $\hat{S}_j(\boldsymbol{x}, t)$ to the problems (3.17) and (3.18) with initial cost $J_j$;

**end**

Compute the index $r$ by:

$$r \in \underset{j \in \{1,\ldots,m\}}{\arg\min}\, \hat{S}_j(\boldsymbol{x}, t). \tag{3.14}$$

Output an optimal trajectory $\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t)$ and the solution value $\hat{S}(\boldsymbol{x}, t)$ using

$$\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t) = \hat{\boldsymbol{\gamma}}_r(s; \boldsymbol{x}, t), \qquad \hat{S}(\boldsymbol{x}, t) = \hat{S}_r(\boldsymbol{x}, t) = \min_{j \in \{1,\ldots,m\}} \hat{S}_j(\boldsymbol{x}, t). \tag{3.15}$$

**Proposition 3.2.2.** *Let $J\colon \mathbb{R}^n \to \mathbb{R}$ be a function satisfying (3.9) for some convex functions $J_1, \dots, J_m\colon \mathbb{R}^n \to \mathbb{R}$ and $\boldsymbol{a}, \boldsymbol{b}$ be two vectors in $(0, +\infty)^n$. Let $\boldsymbol{x}$ be any vector in $\mathbb{R}^n$ and $t > 0$ be any scalar. Let $S$ be the function defined in (3.19) with initial condition $J$. Denote the output solution and trajectory of Algorithm 4 by $\hat{S}$ and $\hat{\boldsymbol{\gamma}}$, respectively. For the $j$-th subproblem, denote the analytical solution and the numerical solution by $S_j$ and $\hat{S}_j$, respectively, and denote the analytical optimal trajectory and the numerical trajectory by $\boldsymbol{\gamma}_j$ and $\hat{\boldsymbol{\gamma}}_j$, respectively. Assume there exists $\epsilon > 0$, such that there holds*

$$|S_j(\boldsymbol{x}, t) - \hat{S}_j(\boldsymbol{x}, t)| \leq \epsilon \quad \forall j \in \{1, \dots, m\}. \tag{3.16}$$

*Then, we have*

$$|S(\boldsymbol{x}, t) - \hat{S}(\boldsymbol{x}, t)| \leq \epsilon. \tag{3.17}$$

*Further assume $S_j(\boldsymbol{x}, t) > S(\boldsymbol{x}, t) + 2\epsilon$ for each index $j$ satisfying $S_j(\boldsymbol{x}, t) \neq S(\boldsymbol{x}, t)$. Then, there exists an optimal trajectory $\boldsymbol{\gamma}$ of the optimal control problem (3.17) with initial cost $J$, such that there holds*

$$\sup_{s\in[0,t]} \|\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t) - \boldsymbol{\gamma}(s; \boldsymbol{x}, t)\| \leq \sup_{s\in[0,t]} \|\hat{\boldsymbol{\gamma}}_r(s; \boldsymbol{x}, t) - \boldsymbol{\gamma}_r(s; \boldsymbol{x}, t)\|, \tag{3.18}$$

*where $r$ is the index in (3.14).*

*Proof.* The proof is provided in Section 3.5.2. □

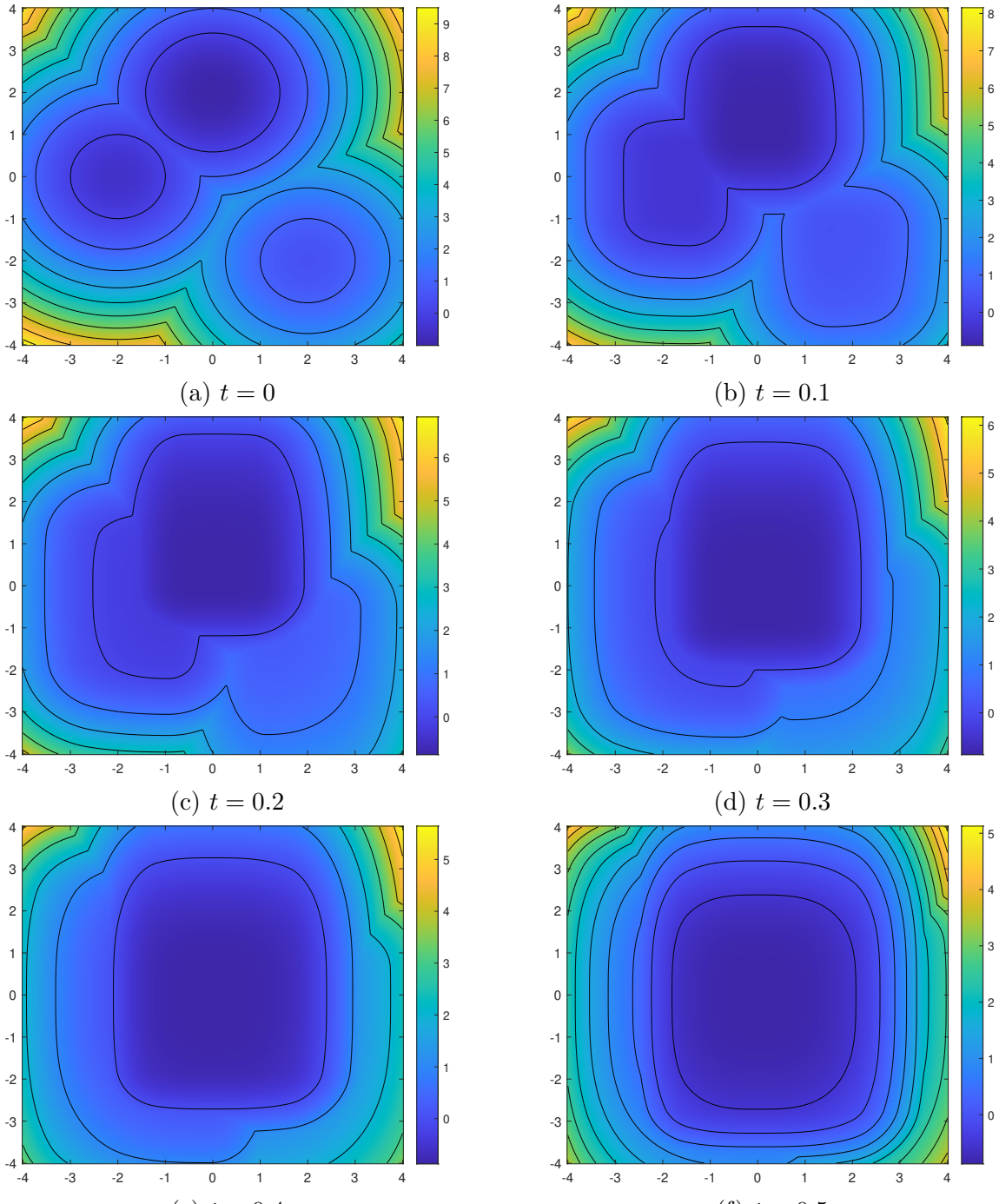
Now, we present a high-dimensional numerical example using nonconvex initial cost $J$ defined by:

$$J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} \left\{ \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j \right\}, \tag{3.19}$$

where $\boldsymbol{y}_1 = (-2, 0, \ldots, 0)$, $\boldsymbol{y}_2 = (2, -2, -1, 0, \ldots, 0)$, and $\boldsymbol{y}_3 = (0, 2, 0, \ldots, 0)$ are vectors in $\mathbb{R}^n$ and $\alpha_1 = -0.5$, $\alpha_2 = 0$, and $\alpha_3 = -1$ are scalars in $\mathbb{R}$. Recall that $\|\cdot\|$ denotes the $\ell^2$-norm in the Euclidean space $\mathbb{R}^n$. We also use the parameters $\boldsymbol{a}, \boldsymbol{b}$ defined in (3.4), i.e., we set $\boldsymbol{a} = (4, 6, 5, \ldots, 5)$ and $\boldsymbol{b} = (3, 9, 6, \ldots, 6)$. Note that each subproblem has quadratic initial cost and thus can be solved using the solver in Section 3.2.1. In other words, the solver proposed in Section 3.2.1 serves as the building block for Algorithm 4 in this example. Recall that the solver in Section 3.2.1 has complexity $\Theta(n)$. Therefore, Algorithm 4 has complexity $\Theta(mn)$ in this example, and hence, it overcomes the curse of dimensionality.

We solve the 10-dimensional problem (i.e., we set $n = 10$) and plot the solution $\hat{S}$ and the optimal trajectories $\hat{\boldsymbol{\gamma}}$ in Figures 3.5 and 3.6, respectively. Figure 3.5 depicts two-dimensional slices of the numerical solution $\hat{S}(\boldsymbol{x}, t)$, as computed using Algorithm 4, to HJ PDE (3.18) for different positions $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$ and different times $t$. In Figure 3.5(a), we can clearly see that the initial condition $J$ is not smooth at the interfaces of the quadratics $J_i$; e.g., there are obvious kinks near $(x_1, x_2) = (0, 0)$, $(-2, 2)$, $(0, -2)$, $(2.5, 0)$. In Figures 3.5(b)-(f), we see that, over time, the solution also evolves with several kinks. These kinks provide numerical validation that our algorithm does indeed provide the non-smooth viscosity solution to the corresponding HJ PDE. Overall, the solution appears to be continuous (but not necessarily differentiable) in $(x_1, x_2)$ at all times $t$, which is consistent with the results of Proposition 3.1.5.

Figure 3.6 depicts one-dimensional slices of an optimal trajectory $\hat{\boldsymbol{\gamma}}(s; \boldsymbol{x}, t)$ of the corresponding 10-dimensional optimal control problem for different terminal positions $\boldsymbol{x} = (x, -x, 0, \ldots, 0)$ and different time horizons $t$. We observe that the one-dimensional slices are piecewise linear and continuous in $s$, which is consistent with the properties of the formulas in (3.10), (3.11), (3.12), and (3.14). We also

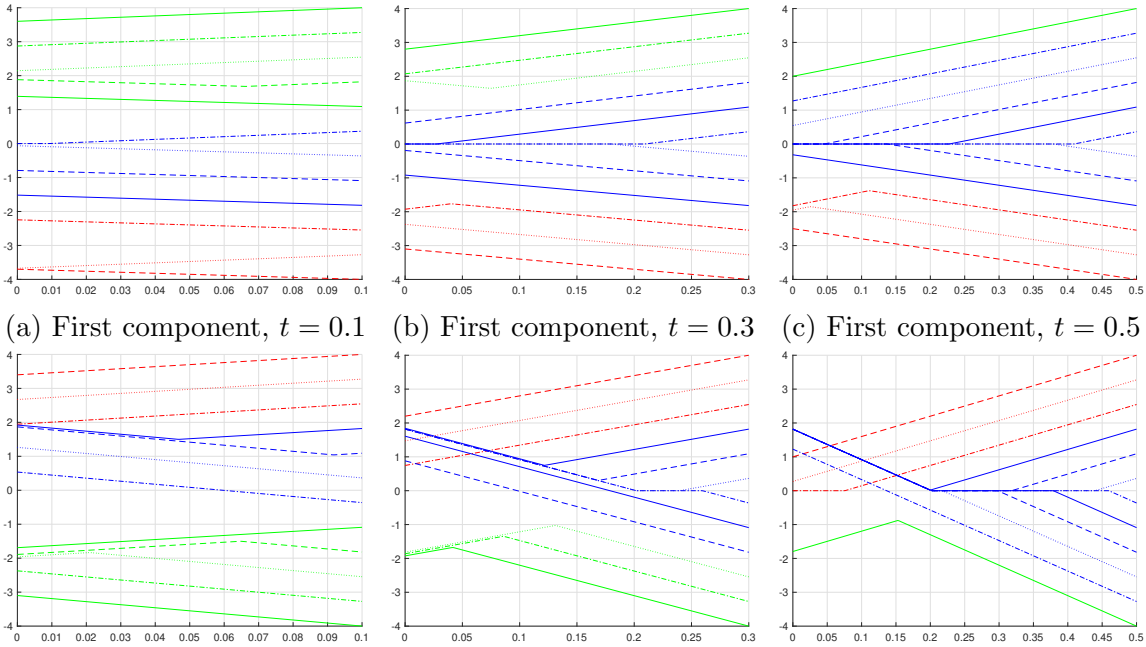


(a) $t = 0$ (b) $t = 0.1$

(c) $t = 0.2$ (d) $t = 0.3$

(e) $t = 0.4$ (f) $t = 0.5$

Figure 3.5: Evaluation of the solution $\hat{S}(\boldsymbol{x}, t)$ of the 10-dimensional HJ PDE (3.18) with $\boldsymbol{a} = (4, 6, 5, \ldots, 5)$, $\boldsymbol{b} = (3, 9, 6, \ldots, 6)$, and initial condition $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}}\{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j\}$, where $\boldsymbol{y}_1 = (-2, 0, \ldots, 0)$, $\boldsymbol{y}_2 = (2, -2, -1, 0, \ldots, 0)$, $\boldsymbol{y}_3 = (0, 2, 0, \ldots, 0)$, $\alpha_1 = -0.5$, $\alpha_2 = 0$, and $\alpha_3 = -1$, for $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$, where $(x_1, x_2) \in [-4, 4]^2$, and different times $t$. Plots for $t = 0$, 0.1, 0.2, 0.3, 0.4, and 0.5 are depicted in (a)-(f), respectively. Level lines are superimposed on the plots.



(a) First component, $t = 0.1$ (b) First component, $t = 0.3$ (c) First component, $t = 0.5$

(d) Second component, $t = 0.1$ (e) Second component, $t = 0.3$ (f) Second component, $t = 0.5$

Figure 3.6: Evaluation of an optimal trajectory $\hat{\boldsymbol{\gamma}}(s; (x, -x, 0, \ldots, 0), t)$ of the 10-dimensional optimal control problem (3.17) with $\boldsymbol{a} = (4, 6, 5, \ldots, 5)$, $\boldsymbol{b} = (3, 9, 6, \ldots, 6)$, and initial cost $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}}\{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}_j\|^2 + \alpha_j\}$, where $\boldsymbol{y}_1 = (-2, 0, \ldots, 0)$, $\boldsymbol{y}_2 = (2, -2, -1, 0, \ldots, 0)$, $\boldsymbol{y}_3 = (0, 2, 0, \ldots, 0)$, $\alpha_1 = -0.5$, $\alpha_2 = 0$, and $\alpha_3 = -1$, versus $s \in [0, t]$ for different terminal positions $(x, -x, 0, \ldots, 0)$ $(x \in [-4, 4])$ and different time horizons $t$. The color of the lines denotes which initial cost was used, i.e., $r \in \arg\min_{j\in\{1,2,3\}} \hat{S}_j(\boldsymbol{x}, t)$ for $r = 1, 2, 3$ corresponds to red, green, and blue, respectively. The different line markers simply help to differentiate between the different trajectories. Figures (a)-(c) depict the first component of the trajectory versus $s \in [0, t]$ with different time horizons $t$, while Figures (d)-(f) depict the second component of the trajectory versus $s \in [0, t]$ with different time horizons $t$. Plots for time horizons $t = 0.1$, 0.3, and 0.5 are depicted in Figures (a)/(d), (b)/(e), and (c)/(f), respectively. We note that because of our choice of $\boldsymbol{a} = (4, 6, 5, \ldots, 5)$ and $\boldsymbol{b} = (3, 9, 6, \ldots, 6)$, the piecewise slopes of our trajectories are not symmetric about 0.

note that in each subplot, all line segments with positive slope are parallel with slope $a_i$ (i.e. the $i$-th component of the trajectory has velocity $a_i$), while all lines segments with negative slope are parallel with slope $-b_i$ (i.e. the $i$-th component of the trajectory has velocity $-b_i$). As such, in any given subplot, the piecewise slopes of the trajectories are not symmetric about 0 due to our choice of $\boldsymbol{a}$ and $\boldsymbol{b}$. We also observe different patterns of trajectories depending on the terminal positions and time horizons.

| n | CPU time (s) | FPGA time (s) | Speedup |
|---|---|---|---|
| 4 | 1.7887e-07 | 1.334e-08 | 13.4085 |
| 8 | 4.5562e-07 | 2.667e-08 | 17.0836 |
| 12 | 1.3138e-06 | 4.001e-08 | 32.8370 |
| 16 | 2.1028e-06 | 5.334e-08 | 39.4226 |

Table 3.5: Comparison of the average time per call over $100,000$ runs for evaluating the solution of the HJ PDE (3.18) with initial condition $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}}\{\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}_j\|^2 + \alpha_j\}$, where $\boldsymbol{y}_1 = (-2, 0, \ldots, 0)$, $\boldsymbol{y}_2 = (2, -2, -1, 0, \ldots, 0)$, $\boldsymbol{y}_3 = (0, 2, 0, \ldots, 0)$, $\alpha_1 = -0.5$, $\alpha_2 = 0$, and $\alpha_3 = -1$, for various dimensions $n$ using a CPU implementation on a single Intel Core i7-1165G7 versus an FPGA implementation on a Xilinx Alveo U280 board with a frequency of 300 MHz.

| n | Latency (ns) | BRAM | DSPs | FFs | LUTs |
|---|---|---|---|---|---|
| 4 | 400,244 (1.334e06) | 0 (0%) | 2,541 (28%) | 278,904 (10%) | 168,376 (12%) |
| 8 | 800,264 (2.667e06) | 0 (0%) | 2,541 (28%) | 279,557 (10%) | 169,091 (12%) |
| 12 | 1,200,272 (4.001e06) | 0 (0%) | 2,541 (28%) | 280,337 (10%) | 169,759 (13%) |
| 16 | 1,600,288 (5.334e06) | 0 (0%) | 2,541 (28%) | 280,548 (10%) | 170,441 (13%) |

Table 3.6: FPGA resources and latencies in cycles and nanoseconds (ns) for evaluating the solution of the HJ PDE (3.18) with initial condition $J(\boldsymbol{x}) = \min_{j\in\{1,2,3\}} J_j(\boldsymbol{x}) = \min_{j\in\{1,2,3\}}\{\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}_j\|^2 + \alpha_j\}$, where $\boldsymbol{y}_1 = (-2, 0, \ldots, 0)$, $\boldsymbol{y}_2 = (2, -2, -1, 0, \ldots, 0)$, $\boldsymbol{y}_3 = (0, 2, 0, \ldots, 0)$, $\alpha_1 = -0.5$, $\alpha_2 = 0$, and $\alpha_3 = -1$, at $100,000$ points $(\boldsymbol{x}, t, \boldsymbol{y}) \in \mathbb{R}^n \times [0, \infty) \times \mathbb{R}^n$ for various dimensions $n$ using double precision floating points on a Xilinx Alveo U280 board with a frequency of 300 MHz.

In Table 3.5, we show the running time of a CPU and an FPGA implementation of Algorithm 4 for this example for different dimensions $n$. We measure the running time using the same method described in Section 3.2.1, and we use the solver and corresponding implementations from Section 3.2.1 in line 2 of Algorithm 4 to solve each of the $j$-th subproblems, $j = 1, 2, 3$. For this example, we design a high throughput

FPGA implementation with an II of 1 and that streams the points $(\boldsymbol{x}, t) \in \mathbb{R}^n \times [0, \infty)$ elementwise (i.e., our FPGA kernel takes input $(x_i, t) \in \mathbb{R} \times [0, \infty)$). Specifically, our FPGA implementation essentially replicates the high-throughput implementation of the numerical solver from Section 3.2.1 three times (once per $j$-th subproblem), where the three copies of the building block run in parallel. The outputs of the three building blocks are then combined using (3.14).

As such, in Table 3.5, we observe that the average runtimes for our FPGA implementation for this example are nearly identical to those for the FPGA implementation of the building block in Table 3.1. In contrast, the CPU implementation now has to parallelize computations for the $m = 3$ sub-problems in addition to its elementwise (i.e., for each $(x_i, t) \in \mathbb{R} \times [0, \infty)$, $i = 1, \ldots, n$) and pointwise (i.e., for various points $(\boldsymbol{x}, t) \in \mathbb{R}^n \times [0, \infty)$) parallelizations. As a result, the average CPU runtimes in Table 3.5 are almost three times slower than those for the CPU implementation of the building block in Table 3.1.

Now comparing the CPU and FPGA timing results for this example, in Table 3.5, we see that the CPU implementation takes less than $3 \times 10^{-6}$ seconds on average to compute the solution at one point for the 16-dimensional problem, which shows the efficiency of our proposed algorithm even in high dimensions. Meanwhile, our FPGA implementation takes less than $6 \times 10^{-8}$ seconds on average to compute the solution at one point for the 16-dimensional problem, for a speedup of about 40 compared to the CPU. These results highlight the promising performance boosts FPGAs are able to achieve over CPUs.

Table 3.6 shows the FPGA resources and latencies used by our FPGA implementation for this example for different dimensions $n$. We observe that, due to our use of elementwise streaming, the latency of our FPGA implementation scales linearly

in the dimension $n$, while the amount of FPGA resources used remains essentially constant in $n$. We also observe that the latencies in Table 3.6 are nearly identical to those for the FPGA implementation of the building block in Table 3.2. Meanwhile, the FPGA resources used in Table 3.6 are approximately 3 times greater than those used by the FPGA implementation of the building block in Table 3.2. These results are consistent with the fact that our FPGA implementation of the algorithm for this example consists of three copies of the building block from Section 3.2.1 running in parallel.

In Table 3.6, we also see that our design uses less than 30% of the FPGA resources available on the Xilinx Alveo U280 board. This means that we could either use a smaller (i.e., cheaper) FPGA to implement our numerical solver with similar performance as we report here or we could parallelize by simply implementing multiple, independent copies of our FPGA kernel to maximize usage of the FPGA board. In the latter case, we could achieve a further speedup of $\times 3$ (i.e., 1 copy of our FPGA kernel per each of the 3 chiplets, ensuring that no kernel requires crossing chiplets) for a total speedup of about 40 to 118 over the CPU depending on the dimension $n$.

## 3.3 Some proofs and technical lemmas for the analytical solutions

### 3.3.1 Technical lemmas for Section 3.1

**Lemma 3.3.1.** *Let $a, b, t$ be positive scalars and $x, u$ be real numbers satisfying $u - bt \leq x \leq u + at$. Let $S$ be the function defined in (3.7) and (3.13) and $[0, t] \ni s \mapsto$*

*$\gamma(s;x,t,u,a,b) \in \mathbb{R}$ be the trajectory defined in (3.10), (3.11), (3.12), and (3.14) for different cases. Then, there holds*

$$\int_0^t \frac{1}{2}\left(\gamma(s;x,t,u,a,b)\right)^2 ds = S(x,t;u,a,b). \tag{3.1}$$

*Proof.* If $(x,t,u) \in \Omega_1$ holds, we have

$$\begin{aligned}
\int_0^t \frac{1}{2}\left(\gamma(s;x,t,u,a,b)\right)^2 ds &= \int_0^{\frac{-x+u+at}{a+b}} \frac{1}{2}(u-bs)^2 ds + \int_{\frac{-x+u+at}{a+b}}^t \frac{1}{2}(as-at+x)^2 ds \\
&= -\frac{1}{6b}(u-bs)^3\Big|_0^{\frac{-x+u+at}{a+b}} + \frac{1}{6a}(as-at+x)^3\Big|_{\frac{-x+u+at}{a+b}}^t \\
&= -\left(\frac{1}{6b}+\frac{1}{6a}\right)\left(\frac{au+bx-abt}{a+b}\right)^3 + \frac{u^3}{6b} + \frac{x^3}{6a} \\
&= S(x,t;u,a,b).
\end{aligned}$$

If $(x,t,u) \in \Omega_2$ holds, we have

$$\begin{aligned}
\int_0^t \frac{1}{2}\left(\gamma(s;x,t,u,a,b)\right)^2 ds &= \int_0^{\frac{u}{b}} \frac{1}{2}(u-bs)^2 ds + \int_{t-\frac{x}{a}}^t \frac{1}{2}(as-at+x)^2 ds \\
&= -\frac{1}{6b}(u-bs)^3\Big|_0^{\frac{u}{b}} + \frac{1}{6a}(as-at+x)^3\Big|_{t-\frac{x}{a}}^t \\
&= \frac{u^3}{6b} + \frac{x^3}{6a} \\
&= S(x,t;u,a,b).
\end{aligned}$$

If $(x,t,u) \in \Omega_3$ holds, we have

$$\begin{aligned}\int_0^t \frac{1}{2}\left(\gamma(s;x,t,u,a,b)\right)^2 ds &= \int_0^{\frac{u}{b}} \frac{1}{2}(u-bs)^2 ds + \int_{t-\frac{-x}{b}}^t \frac{1}{2}(-bs+bt+x)^2 ds \\ &= -\frac{1}{6b}(u-bs)^3\Big|_0^{\frac{u}{b}} - \frac{1}{6b}(-bs+bt+x)^3\Big|_{t-\frac{-x}{b}}^t \\ &= \frac{u^3}{6b} - \frac{x^3}{6b} \\ &= S(x,t;u,a,b).\end{aligned}$$

If $u < 0$, we have

$$\begin{aligned}\int_0^t \frac{1}{2}\left(\gamma(s;x,t,u,a,b)\right)^2 ds &= \int_0^t \frac{1}{2}\left(\gamma(s;-x,t,-u,b,a)\right)^2 ds \\ &= S(-x,t;-u,b,a) = S(x,t;u,a,b).\end{aligned}$$

Therefore, (3.1) holds for any $(x,t,u) \in \mathbb{R}\times(0,+\infty)\times\mathbb{R}$ satisfying $u-bt \le x \le u+at$. □

**Lemma 3.3.2.** *Let $a, b$ be positive scalars. Let $S$ be the function defined in (3.7) and (3.13). Then, for any $x \in \mathbb{R}$, $t > 0$, the function $\mathbb{R} \ni u \mapsto S(x,t;u,a,b) \in \mathbb{R} \cup \{+\infty\}$ is strictly convex and twice continuously differentiable in its domain.*

*Proof.* In this proof, we regard the function $S(x,t;u,a,b)$ as a function of $u$ from its domain $[x-at, x+bt]$ to $\mathbb{R}$, and we use its derivative to mean the derivative of $S$ with respect to $u$, by default. To prove the statement, we need to prove that $S$ is twice continuously differentiable and that the second-order derivative is positive almost everywhere in the domain. We consider the following cases.

First, assume $x \ge at$ holds. After some computation, the function $u \mapsto$

$S(x,t;u,a,b)$ can be written as follows:

$$S(x,t;u,a,b) = \frac{u^3}{6b} + \frac{x^3}{6a} - \left(\frac{1}{6a} + \frac{1}{6b}\right)\left(\frac{au+bx-abt}{a+b}\right)^3 \quad \forall u \in [x-at, x+bt],$$

which is twice continuously differentiable. The second-order derivative is given by

$$\begin{aligned}\frac{\partial^2 S(x,t;u,a,b)}{\partial u^2} &= \frac{u}{b} - \frac{a}{b(a+b)^2}(au+bx-abt) = \frac{(b^2+2ab)u - ab(x-at)}{b(a+b)^2} \\ &\geq \frac{(b^2+ab)u}{b(a+b)^2} \geq 0,\end{aligned} \tag{3.2}$$

where the first and second inequalities hold since we have $u \geq x-at \geq 0$. Moreover, the second inequality becomes equality if and only if $u$ is zero. In other words, the second-order derivative in (3.2) is positive almost everywhere, and hence, the conclusion holds in this case.

Next, assume that $x$ is a point in $[0, at)$. In this case, the function $u \mapsto S(x,t;u,a,b)$ can be written as follows:

$$S(x,t;u,a,b) = \begin{cases} -\dfrac{u^3}{6a} + \dfrac{x^3}{6a} & x-at \leq u < 0, \\ \dfrac{u^3}{6b} + \dfrac{x^3}{6a} & 0 \leq u < bt - \dfrac{bx}{a}, \\ \dfrac{u^3}{6b} + \dfrac{x^3}{6a} - \left(\dfrac{1}{6a} + \dfrac{1}{6b}\right)\left(\dfrac{au+bx-abt}{a+b}\right)^3 & bt - \dfrac{bx}{a} \leq u \leq x+bt. \end{cases}$$

It is straightforward to check that this function is twice continuously differentiable and that the second-order derivative reads:

$$\frac{\partial^2 S(x,t;u,a,b)}{\partial u^2} = \begin{cases} -\dfrac{u}{a} & x-at < u < 0, \\ \dfrac{u}{b} & 0 \leq u < bt - \dfrac{bx}{a}, \\ \dfrac{(b^2+2ab)u - ab(x-at)}{b(a+b)^2} & bt - \dfrac{bx}{a} \leq u < x+bt, \end{cases} \tag{3.3}$$

where the first line is positive since $u < 0$ holds in the first line, the second line is positive almost everywhere since $u > 0$ holds almost everywhere in the second line, and the third line is positive since the inequalities in (3.2) also hold according to the condition on $u$ (there holds $u \geq bt - \frac{bx}{a} > 0 > x - at$). Therefore, the conclusion follows in this case.

Finally, we consider the case when $x < 0$. By definition, we have that $S(x, t; u, a, b) = S(-x, t; -u, b, a)$, where the right-hand side is twice continuously differentiable and whose second-order derivative with respect to $-u$ is positive almost everywhere by the same argument above. Therefore, the function $S(x, t; u, a, b)$ is also strictly convex and twice continuously differentiable with respect to $u$, and the conclusion holds. □

### 3.3.2 Proof of Lemma 3.1.1

In this proof, whenever there is no ambiguity, we write $\gamma(s)$ instead of $\gamma(s; x, t, u, a, b)$. We assume $u \geq 0$. For the case where $u$ is negative, the proof is similar, so we omit it here. Let $s \mapsto x(s)$ be an arbitrary feasible trajectory. In other words, $s \mapsto x(s)$ is an absolutely continuous function satisfying $x(0) = u$, $x(t) = x$, and $\dot{x}(s) \in [-b, a]$ for all $s \in (0, t)$. Our goal is to prove $\int_0^t \frac{\gamma(s)^2}{2} ds \leq \int_0^t \frac{x(s)^2}{2} ds$. For this, it suffices to prove that $|\gamma(s)| \leq |x(s)|$ holds for all $s \in [0, t]$.

First, assume $(x, t, u) \in \Omega_1$. If $0 \leq s < \frac{-x+u+at}{a+b}$ holds, then we have that

$$x(s) = x(0) + \int_0^s \dot{x}(\tau) d\tau \geq u - bs = \gamma(s) \geq 0, \tag{3.4}$$

where the first inequality holds since we have $\dot{x}(\tau) \geq -b$ for all $\tau \in [0, t]$ and $x(0) = u$. If $\frac{-x+u+at}{a+b} \leq s \leq t$ holds, we obtain that

$$x(s) = x - \int_s^t \dot{x}(\tau) d\tau \geq x - a(t-s) = \gamma(s) \geq 0, \tag{3.5}$$

where the first inequality holds since we have $\dot{x}(\tau) \leq a$ for all $\tau \in [0, t]$. As a result, we have shown that $|\gamma(s)| \leq |x(s)|$ for all $s \in [0, t]$.

Next, we consider the case where $(x, t, u) \in \Omega_2$. Note that (3.4) and (3.5) still hold for $s \in [0, \frac{u}{b})$ and $s \in [t - \frac{x}{a}, t]$, respectively. For $s \in [\frac{u}{b}, t - \frac{x}{a})$, we have that

$$|\gamma(s)| = 0 \leq |x(s)|. \tag{3.6}$$

As a result, $|\gamma(s)| \leq |x(s)|$ holds for all $s \in [0, t]$.

Now, we assume $(x, t, u) \in \Omega_3$. Note that (3.4) and (3.6) still hold for $s \in [0, \frac{u}{b})$ and $s \in [\frac{u}{b}, t - \frac{|x|}{b})$, respectively. For $s \in [t - \frac{|x|}{b}, t]$, we have that

$$x(s) = x - \int_s^t \dot{x}(\tau) d\tau \leq x + b(t-s) = \gamma(s) \leq 0,$$

where the first inequality holds since we have $-\dot{x}(\tau) \leq b$ for all $\tau \in [0, t]$. As a result, we conclude that $|\gamma(s)| \leq |x(s)|$ holds for all $s \in [0, t]$.

Therefore, $\gamma$ is an optimal trajectory. Moreover, it is the unique optimal trajectory since the optimal control problem (3.6) is a convex optimization problem with a strictly convex objective function. Finally, by Lemma 3.3.1 (see Section 3.3), the cost of the trajectory $\gamma$ equals $S(x, t; u, a, b)$, and hence, $S(x, t; u, a, b)$ is the optimal value in the problem (3.6). □

### 3.3.3 Proof of Proposition 3.1.4

In this proof, whenever there is no ambiguity, we use the notation $\boldsymbol{\gamma}(s)$ instead of $\boldsymbol{\gamma}(s;\boldsymbol{x},t)$. Since $J$ is lower semi-continuous and the domain of the optimization problem in (3.19) is $\prod_{i=1}^{n}[x_i-a_it,x_i+b_it]$, which is compact, the minimizer $\boldsymbol{u}^*$ exists (see [119, Theorem 1.9]). Moreover, each component on the right-hand side of (3.20) is well-defined due to the constraints on $\boldsymbol{u}^*$, i.e., that $u_i^* \in [x_i - a_it, x_i + b_it]$ (see the discussion below (3.20)). Thus, $\boldsymbol{\gamma}$ is well-defined. Note that if $\boldsymbol{u}^*$ is not unique, we will prove that any arbitrary minimizer $\boldsymbol{u}^*$ of (3.19) corresponds to an optimal trajectory $\boldsymbol{\gamma}$.

We prove by contradiction that $\boldsymbol{\gamma}$ is an optimal trajectory. Assume $\boldsymbol{\gamma}$ is not an optimal trajectory. Then, there exists another trajectory $\tilde{\boldsymbol{\gamma}}$ satisfying

$$\tilde{\boldsymbol{\gamma}}(t) = \boldsymbol{x}, \qquad \dot{\tilde{\boldsymbol{\gamma}}}(s) \in \prod_{i=1}^{n}[-b_i, a_i] \quad \forall\, s \in (0,t), \tag{3.7}$$

and

$$J(\tilde{\boldsymbol{\gamma}}(0)) + \sum_{i=1}^{n}\int_0^t \frac{\tilde{\gamma}_i(s)^2}{2}ds < J(\boldsymbol{\gamma}(0)) + \sum_{i=1}^{n}\int_0^t \frac{\gamma_i(s)^2}{2}ds, \tag{3.8}$$

where $\gamma_i$ and $\tilde{\gamma}_i$ denote the $i$-th components of $\boldsymbol{\gamma}$ and $\tilde{\boldsymbol{\gamma}}$, respectively. By Lemma 3.1.1, we have that

$$\int_0^t \frac{\gamma(s;x_i,t,\tilde{\gamma}_i(0),a_i,b_i)^2}{2}ds \le \int_0^t \frac{\tilde{\gamma}_i(s)^2}{2}ds, \quad \forall i \in \{1,\dots,n\}, \tag{3.9}$$

where $\gamma(s;x_i,t,\tilde{\gamma}_i(0),a_i,b_i)$ is the one-dimensional optimal trajectory of the problem (3.6) with parameters $x = x_i$, $a = a_i$, $b = b_i$, and $u = \tilde{\gamma}_i(0)$ and is defined by (3.10), (3.11), (3.12), and (3.14) for different cases. Note that each $\gamma(s;x_i,t,\tilde{\gamma}_i(0),a_i,b_i)$ is well-defined since we have $\tilde{\gamma}_i(0) - b_it \le x_i \le \tilde{\gamma}_i(0) + a_it$

by (3.7). Moreover, by Lemma 3.3.1 (in Section 3.3), we have that

$$\begin{aligned}\int_0^t \frac{\gamma(s; x_i, t, \tilde{\gamma}_i(0), a_i, b_i)^2}{2} ds &= S(x_i, t; \tilde{\gamma}_i(0), a_i, b_i), \\ \int_0^t \frac{\gamma_i(s)^2}{2} &= S(x_i, t; \gamma_i(0), a_i, b_i),\end{aligned} \tag{3.10}$$

for each $i \in \{1, \dots, n\}$. Then, by (3.8), (3.9), and (3.10), we have

$$\begin{aligned}&J(\tilde{\boldsymbol{\gamma}}(0)) + \sum_{i=1}^n S(x_i, t; \tilde{\gamma}_i(0), a_i, b_i) \\ &= J(\tilde{\boldsymbol{\gamma}}(0)) + \sum_{i=1}^n \int_0^t \frac{\gamma(s; x_i, t, \tilde{\gamma}_i(0), a_i, b_i)^2}{2} ds \\ &\leq J(\tilde{\boldsymbol{\gamma}}(0)) + \sum_{i=1}^n \int_0^t \frac{\tilde{\gamma}_i(s)^2}{2} ds \\ &< J(\boldsymbol{\gamma}(0)) + \sum_{i=1}^n \int_0^t \frac{\gamma_i(s)^2}{2} ds \\ &= J(\boldsymbol{\gamma}(0)) + \sum_{i=1}^n S(x_i, t; \gamma_i(0), a_i, b_i) \\ &= J(\boldsymbol{u}^*) + \sum_{i=1}^n S(x_i, t; u_i^*, a_i, b_i) \\ &= \inf_{\boldsymbol{u} \in \prod_{i=1}^n [x_i - a_i t, x_i + b_i t]} \left\{ \sum_{i=1}^n S(x_i, t; u_i, a_i, b_i) + J(\boldsymbol{u}) \right\} \\ &= S(\boldsymbol{x}, t),\end{aligned} \tag{3.11}$$

where the third equality holds since, by definition of $\boldsymbol{\gamma}$, we have that $\boldsymbol{\gamma}(0) = \boldsymbol{u}^*$ and the fourth equality holds by definition of $\boldsymbol{u}^*$. Note that by (3.7), we have that $\tilde{\boldsymbol{\gamma}}(0) \in \prod_{i=1}^n [x_i - a_i t, x_i + b_i t]$, and hence, $\tilde{\boldsymbol{\gamma}}(0)$ satisfies the constraint in the minimization problem in (3.11). Thus, the strict inequality in (3.11) yields a contradiction, and we conclude that $\boldsymbol{\gamma}$ is an optimal trajectory, whose cost equals $S(\boldsymbol{x}, t)$ by (3.11). Moreover, when $J$ is convex, the optimal control problem (3.17) is a convex optimization problem with a strictly convex objective function, which implies the uniqueness of the optimal trajectory in this case. □

### 3.3.4 Proof of Proposition 3.1.6

Fix $\boldsymbol{x} \in \mathbb{R}^n$ and $t > 0$. Define $\tilde{J}\colon \mathbb{R}^n \to \mathbb{R}$ by:

$$\tilde{J}(\boldsymbol{z}) := J\left((P^T)^{-1}\boldsymbol{z} + \boldsymbol{v}_0\right) \quad \forall\, \boldsymbol{z} \in \mathbb{R}^n. \tag{3.12}$$

Let $\boldsymbol{x}\colon [0,t] \to \mathbb{R}^n$ be a feasible trajectory in problem (3.1), i.e., let $\boldsymbol{x}$ satisfy the constraints in problem (3.1), and let $\boldsymbol{y}(s) = P^T(\boldsymbol{x}(s) - \boldsymbol{v}_0)$ for all $s \in [0,t]$. Then, by straightforward computation, we have

$$\begin{aligned}
\dot{\boldsymbol{y}}(s) &= P^T\dot{\boldsymbol{x}}(s) \in \prod_{i=1}^{n}[-b_i, a_i], \quad \forall s \in (0,t),\\
\boldsymbol{y}(t) &= P^T(\boldsymbol{x}(t) - \boldsymbol{v}_0) = P^T(\boldsymbol{x} - \boldsymbol{v}_0) = \boldsymbol{y},
\end{aligned}$$

where $\boldsymbol{y}$ is the vector defined in (3.22). Therefore, $\boldsymbol{y}(\cdot)$ is a feasible trajectory for the following optimal control problem:

$$\min\left\{\int_0^t \frac{1}{2}\|\boldsymbol{y}(s)\|^2 ds + \tilde{J}(\boldsymbol{y}(0))\colon \dot{\boldsymbol{y}}(s) \in \prod_{i=1}^{n}[-b_i, a_i]\ \forall s \in (0,t),\ \boldsymbol{y}(t) = \boldsymbol{y}\right\}. \tag{3.13}$$

Moreover, by some computation involving the definitions of $M$ and $\tilde{J}$, the cost for the trajectory $\boldsymbol{x}(\cdot)$ in problem (3.1) equals

$$\begin{aligned}
&\int_0^t \frac{1}{2}\|\boldsymbol{x}(s) - \boldsymbol{v}_0\|_M^2 ds + J(\boldsymbol{x}(0))\\
&= \int_0^t \frac{1}{2}\|(P^T)^{-1}\boldsymbol{y}(s)\|_M^2 ds + J\left((P^T)^{-1}\boldsymbol{y}(0) + \boldsymbol{v}_0\right)\\
&= \int_0^t \frac{1}{2}\|\boldsymbol{y}(s)\|^2 ds + \tilde{J}(\boldsymbol{y}(0)),
\end{aligned}$$

which equals the cost of $\boldsymbol{y}(\cdot)$ in problem (3.13). Similarly, if $\boldsymbol{y}(\cdot)$ is a feasible trajectory in (3.13), then $s \mapsto (P^T)^{-1}\boldsymbol{y}(s) + \boldsymbol{v}_0$ is a feasible trajectory in (3.1) whose

cost equals the cost of $\boldsymbol{y}(\cdot)$ in (3.13). Therefore, the two optimal control problems (3.1) and (3.13) are equivalent to each other. By Proposition 3.1.4, the trajectory $s \mapsto \boldsymbol{y}^*(s) := (\gamma(s; y_1, t, u_1^*, a_1, b_1), \ldots, \gamma(s; y_n, t, u_n^*, a_n, b_n))$ is an optimal trajectory for problem (3.13), whose optimal value equals $S(\boldsymbol{x}, t)$ in (3.21). Hence, the corresponding trajectory

$$\begin{aligned} s \mapsto & (P^T)^{-1}\boldsymbol{y}^*(s) + \boldsymbol{v}_0 \\ = & (P^T)^{-1}\left(\gamma(s; y_1, t, u_1^*, a_1, b_1), \ldots, \gamma(s; y_n, t, u_n^*, a_n, b_n)\right) + \boldsymbol{v}_0 \\ = & \boldsymbol{\gamma}(s; \boldsymbol{x}, t) \end{aligned}$$

is an optimal trajectory for problem (3.1), whose optimal value also equals $S(\boldsymbol{x}, t)$.

Furthermore, if $J$ is convex, then problem (3.1) is a convex optimization problem with a strictly convex objective function since the matrix $M$ is positive definite. Thus, if $J$ is convex, the minimizer is unique and the unique optimal trajectory is $s \mapsto \boldsymbol{\gamma}(s; \boldsymbol{x}, t)$. □

### 3.3.5 Proof of Proposition 3.1.7

We prove this proposition using [8, Theorem III.3.17]. We write the optimal control problem (3.1) in the standard form in [8, Chapter III], which reads:

$$\inf \left\{ \int_0^t \ell(\boldsymbol{x}(s), \boldsymbol{\alpha}(s)) ds + J(\boldsymbol{x}(0)) \right\}, \tag{3.14}$$

subject to the constraint that $\boldsymbol{\alpha}(\cdot)\colon [0,t] \to A$ is a measurable function and $\boldsymbol{x}(\cdot)\colon [0,t] \to \mathbb{R}^n$ is an absolutely continuous function satisfying the following ODE:

$$\begin{cases} \dot{\boldsymbol{x}}(s) = f(\boldsymbol{x}(s), \boldsymbol{\alpha}(s)) & s \in (0,t), \\ \boldsymbol{x}(t) = \boldsymbol{x}. \end{cases}$$

In our case, the set $A$, the source term $f\colon \mathbb{R}^n \times A \to \mathbb{R}^n$, and the running cost $\ell\colon \mathbb{R}^n \times A \to \mathbb{R}$ are given by:

$$A = \prod_{i=1}^{n} [-b_i, a_i] \subset \mathbb{R}^n, \quad f(\boldsymbol{x}, \boldsymbol{\alpha}) = (P^T)^{-1}\boldsymbol{\alpha}, \quad \ell(\boldsymbol{x}, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{v}_0\|_M^2.$$

In [8, Chapter III], it is shown that the optimal control problem (3.14) corresponds to the following HJ PDE:

$$\begin{cases} \dfrac{\partial S}{\partial t}(\boldsymbol{x}, t) + H(\boldsymbol{x}, \nabla_{\boldsymbol{x}} S(\boldsymbol{x}, t)) = 0 & \boldsymbol{x} \in \mathbb{R}^n, t \in (0, +\infty), \\ S(\boldsymbol{x}, 0) = J(\boldsymbol{x}) & \boldsymbol{x} \in \mathbb{R}^n, \end{cases} \tag{3.15}$$

where $J$ is the initial condition given by the initial cost in (3.14) and $H$ is the Hamiltonian given by $A, f, \ell$ as follows:

$$H(\boldsymbol{x}, \boldsymbol{p}) = \sup_{\boldsymbol{\alpha} \in A} \{\langle f(\boldsymbol{x}, \boldsymbol{\alpha}), \boldsymbol{p} \rangle - \ell(\boldsymbol{x}, \boldsymbol{\alpha})\},$$

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product in $\mathbb{R}^n$. In our case, the Hamiltonian $H$ is

$$\begin{aligned} H(\boldsymbol{x}, \boldsymbol{p}) &= \sup_{\boldsymbol{\alpha} \in A} \{ \langle f(\boldsymbol{x}, \boldsymbol{\alpha}), \boldsymbol{p} \rangle - \ell(\boldsymbol{x}, \boldsymbol{\alpha}) \} \\ &= \sup_{\boldsymbol{\alpha} \in \prod_{i=1}^n [-b_i, a_i]} \left\{ \langle (P^T)^{-1} \boldsymbol{\alpha}, \boldsymbol{p} \rangle - \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{v}_0\|_M^2 \right\} \\ &= \sup_{\boldsymbol{\alpha} \in \prod_{i=1}^n [-b_i, a_i]} \langle \boldsymbol{\alpha}, P^{-1} \boldsymbol{p} \rangle - \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{v}_0\|_M^2 \\ &= \sum_{i=1}^n \sup_{\alpha_i \in [-b_i, a_i]} \alpha_i (P^{-1} \boldsymbol{p})_i - \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{v}_0\|_M^2 \\ &=: h(\boldsymbol{p}) - \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{v}_0\|_M^2, \end{aligned}$$

where, in the last line, we define the function $h \colon \mathbb{R}^n \to \mathbb{R}$ by:

$$h(\boldsymbol{p}) := \sum_{i=1}^n \sup_{\alpha_i \in [-b_i, a_i]} \alpha_i (P^{-1} \boldsymbol{p})_i \quad \forall \boldsymbol{p} \in \mathbb{R}^n.$$

By definition, $h$ is a non-negative convex 1-homogeneous function, and hence, it is uniquely determined by its sublevel set $\{\boldsymbol{p} \in \mathbb{R}^n \colon h(\boldsymbol{p}) \leq 1\}$, which is computed as follows:

$$\begin{aligned} &\{\boldsymbol{p} \in \mathbb{R}^n \colon h(\boldsymbol{p}) \leq 1\} \\ &= \{\boldsymbol{p} \in \mathbb{R}^n \colon \exists \boldsymbol{\beta} \in \Delta_n \text{ s.t. } \alpha_i (P^{-1} \boldsymbol{p})_i \leq \beta_i \, \forall \alpha_i \in [-b_i, a_i] \, \forall i \in \{1, \ldots, n\}\} \\ &= \left\{ \boldsymbol{p} \in \mathbb{R}^n \colon \exists \boldsymbol{\beta} \in \Delta_n \text{ s.t. } (P^{-1} \boldsymbol{p})_i \in \beta_i \left[ -\frac{1}{b_i}, \frac{1}{a_i} \right] \, \forall i \in \{1, \ldots, n\} \right\}, \end{aligned}$$

where $\Delta_n$ denotes the standard simplex set defined by:

$$\Delta_n := \left\{ (\beta_1, \ldots, \beta_n) \in [0, 1]^n \colon \sum_{i=1}^n \beta_i = 1 \right\}.$$

Therefore, $h(\boldsymbol{p}) \leq 1$ holds if and only if there exists $\boldsymbol{\beta} \in \Delta_n$, such that

$$\boldsymbol{p} = P(P^{-1}\boldsymbol{p}) = \sum_{i=1}^{n} (P^{-1}\boldsymbol{p})_i \boldsymbol{u}_i \in \sum_{i=1}^{n} \beta_i \mathrm{co} \left\{ -\frac{\boldsymbol{u}_i}{b_i}, \frac{\boldsymbol{u}_i}{a_i} \right\}.$$

Thus, we have

$$\{\boldsymbol{p} \in \mathbb{R}^n \colon h(\boldsymbol{p}) \leq 1\} = \mathrm{co} \left( \bigcup_{i=1}^{n} \left\{ \frac{\boldsymbol{u}_i}{a_i}, -\frac{\boldsymbol{u}_i}{b_i} \right\} \right) = \{\boldsymbol{x} \in \mathbb{R}^n \colon K(\boldsymbol{x}) \leq 1\},$$

which implies $h = K$, and hence, the corresponding HJ PDE (3.15) is the HJ PDE in (3.2).

Now, we apply [8, Theorem III.3.17] to prove the conclusion. If the assumptions are satisfied, then [8, Theorem III.3.17] implies that the value function in the optimal control problem (3.14) (which is (3.1) in our case) is the unique viscosity solution to the corresponding HJ PDE (3.15) (which is (3.2) in our case). Our goal is to check the assumptions of [8, Theorem III.3.17], which include:

($A_0$) The set $A$ is a topological space, and the function $f \colon \mathbb{R}^n \times A \to \mathbb{R}^n$ is continuous.

($A_1$) The function $f$ is bounded on $B_R(\mathbb{R}^n) \times A$ for all $R > 0$. Here and after, $B_R(\mathbb{R}^n)$ denotes the closed ball in $\mathbb{R}^n$ centered at zero with radius $R$.

($A_2$) There exists some positive constant $L_R$ depending on $R$, such that there holds $\|f(\boldsymbol{y}, \boldsymbol{\alpha}) - f(\boldsymbol{x}, \boldsymbol{\alpha})\| \leq L_R \|\boldsymbol{x} - \boldsymbol{y}\|$ for all $\boldsymbol{x}, \boldsymbol{y} \in B_R(\mathbb{R}^n)$, $\boldsymbol{\alpha} \in A$, and $R > 0$.

(3.27) There exists some positive constant $K$, such that $\|f(\boldsymbol{x}, \boldsymbol{\alpha})\| \leq K(\|\boldsymbol{x}\| + 1)$ holds for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{\alpha} \in A$.

(3.40) For all $R > 0$, $\ell$ is continuous and bounded on $B_R(\mathbb{R}^n) \times A$ and there exists some positive constant $L_R$ depending on $R$, such that $|\ell(\boldsymbol{x}, \boldsymbol{\alpha}) - \ell(\boldsymbol{y}, \boldsymbol{\alpha})| \leq L_R \|\boldsymbol{x} - \boldsymbol{y}\|$

holds for all $\boldsymbol{x}, \boldsymbol{y} \in B_R(\mathbb{R}^n)$ and $\boldsymbol{\alpha} \in A$.

The assumption $(A_0)$ is satisfied by definition. $(A_1)$, $(A_2)$, and (3.27) are satisfied because the set $A$ is compact and the function $f$ does not depend on $\boldsymbol{x}$. Since $\ell$ does not depend on $\boldsymbol{\alpha}$ and $\ell$ is continuous by definition, the function $\ell$ is bounded in $B_R(\mathbb{R}^n) \times A$ for all $R > 0$. By straightforward computation, for all $R > 0$, we have that

$$|\ell(\boldsymbol{x}, \boldsymbol{\alpha}) - \ell(\boldsymbol{y}, \boldsymbol{\alpha})| = \frac{1}{2} \left| (\boldsymbol{x} + \boldsymbol{y} - 2\boldsymbol{v}_0)^T M (\boldsymbol{x} - \boldsymbol{y}) \right| \leq \|M\|(R + \|\boldsymbol{v}_0\|)\|\boldsymbol{x} - \boldsymbol{y}\|,$$

for all $\boldsymbol{x}, \boldsymbol{y} \in B_R(\mathbb{R}^n)$ and $\boldsymbol{\alpha} \in A$. Hence, the assumptions of [8, Theorem III.3.17] are all satisfied. As a result, the value function in (3.1) is the unique continuous viscosity solution to the HJ PDE (3.2). According to Proposition 3.1.6, $S$ defined by (3.21) is the value function of the problem (3.1). Therefore, $S$ is the unique continuous viscosity solution to the HJ PDE (3.2). □

## 3.4 Some computations for the numerical implementation

### 3.4.1 A numerical method for computing the building block

Here, we discuss how to compute the proximal point of the function $\mathbb{R} \ni u \mapsto \frac{1}{\lambda} S(x, t; u, a, b) \in \mathbb{R} \cup \{+\infty\}$, i.e., how to solve the following convex optimization

problem:

$$\begin{aligned} u^* &= \arg\min_{u\in\mathbb{R}} \left\{ S(x,t;u,a,b) + \frac{\lambda}{2}(u-y)^2 \right\} \\ &= \arg\min_{u\in[x-at,x+bt]} \left\{ S(x,t;u,a,b) + \frac{\lambda}{2}(u-y)^2 \right\}, \end{aligned} \tag{3.1}$$

for any $\lambda, t, a, b > 0$, and $x, y \in \mathbb{R}$. We consider the following two cases for the variable $u$.

If $u \geq 0$, after some computation, the objective function in (3.1) can be written as

$$\begin{aligned} F(u;x,t,a,b) &:= S(x,t;u,a,b) + \frac{\lambda}{2}(u-y)^2 \\ &= \begin{cases} \dfrac{u^3}{6b} + \dfrac{x^3}{6a} - \left(\dfrac{1}{6a} + \dfrac{1}{6b}\right)\left(\dfrac{au+bx-abt}{a+b}\right)^3 + \dfrac{\lambda}{2}(u-y)^2 & u \in \Omega_1(x,t,a,b), \\ \dfrac{u^3}{6b} + \dfrac{x^3}{6a} + \dfrac{\lambda}{2}(u-y)^2 & u \in \Omega_2(x,t,a,b), \\ \dfrac{u^3}{6b} - \dfrac{x^3}{6b} + \dfrac{\lambda}{2}(u-y)^2 & u \in \Omega_3(x,t,a,b), \\ +\infty & \text{otherwise}, \end{cases} \end{aligned}$$

where the three regions $\Omega_1(x,t,a,b), \Omega_2(x,t,a,b), \Omega_3(x,t,a,b) \subset [0,+\infty)$ are defined by:

$$\begin{aligned} \Omega_1(x,t,a,b) &:= \{u \in (bt,+\infty) \colon x - at \leq u \leq x + bt\} \\ &\qquad \bigcup \left\{ u \in [0,bt] \colon u \geq x - at,\ u \geq bt - \frac{bx}{a} \right\}, \\ \Omega_2(x,t,a,b) &:= \begin{cases} \left[0, bt - \dfrac{bx}{a}\right) & x \geq 0, \\ \emptyset & x < 0, \end{cases} \\ \Omega_3(x,t,a,b) &:= \begin{cases} [0, x+bt] & x < 0, \\ \emptyset & x \geq 0. \end{cases} \end{aligned}$$

In this case, the derivative of $F$ with respect to $u$ is given by:

$$\begin{aligned}&\frac{\partial}{\partial u}F(u;x,t,a,b)\\&=\begin{cases}\dfrac{(2a+b)u^2-2a(x-at)u-b(x-at)^2}{2(a+b)^2}+\lambda(u-y) & u\in\Omega_1(x,t,a,b),\\ \dfrac{u^2}{2b}+\lambda(u-y) & u\in\Omega_2(x,t,a,b)\cup\Omega_3(x,t,a,b),\end{cases}\end{aligned}\tag{3.2}$$

and the second derivative of $F$ with respect to $u$ can be easily computed using (3.2) and (3.3), for different cases. To get possible candidates for the minimizer $u^*$ of $F$ in this case, we compute the roots of the functions in the two lines of (3.2) and select the roots where the second derivative of $F$ is non-negative. After some calculations, the candidates are given by $u_1$ and $u_2$, which are defined as follows:

$$\begin{aligned}u_1 &:= -\frac{\lambda(a+b)^2-a(x-at)}{2a+b}\\&\quad+\sqrt{\left(\frac{\lambda(a+b)^2-a(x-at)}{2a+b}\right)^2+\frac{b(x-at)^2}{2a+b}+\frac{2\lambda(a+b)^2y}{2a+b}},\\u_2 &:= -\lambda b+\sqrt{(\lambda b)^2+2\lambda by}.\end{aligned}\tag{3.3}$$

Note that $u_1$ and $u_2$ may be not well-defined if the term under the square root is negative, in which case the corresponding function does not provide a possible candidate for $u^*$. Therefore, we assign $u_i$ $(i=1,2)$ to be an arbitrary point in $[x-at, x+bt]$ if it is not well-defined.

If $u<0$, after some computation, the objective function in (3.1) can be written

as

$$
\begin{aligned}
&F(u;x,t,a,b) := S(x,t;u,a,b) + \frac{\lambda}{2}(u-y)^2 \\
&= S(-x,t;-u,b,a) + \frac{\lambda}{2}(u-y)^2 \\
&= \begin{cases}
-\dfrac{u^3}{6a} - \dfrac{x^3}{6b} + \left(\dfrac{1}{6a} + \dfrac{1}{6b}\right)\left(\dfrac{bu+ax+abt}{a+b}\right)^3 + \dfrac{\lambda}{2}(u-y)^2 & -u \in \Omega_1(-x,t,b,a), \\
-\dfrac{u^3}{6a} - \dfrac{x^3}{6b} + \dfrac{\lambda}{2}(u-y)^2 & -u \in \Omega_2(-x,t,b,a), \\
-\dfrac{u^3}{6a} + \dfrac{x^3}{6a} + \dfrac{\lambda}{2}(u-y)^2 & -u \in \Omega_3(-x,t,b,a), \\
+\infty & \text{otherwise.}
\end{cases}
\end{aligned}
$$

Thus, for $u < 0$, the derivative of $F$ with respect to $u$ is given by:

$$
\begin{aligned}
&\frac{\partial}{\partial u} F(u;x,t,a,b) \\
&= \begin{cases}
\dfrac{-(a+2b)u^2 + 2b(x+bt)u + a(x+bt)^2}{2(a+b)^2} + \lambda(u-y) & -u \in \Omega_1(-x,t,b,a), \\
-\dfrac{u^2}{2a} + \lambda(u-y) & -u \in \Omega_2(-x,t,b,a) \cup \Omega_3(-x,t,b,a).
\end{cases}
\end{aligned}
\tag{3.4}
$$

Similarly as in the first case, we take the roots of the two functions in (3.4), such that the second order derivative of $F$ is non-negative. These roots provide possible candidates for $u^*$. We denote these candidates by $u_1'$ and $u_2'$, which are defined by:

$$
\begin{aligned}
u_1' &:= \frac{\lambda(a+b)^2 + b(x+bt)}{a+2b} - \sqrt{\left(\frac{\lambda(a+b)^2 + b(x+bt)}{a+2b}\right)^2 + \frac{a(x+bt)^2}{a+2b} - \frac{2\lambda(a+b)^2 y}{a+2b}}, \\
u_2' &:= \lambda a - \sqrt{(\lambda a)^2 - 2\lambda a y}.
\end{aligned}
\tag{3.5}
$$

Similarly, if $u_1'$ or $u_2'$ is not well-defined, we set it to be any point in $[x-at, x+bt]$.

Note that the objective function $F$ is strictly convex and twice continuously

differentiable with respect to $u$ by Lemma 3.3.2. Then, by the first and second derivative tests, the minimizer $u^*$ in (3.1) is selected among the possible candidates $u_1, u_2, u_1', u_2'$ defined in (3.3) and (3.5), as well as the boundary points $x - at$ and $x + bt$. In other words, the minimizer $u^*$ satisfies

$$u^* = \underset{u \in \{u_1, u_2, u_1', u_2', x-at, x+bt\}}{\arg\min} F(u; x, t, a, b). \tag{3.6}$$

Numerically, we solve the optimization problem (3.1) by computing the six candidates $u_1, u_2, u_1', u_2', x-at, x+bt$ and comparing the objective function values at those points. Then, the minimizer $u^*$ is selected using (3.6). Therefore, the complexity of solving (3.1) is $\Theta(1)$.

### 3.4.2 An equivalent expression for $S$ and $\gamma$

Let $S$ be the function defined by (3.7) and (3.13), and let $\gamma$ be the function defined by (3.10), (3.11), (3.12), and (3.14) for different cases. Now, we present an equivalent expression for $S$ and $\gamma$, which is used in our numerical implementation.

By straightforward calculation, the function $S$ can equivalently be expressed as

$$\begin{aligned} &S(x, t; u, a, b) \\ &= \begin{cases} \max\{S_3(x, t; u, a, b), \min\{S_1(x, t; u, a, b), S_2(x, t; u, a, b)\}\} & \text{if } u \in [0, x + bt], \\ \max\{S_3(-x, t; -u, b, a), \min\{S_1(-x, t; -u, b, a), S_2(-x, t; -u, b, a)\}\} & \text{if } u \in [x - at, 0), \\ +\infty & \text{otherwise,} \end{cases} \end{aligned}$$

where $S_1, S_2$, and $S_3$ are the functions in the first, second, and third lines of (3.7), respectively.

Similarly, assuming $u \in [x - at, x + bt]$ holds, the function $\gamma$ can be expressed as

$$\gamma(s; x, t, u, a, b)$$
$$= \begin{cases} \max\{u - bs, a(s-t) + x, 0\} & \text{if } x \geq 0, 0 \leq u \leq x + bt, \\ \max\{u - bs, 0\} + \min\{-b(s-t) + x, 0\} & \text{if } x < 0, 0 \leq u \leq x + bt, \\ \min\{u + as, -b(s-t) + x, 0\} & \text{if } x < 0, x - at \leq u < 0, \\ \min\{u + as, 0\} + \max\{a(s-t) + x, 0\} & \text{if } x \geq 0, x - at \leq u < 0. \end{cases}$$

Compared to the definitions (3.7), (3.13), (3.10), (3.11), (3.12), and (3.14), these equivalent formulas involve less conditional branching, and hence, they are more favorable for the performance of our numerical implementation.

## 3.5 Proofs of convergence results in Section 3.2

In this section, we provide the proof of Proposition 3.2.1 in Section 3.5.1 and the proof of Proposition 3.2.2 in Section 3.5.2.

### 3.5.1 Proof of Proposition 3.2.1

Let $\boldsymbol{v}^N$, $\boldsymbol{d}^N$, and $\boldsymbol{u}^N$ be the corresponding vectors in the algorithm at the $N$-th iteration. Let $\boldsymbol{u}^*$ be the minimizer of the minimization problem in (3.19), which is unique since $J$ is convex and each $u_i \mapsto S(x_i, t; u_i, a, b)$ is strictly convex by Lemma 3.3.2. According to [43, Theorem 2.2] whose assumptions are proved using [43, Remark 2.2], both $\boldsymbol{v}^N$ and $\boldsymbol{d}^N$ converge to the point $\boldsymbol{u}^*$ as $N$ approaches infinity, and hence, $\boldsymbol{u}^N$ in Algorithm 3 also converges to $\boldsymbol{u}^*$. Since $J$ is a real-valued convex function, it is

continuous in $\mathbb{R}^n$ and we have

$$\lim_{N\to\infty} J(\boldsymbol{u}^N) = J(\boldsymbol{u}^*). \tag{3.1}$$

Note that the domain of the function $\boldsymbol{u} \mapsto \sum_{i=1}^n S(x_i, t; u_i, a_i, b_i)$ equals

$$\prod_{i=1}^n [x_i - a_i t, x_i + b_i t], \tag{3.2}$$

and the point $\boldsymbol{u}^N = \boldsymbol{d}^N$ is in the set in (3.2) by definition of $\boldsymbol{d}^N$. Thus, the point $\boldsymbol{u}^N$ is in the domain of the function $\boldsymbol{u} \mapsto \sum_{i=1}^n S(x_i, t; u_i, a_i, b_i)$. By Lemma 3.3.2, the function $\boldsymbol{u} \mapsto \sum_{i=1}^n S(x_i, t; u_i, a_i, b_i)$ is continuous in its domain. It is also straightforward to check that the function $\boldsymbol{u} \mapsto \sum_{i=1}^n S(x_i, t; u_i, a_i, b_i)$ is Lipschitz in its domain, and we denote its Lipschitz constant by $L_i$. Thus, we have that

$$\left| \sum_{i=1}^n S(x_i, t; u_i^N, a_i, b_i) - \sum_{i=1}^n S(x_i, t; u_i^*, a_i, b_i) \right| \le \left( \sum_{i=1}^n L_i \right) \|\boldsymbol{u}^N - \boldsymbol{u}^*\|. \tag{3.3}$$

Then, the convergence of $\hat{S}^N(\boldsymbol{x}, t)$ to $S(\boldsymbol{x}, t)$ follows from (3.1) and (3.3).

Now, it remains to prove the second formula in (3.7). We have proved that $\boldsymbol{u}^N$ and $\boldsymbol{u}^*$ are both in the set in (3.2). Let $\{\boldsymbol{u}^{N_j}\}_j$ be a subsequence of $\{\boldsymbol{u}^N\}_N$ (i.e., we assume $N_1 < N_2 < \cdots$ and $\lim_{j\to\infty} N_j = +\infty$), such that for each $i \in \{1, \ldots, n\}$, the $i$-th component $\{u_i^{N_j}\}_j$ of the subsequence satisfies one of the following assumptions:

(i) There exists an index $r_i$ in $\{1, 2, 3\}$, such that there hold

$$u_i^{N_j} \ge 0 \quad \text{and} \quad (x_i, t, u_i^{N_j}) \in \bar{\Omega}_{r_i}(a_i, b_i) \quad \forall\, j \in \mathbb{N}.$$

(ii) There exists an index $r_i$ in $\{1,2,3\}$, such that there hold

$$u_i^{N_j} < 0 \quad \text{and} \quad (-x_i, t, -u_i^{N_j}) \in \bar{\Omega}_{r_i}(b_i, a_i) \quad \forall\, j \in \mathbb{N}.$$

Here, to emphasize the dependence on $a_i$ and $b_i$, we use $\Omega_{r_i}(a_i, b_i)$ and $\bar{\Omega}_{r_i}(a_i, b_i)$ to respectively denote the set defined in (3.8) with constants $a = a_i$ and $b = b_i$ and its closure. Note that the situations considered in cases (i) and (ii) give a partition of the set in (3.2) (where some sets in the partition may be empty and the sets may overlap on the boundary, but neither of these possibilities affect the result). Hence, if the statement is proved for any such subsequence $\{\boldsymbol{u}^{N_j}\}_j$, then the statement also holds for the whole sequence $\{\boldsymbol{u}^N\}_N$. Thus, it suffices to prove the statement for the subsequence $\{\boldsymbol{u}^{N_j}\}_j$.

Let $i \in \{1, \dots, n\}$ be any index. Assume case (i) holds for $\{u_i^{N_j}\}_j$ with the index $r_i$. In other words, we assume $(x_i, t, u_i^{N_j}) \in \bar{\Omega}_{r_i}(a_i, b_i)$ holds for any $j \in \mathbb{N}$. Since the set $\bar{\Omega}_{r_i}(a_i, b_i)$ is closed and the subsequence $\{u_i^{N_j}\}_j$ converges to $u_i^*$, we conclude that $(x_i, t, u_i^*) \in \bar{\Omega}_{r_i}(a_i, b_i)$ also holds. Then, by definition of $\gamma$ in $\Omega_{r_i}(a_i, b_i)$, it is straightforward to check that

$$\sup_{s\in[0,t]} \left| \gamma(s; x_i, t, u_i^{N_j}, a_i, b_i) - \gamma(s; x_i, t, u_i^*, a_i, b_i) \right| \le \left| u_i^{N_j} - u_i^* \right|. \tag{3.4}$$

The proof for case (ii) is similar, so we omit it. Note that (3.4) holds for any arbitrary

index $i \in \{1, \ldots, n\}$. Hence, we have that

$$\begin{aligned}
&\sup_{s\in[0,t]} \left\| \hat{\boldsymbol{\gamma}}^{N_j}(s;\boldsymbol{x},t) - \boldsymbol{\gamma}(s;\boldsymbol{x},t) \right\|^2 \\
&\qquad = \sup_{s\in[0,t]} \sum_{i=1}^{n} \left| \gamma(s;x_i,t,u_i^{N_j},a_i,b_i) - \gamma(s;x_i,t,u_i^*,a_i,b_i) \right|^2 \\
&\qquad \leq \sum_{i=1}^{n} \sup_{s\in[0,t]} \left| \gamma(s;x_i,t,u_i^{N_j},a_i,b_i) - \gamma(s;x_i,t,u_i^*,a_i,b_i) \right|^2 \\
&\qquad \leq \sum_{i=1}^{n} \left| u_i^{N_j} - u_i^* \right|^2 = \left\| \boldsymbol{u}^{N_j} - \boldsymbol{u}^* \right\|^2,
\end{aligned}$$

where the second inequality holds by (3.4). Thus, the second formula in (3.7) holds for the subsequence by the convergence of $\boldsymbol{u}^{N_j}$ to $\boldsymbol{u}^*$. Moreover, the argument holds for any such subsequence, and hence, the statement holds for the whole sequence. □

### 3.5.2 Proof of Proposition 3.2.2

Let $r$ be the index defined in (3.14), and let the index set $\mathcal{J} \subseteq \{1, \ldots, m\}$ be defined by:

$$\mathcal{J} := \operatorname*{arg\,min}_{i\in\{1,\ldots,m\}} S_i(\boldsymbol{x},t).$$

Then, we have that

$$S(\boldsymbol{x},t) - \hat{S}(\boldsymbol{x},t) = S(\boldsymbol{x},t) - \hat{S}_r(\boldsymbol{x},t) \leq S_r(\boldsymbol{x},t) - \hat{S}_r(\boldsymbol{x},t) \leq \epsilon,$$

where the first equality holds by definition of $r$ and the first inequality holds since $S$ satisfies (3.10). Similarly, for any $j \in \mathcal{J}$, we have that

$$S(\boldsymbol{x},t) - \hat{S}(\boldsymbol{x},t) = S_j(\boldsymbol{x},t) - \hat{S}(\boldsymbol{x},t) \geq S_j(\boldsymbol{x},t) - \hat{S}_j(\boldsymbol{x},t) \geq -\epsilon,$$

where the first equality holds by definition of $\mathcal{J}$ and the first inequality holds since $\hat{S}$ satisfies (3.15). Therefore, (3.17) holds.

Now, assume $S_j(\boldsymbol{x},t) > S(\boldsymbol{x},t) + 2\epsilon$ holds for each index $j$ satisfying $S_j(\boldsymbol{x},t) \neq S(\boldsymbol{x},t)$. We prove $r \in \mathcal{J}$ by contradiction. Assume $r$ is not in $\mathcal{J}$. Then, we have $S_r(\boldsymbol{x},t) \neq S(\boldsymbol{x},t)$. However, from straightforward calculation, we also have

$$\begin{aligned} S_r(\boldsymbol{x},t) - S(\boldsymbol{x},t) &\leq (S_r(\boldsymbol{x},t) - \hat{S}_r(\boldsymbol{x},t)) + (\hat{S}_r(\boldsymbol{x},t) - \hat{S}(\boldsymbol{x},t)) + (\hat{S}(\boldsymbol{x},t) - S(\boldsymbol{x},t)) \\ &\leq \epsilon + 0 + \epsilon = 2\epsilon, \end{aligned}$$

which leads to a contradiction with our assumption. Therefore, we have $r \in \mathcal{J}$, and hence (3.18) holds by definition of $r$ and $\mathcal{J}$. □

## 3.6 Summary

In this chapter, we presented analytical solutions to certain classes of control-constrained optimal control problems and the corresponding HJ PDEs where the associated running cost and Hamiltonian are dependent on the state variable. Moreover, we provided efficient numerical methods for these problems and describe both CPU and FPGA implementations for these methods. While our CPU implementations already demonstrate the efficiency of our solvers in high dimensions, our FPGA implementations demonstrate the additional performance boosts and benefits that FPGAs can achieve over CPUs. Our numerical results provide several examples for which our numerical algorithms overcome the curse of dimensionality and demonstrate that our algorithms have potential for real-time high-dimensional optimal control applications. An interesting future research direction would be to combine our algorithms with other methods, such as numerical algorithms involving

the LQR solver and/or more complicated state or control constraints (see, for instance, [48]), to address broader classes of optimal control problems. Additionally, while the FPGA implementations presented in this chapter provide some insight into common challenges that arise when designing FPGA implementations, these challenges and our proposed solutions for overcoming these challenges are not fully understood. As such, more research needs to be done in order to establish more robust FPGA implementation methodology for scientific computing methods.

# Part II

# Connection between the multi-time Hopf formula and learning problems

In the previous part, we focused on the connection between HJ PDEs and optimal control to derive new Hopf- and Lax-Oleinik-type representation formulas for the solutions to certain classes of HJ PDEs with state-dependent Hamiltonians and their corresponding optimal control problems. We then leveraged those representation formulas to create efficient numerical algorithms to solve those problems in high dimensions. In this part, we again consider the connection between HJ PDEs and optimal control. However, instead of developing new representation formulas or solvers for HJ PDEs (and their corresponding optimal control problems), we use existing representation formulas for certain multi-time HJ PDEs to establish a novel connection between multi-time HJ PDEs (and their corresponding optimal control problems) and particular optimization problems that arise in machine learning. We then leverage this novel connection to reuse existing numerical methods from optimal control (namely, those for solving the Riccati ODEs) to design new training approaches for certain machine learning applications. The content of this part is joint work with Tingwei Meng, Zongren Zou, Jérôme Darbon, and George Em Karniadakis [27]. My main contributions include the development of the new theoretical connection and our Riccati-based methodology.

Chapter Four

---

# Leveraging Multi-time Hamilton-Jacobi PDEs for Certain Scientific Machine Learning Problems

## 4.1 Introduction

In this chapter, we establish a novel theoretical connection between certain optimization problems arising in machine learning and the multi-time Hopf formula (Section 4.2). Specifically, we show that there are one-to-one correspondences between the loss functions in certain learning problems and the objective function of the multi-time Hopf formula, which in turn yields connections to certain optimal control problems. See Figure 4.1 for an illustration of these correspondences. As such, our novel connection increases the interpretability of the training process of certain machine learning applications by showing that when we solve these learning problems, we actually solve a multi-time HJ PDE and by extension, its corresponding optimal control problem. In this chapter, we show that our novel connection allows us to leverage HJ PDE and optimal control theory to solve optimization problems arising in certain machine learning applications, and we reserve the reverse direction for future work.

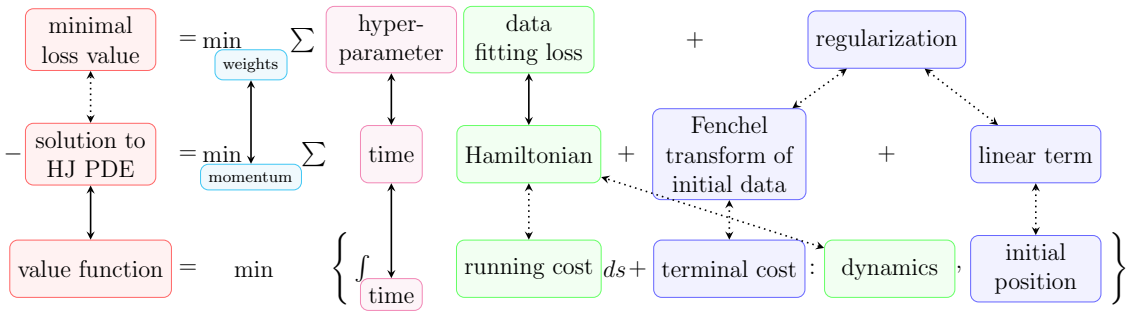



Figure 4.1: (See Section 4.2) Illustration of a connection between a regularized learning problem (**top**), the multi-time Hopf formula (**middle**), and an optimal control problem (**bottom**). The colors indicate the associated quantities between each problem. For example, the optimal weights in the learning problem are equivalent to the momentum in the HJ PDE (cyan), and the hyper-parameters for the data fitting terms correspond to the time variables in the HJ PDE and the time horizon in the optimal control problem (magenta). This color scheme is reused in the subsequent illustrations of our connection. The solid-line arrows denote direct equivalences. The dotted arrows represent additional mathematical relations.

We focus on the connection between the regularized linear regression problem and the Linear Quadratic Regulator (LQR) (Section 4.3). We use this case to gain a deeper understanding of our novel connection and illustrate its potential benefits. Linear regression consists of learning a linear prediction model and is one of the fundamental learning problems in supervised machine learning [122, 101, 131]. LQR [128] is a well-studied optimal control problem with certain quadratic running and terminal costs and linear dynamics and is typically solved using the Riccati ordinary differential equations (ODEs) [98, 32, 104]. Through our novel connection, we establish that solving certain regularized linear regression problems is equivalent to solving certain LQR problems. As a result, we can leverage standard LQR solvers to design new training approaches for machine learning. In particular, we develop new methodology for solving certain regularized linear regression problems by adapting solvers for the Riccati ODEs to these new settings (Section 4.4).

To highlight the versatility and possible computational advantages of our new Riccati-based approach, we apply our methodology to several test problems in machine learning (Section 4.5). In the first example, we consider a function approximation problem to demonstrate the computational and memory advantages of our Riccati-based approach in the context of continual learning [109, 83, 129]. Specifically, we show that our Riccati-based approach naturally enables us to continually adapt the learned model to new data without having to store or retrain on the previous data (which is especially significant given the rise of big data), while also avoiding catastrophic forgetting [83, 109]. In the second example, we demonstrate how our Riccati-based approach can be used to perform post-training calibrations. In particular, our approach gives us the flexibility to add or remove data points and tune hyper-parameters to increase the accuracy of the learned model without having to retrain it entirely, which again provides computational and memory advantages

over conventional learning methods. In the third example, we use our Riccati-based method to fit the last layer of a physics-informed neural network (PINN) [116] using transfer learning [135, 45]. In this application, we demonstrate that as we change the value of the hyper-parameters, solving the associated Riccati ODEs not only provides the solution to the updated problem, but also a *continuum* of solutions along a 1D curve on the Pareto front of the data fitting losses and regularization. Finally, in the fourth example, we highlight the versatility of our Riccati-based approach by showing how it can be combined with existing optimization methods (e.g., the primal-dual hybrid gradient (PDHG) algorithm [20]) to perform sparse dynamics identification [16].

The main contributions of this work are the development of a new theoretical connection between learning problems and the Hopf formula and a new Riccati-based approach for solving regularized linear regression problems. While we demonstrate promising results, the work presented here has some limitations. For example, while we establish our novel theoretical connection between more general learning problems, multi-time HJ PDEs, and optimal control problems, we have yet to fully explore non-linear learning models, general convex Hamiltonians, or non-linear control dynamics. Additionally, as discussed previously, we have also not yet investigated what possible advantages our novel connection provides for solving HJ PDEs and optimal control problems. In particular, many efficient solvers for high-dimensional problems in machine learning exist [88, 60]; it would be desirable to be able to leverage our connection to reuse this machinery for HJ PDEs and optimal control. Thus, our novel connection presents many exciting opportunities. We discuss some other possible future directions in Section 4.6. The overall structure of this chapter is illustrated in Figure 4.2.

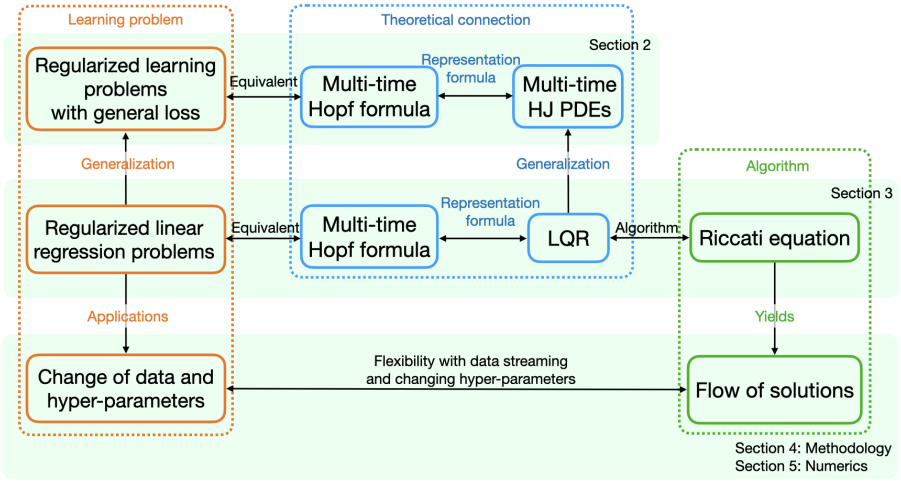



Figure 4.2: Overview of the overall structure of this chapter, the learning problems considered, our new theoretical connection, and our new Riccati-based algorithm. We show the learning problem in the **left** (orange), the theoretical connection in the **middle** (blue), and the algorithm in the **right** (green). The **top** row contains the general connection between learning problems and HJ PDEs (see Section 4.2), the **middle** row contains the connection between the regularized linear regression problem and LQR (see Section 4.3), and the **bottom** row contains the connection between applications in machine learning and our Riccati-based algorithm (see Sections 4.4 and 4.5).

## 4.2 Generalized Hopf formula

In this section, we provide some mathematical background on the single- and multi-time Hopf formulas. Specifically, we review the well-known connections between the Hopf formula, the solution to the HJ PDEs, and the solution to the corresponding optimal control problems. We then present a novel theoretical connection between the Hopf formula and certain learning problems. Through this connection, we establish that when we solve certain learning problems, we are actually evaluating the solution to certain HJ PDEs and their corresponding optimal control problems, and vice versa.

### 4.2.1 Introduction to the Hopf formula

Recall that the single-time HJ PDE is given by

$$\begin{cases} \dfrac{\partial S(\mathbf{x},t)}{\partial t} + H(\nabla_{\mathbf{x}} S(\mathbf{x},t)) = 0 & \mathbf{x} \in \mathbb{R}^n, t > 0, \\ S(\mathbf{x},0) = J(\mathbf{x}) & \mathbf{x} \in \mathbb{R}^n, \end{cases} \tag{4.1}$$

where $H : \mathbb{R}^n \to \mathbb{R}$ is the Hamiltonian and $J : \mathbb{R}^n \to \mathbb{R}$ is the initial condition. Assume that $H$ and $J$ are convex. Then, the viscosity solution to the single-time HJ PDE (4.1) is given by the Hopf formula [65]:

$$S(\mathbf{x},t) = \sup_{\mathbf{p}\in\mathbb{R}^n} \{\langle \mathbf{x},\mathbf{p}\rangle - tH(\mathbf{p}) - J^*(\mathbf{p})\} = -\inf_{\mathbf{p}\in\mathbb{R}^n} \{tH(\mathbf{p}) + J^*(\mathbf{p}) - \langle \mathbf{x},\mathbf{p}\rangle\}, \tag{4.2}$$

where $f^*$ denotes the Fenchel-Legendre transform of the function $f$; i.e., $f^*(p) = \sup_{x\in\mathbb{R}^n}\{\langle x,p\rangle - f(x)\}$.

The Hopf formula (4.2) also solves the following optimal control problem:

$$\min_{\boldsymbol{x}(\cdot)} \left\{ \int_0^t L(\mathbf{u}(s))ds + J(\boldsymbol{x}(t)) \colon \dot{\boldsymbol{x}}(s) = f(\mathbf{u}(s)) \forall s \in (0,t], \boldsymbol{x}(0) = \mathbf{x} \right\}, \tag{4.3}$$

where the running cost $L$ and the source term $f$ of the dynamics are related to the Hamiltonian $H$ by $H(\mathbf{p}) = \sup_{\mathbf{u}\in\mathbb{R}^n}\{\langle -f(\mathbf{u}),\mathbf{p}\rangle - L(\mathbf{u})\}$ and, in this context, we interpret $J$ to be the terminal cost.

A natural generalization of this formulation to the multi-time case is as follows. Let $H_1, \ldots, H_N$ be convex Hamiltonians, such that dom $H_i = \mathbb{R}^n$ for all $i = 1, \ldots, N$,

and let $J$ be a convex initial condition. Then, the multi-time HJ PDE is given by

$$\begin{cases} \dfrac{\partial S(\mathbf{x},t)}{\partial t_i} + H_i(\nabla_{\mathbf{x}} S(\mathbf{x},t)) = 0 \text{ for } i \in \{1,\dots,N\} & \mathbf{x} \in \mathbb{R}^n, t_1,\dots,t_N > 0, \\ S(\mathbf{x},0,\dots,0) = J(\mathbf{x}) & \mathbf{x} \in \mathbb{R}^n, \end{cases} \tag{4.4}$$

and the solution to the multi-time HJ PDE (4.4) can be represented by the following generalized (multi-time) Hopf formula [95]:

$$\begin{aligned} S(\mathbf{x},t_1,\dots,t_N) &= \sup_{\mathbf{p}\in\mathbb{R}^n} \left\{ \langle \mathbf{x},\mathbf{p}\rangle - \sum_{i=1}^N t_i H_i(\mathbf{p}) - J^*(\mathbf{p}) \right\} \\ &= -\inf_{\mathbf{p}\in\mathbb{R}^n} \left\{ \sum_{i=1}^N t_i H_i(\mathbf{p}) + J^*(\mathbf{p}) - \langle \mathbf{x},\mathbf{p}\rangle \right\}. \end{aligned} \tag{4.5}$$

Moreover, the generalized Hopf formula (4.5) also solves an optimal control problem in the form of (4.3) with terminal time $t = \sum_{j=1}^N t_j$ and where the running cost $L$ and the source term $f$ of the dynamics are defined piecewise by $L(s,\mathbf{u}) = L_i(\mathbf{u})$ and $f(s,\mathbf{u}) = f_i(\mathbf{u})$, respectively, for $s \in \left(\sum_{j=1}^{i-1} t_j, \sum_{j=1}^{i} t_j\right]$ and $i = 1,\dots,N$. The piecewise running costs $L_i$ and piecewise source terms $f_i$ of the dynamics are related to the Hamiltonians $H_i$ by $H_i(\mathbf{p}) = \sup_{\mathbf{u}\in\mathbb{R}^n}\{\langle -f_i(\mathbf{u}),\mathbf{p}\rangle - L_i(\mathbf{u})\}$.

### 4.2.2 Connection between the Hopf formula and learning problems

In this section, we connect the Hopf formulas (4.2) and (4.5) with certain learning problems. Consider a learning problem with data points $\{(\boldsymbol{z}_i,\boldsymbol{y}_i)\}_{i=1}^N \subset \mathbb{R}^m \times \mathbb{R}^M$. The goal of the learning problem is to find a function $F(\boldsymbol{z};\boldsymbol{\theta})$ with input $\boldsymbol{z} \in \mathbb{R}^m$ and unknown parameter $\boldsymbol{\theta} \in \mathbb{R}^n$, such that $\mathcal{A}F(\boldsymbol{z}_i;\boldsymbol{\theta})$ approximately equals $\boldsymbol{y}_i$ at every

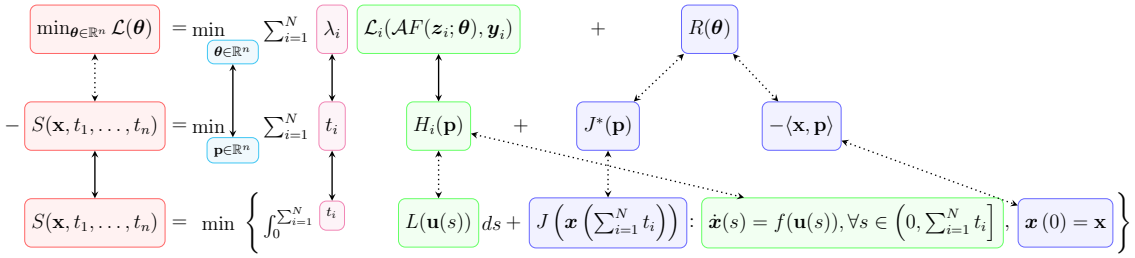



Figure 4.3: (See Section 4.2) Mathematical formulation describing the connection between a regularized learning problem (**top**), the multi-time Hopf formula (**middle**), and an optimal control problem (**bottom**). The content of this illustration matches that of Figure 4.1 by replacing each term in Figure 4.1 with its corresponding mathematical expression. The colors indicate the associated quantities between each problem. The solid-line arrows denote direct equivalences. The dotted arrows represent additional mathematical relations.

$\boldsymbol{z}_i$. Here $\mathcal{A}$ is an operator acting on the function $F$. For instance, $\mathcal{A}$ could be the identity operator (as in the regression problem [131]) or a differential operator (as in PINNs [116]). Then, the learning problem is given by the following optimization problem:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^n} \sum_{i=1}^N \lambda_i \mathcal{L}_i(\mathcal{A}F(\boldsymbol{z}_i;\boldsymbol{\theta}), \boldsymbol{y}_i) + R(\boldsymbol{\theta}). \tag{4.6}$$

The above loss function consists of two parts: each of $\mathcal{L}_i(\mathcal{A}F(\boldsymbol{z}_i;\boldsymbol{\theta}), \boldsymbol{y}_i)$ is a data fitting term at $(\boldsymbol{z}_i, \boldsymbol{y}_i)$ (where $\mathcal{L}_i(\boldsymbol{a}, \boldsymbol{b})$ is a function measuring the discrepancy between $\boldsymbol{a}$ and $\boldsymbol{b}$) and $R$ is a regularization term. In this chapter, we assume the function $\boldsymbol{\theta} \mapsto \mathcal{L}_i(\mathcal{A}F(\boldsymbol{z}_i;\boldsymbol{\theta}), \boldsymbol{y}_i)$ is convex for all $i = 1, \ldots, N$.

Then, the connection between the learning problem (4.6), the Hopf formulas (4.2) and (4.5), and the optimal control problem (4.3) is illustrated in Figure 4.3. Specifically, if there is only one data point ($N = 1$), the learning problem (4.6) is related to the single-time Hopf formula (4.2) by setting $H(\mathbf{p}) = \mathcal{L}_1(\mathcal{A}F(\boldsymbol{z}_1;\mathbf{p}), \boldsymbol{y}_1)$, $t = \lambda_1$, and $R(\mathbf{p}) = J^*(\mathbf{p}) - \langle \mathbf{x}, \mathbf{p}\rangle$. Hence, when we solve these single-point learning problems, we simultaneously evaluate the solution to the HJ PDE (4.1) at the point $(\mathbf{x}, t)$, or equivalently, we solve the corresponding optimal control problem (4.3). Conversely,

when we solve the HJ PDE (4.1), the spatial gradient $\nabla_{\mathbf{x}} S(\mathbf{x}, t)$ of the solution gives the minimizer $\boldsymbol{\theta}^*$ to the single-point learning problem (4.6).

If there are $N > 1$ data points, then the learning problem (4.6) is related to the multi-time Hopf formula (4.5) by setting $H_i(\mathbf{p}) = \mathcal{L}_i(\mathcal{A}F(\boldsymbol{z}_i; \mathbf{p}), \boldsymbol{y}_i)$, $t_i = \lambda_i$, and $R(\mathbf{p}) = J^*(\mathbf{p}) - \langle \mathbf{x}, \mathbf{p} \rangle$. Hence, solving these multi-point learning problems is equivalent to evaluating the solution to the HJ PDE (4.5) at the point $(\mathbf{x}, t_1, \ldots, t_N)$, which is also equivalent to solving the corresponding optimal control problem (4.3) with terminal time $t = \sum_{j=1}^{N} t_j$ and piecewise running costs $L_i$ and dynamics $f_i$ related via $H_i(\mathbf{p}) = \sup_{\mathbf{u} \in \mathbb{R}^n} \{\langle -f_i(\mathbf{u}), \mathbf{p} \rangle - L_i(\mathbf{u})\}$. Similarly to the single-time case, when we solve the multi-time HJ PDE (4.4), the spatial gradient $\nabla_{\mathbf{x}} S(\mathbf{x}, t_1, \ldots, t_N)$ of the solution gives the minimizer $\boldsymbol{\theta}^*$ to the multi-point learning problem (4.6).

## 4.3 Linear Quadratic Regulator

In this section, we develop our theoretical connection from Section 4.2.2 in the specific case where the optimal control problem (4.3) corresponds to the LQR problem [128]. We show that solving certain LQR problems is equivalent to solving learning problems with linear models, quadratic data fitting losses, and quadratic regularization (i.e., an $\ell_2$-regularized linear regression problem). Although broader classes of learning problems are of interest, we restrict to linear regression problems as a starting point for demonstrating the potential of our theoretical connection. Specifically, we leverage our novel theoretical connection to show how established techniques for solving certain HJ PDEs (e.g., the Riccati ODEs [98]) can be reused to solve this class of learning problems.

### 4.3.1 Introduction to the Linear Quadratic Regulator and Riccati equation

The finite-horizon, continuous-time LQR is given by

$$S(\mathbf{x},t) = \min\left\{ \int_0^t \left(\frac{1}{2}\boldsymbol{x}(s)^T\mathcal{Q}\boldsymbol{x}(s) + \frac{1}{2}\mathbf{u}(s)^T\mathcal{R}\mathbf{u}(s) + \boldsymbol{x}(s)^T\mathcal{N}\mathbf{u}(s)\right) ds \right.$$
$$\left. + \frac{1}{2}\boldsymbol{x}(t)^T\mathcal{Q}_f\boldsymbol{x}(t) : \dot{\boldsymbol{x}}(s) = A\boldsymbol{x}(s) + B\mathbf{u}(s) \forall s \in (0,t], \boldsymbol{x}(0) = \mathbf{x}\right\}, \quad (4.1)$$

where $\mathcal{Q}, \mathcal{Q}_f \in \mathbb{R}^{n\times n}$ and $\mathcal{R} \in \mathbb{R}^{m\times m}$ are symmetric positive definite, $\mathcal{N} \in \mathbb{R}^{n\times m}$, $A \in \mathbb{R}^{n\times n}$, and $B \in \mathbb{R}^{n\times m}$. The corresponding HJ PDE is

$$\begin{cases} \dfrac{\partial S(\mathbf{x},t)}{\partial t} + H(\mathbf{x}, \nabla_{\mathbf{x}} S(\mathbf{x},t)) = 0 & \mathbf{x} \in \mathbb{R}^n, t > 0 \\ S(\mathbf{x},0) = J(\mathbf{x}) & \mathbf{x} \in \mathbb{R}^n, \end{cases} \quad (4.2)$$

where the initial data of the HJ PDE is given by the terminal cost $J(\mathbf{x}) := \frac{1}{2}\mathbf{x}^T\mathcal{Q}_f\mathbf{x}$ of the optimal control problem and the Hamiltonian $H$ is defined by

$$\begin{aligned} H(\mathbf{x},\mathbf{p}) &= \sup_{\mathbf{u}\in\mathbb{R}^n} \langle -f(\mathbf{x},\mathbf{u}), \mathbf{p}\rangle - L(\mathbf{x},\mathbf{u}) \\ &= -\langle A\mathbf{x}, \mathbf{p}\rangle - \frac{1}{2}\langle \mathbf{x}, \mathcal{Q}\mathbf{x}\rangle + \frac{1}{2}\langle B^T\mathbf{p} + \mathcal{N}^T\mathbf{x}, \mathcal{R}^{-1}(B^T\mathbf{p} + \mathcal{N}^T\mathbf{x})\rangle, \end{aligned} \quad (4.3)$$

where $f(\mathbf{x},\mathbf{u}) = A\mathbf{x} + B\mathbf{u}$ is the source term of the dynamics and $L(\mathbf{x},\mathbf{u}) = \frac{1}{2}\mathbf{x}^T\mathcal{Q}\mathbf{x} + \frac{1}{2}\mathbf{u}^T\mathcal{R}\mathbf{u} + \mathbf{x}^T\mathcal{N}\mathbf{u}$ is the running cost. Note that because of the spatial dependence in the Hamiltonian $H$, the Hopf formula cannot be applied directly to the above LQR problem without additional assumptions. In Sections 4.3.2 and 4.3.3, we discuss some assumptions under which $H$ is independent of the spatial variable $\mathbf{x}$ and the corresponding learning problems that can be solved via our connection through the Hopf formula (see, for example, Figure 4.4).

It is well-known that this LQR problem can be solved using the Riccati equation as follows [98]. Define $C_{pp} = B\mathcal{R}^{-1}B^T$, $C_{xx} = -\mathcal{N}\mathcal{R}^{-1}\mathcal{N}^T + \mathcal{Q}$, and $C_{xp} = A - B\mathcal{R}^{-1}\mathcal{N}^T$. Then, the solution is also given by $S(\mathbf{x}, t) = \frac{1}{2}\mathbf{x}^T P(t)\mathbf{x}$, where the function $P : [0, \infty) \to \mathbb{R}^{n\times n}$ takes values in the space of symmetric positive definite matrices and solves the following Riccati equation:

$$\begin{cases} \dot{P}(t) = -P(t)^T C_{pp} P(t) + P(t)^T C_{xp} + C_{xp}^T P(t) + C_{xx} \quad t \in (0, +\infty), \\ P(0) = \mathcal{Q}_f. \end{cases} \tag{4.4}$$

When $H$ does not depend on $\mathbf{x}$, we can use our connection to modify the corresponding LQR problem to consider different HJ PDEs and, hence, to accommodate different learning problems. For example, adding lower order terms in the running cost $L$ and/or the source term $f$ of the dynamics corresponds to adding lower order terms in the Hamiltonian of the HJ PDE or, equivalently, in the data fitting term of the learning problem. Similarly, adding lower order terms in the terminal cost $J$ corresponds to adding lower order terms in the initial condition of the HJ PDE or, equivalently, in the regularization term of the learning problem.

### 4.3.2 Connection to single-point regularized linear regression problems

In this section, we establish a relation between the linear regression problem with only one data point and the LQR problem (4.1) with $A = \mathcal{Q} = 0$ and $\mathcal{N} = 0$. Note that, to do this, we also add in some lower order terms to the original LQR

problem (4.1). In this case, the LQR problem becomes

$$S(\mathbf{x},t) = \min\left\{ \int_0^t \left(\frac{1}{2}\mathbf{u}(s)^T\mathcal{R}\mathbf{u}(s) - \boldsymbol{a}^T\mathbf{u}(s)\right) ds \right.$$
$$\left. + \frac{1}{2}\boldsymbol{x}(t)^T\mathcal{Q}_f\boldsymbol{x}(t) + \boldsymbol{b}^T\boldsymbol{x}(t) \colon \dot{\boldsymbol{x}}(s) = B\mathbf{u}(s) \forall s \in (0,t], \boldsymbol{x}(0) = \mathbf{x} \right\}, \quad (4.5)$$

and the corresponding HJ PDE is given by (4.2), where $J(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathcal{Q}_f\mathbf{x} + \boldsymbol{b}^T\mathbf{x}$ is the initial data/terminal cost whose Fenchel transform is given by

$$J^*(\mathbf{p}) = \sup_{\mathbf{x}\in\mathbb{R}^n} \langle \mathbf{x}, \mathbf{p}\rangle - J(\mathbf{x}) = \frac{1}{2}\left\|\mathcal{Q}_f^{-1/2}(\mathbf{p} - \boldsymbol{b})\right\|_2^2$$

and the Hamiltonian $H$ is given by

$$H(\mathbf{p}) = \sup_{\mathbf{u}\in\mathbb{R}^n} \langle -B\mathbf{u}, \mathbf{p}\rangle - \frac{1}{2}\mathbf{u}^T\mathcal{R}\mathbf{u} + \boldsymbol{a}^T\mathbf{u} = \frac{1}{2}\left\|\mathcal{R}^{-1/2}(B^T\mathbf{p} - \boldsymbol{a})\right\|_2^2,$$

where $f(\mathbf{u}) = B\mathbf{u}$ is the source term of the dynamics and $L(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathcal{R}\mathbf{u}$ is the running cost. Then, using the single-time Hopf formula (4.2), we have that the solution to the HJ PDE is given by

$$S(\mathbf{x},t) = \sup_{\mathbf{p}\in\mathbb{R}^n} \left\{ \langle \mathbf{x}, \mathbf{p}\rangle - \frac{t}{2}\left\|\mathcal{R}^{-1/2}(B^T\mathbf{p} - \boldsymbol{a})\right\|_2^2 - \frac{1}{2}\left\|\mathcal{Q}_f^{-1/2}(\mathbf{p} - \boldsymbol{b})\right\|_2^2 \right\}. \quad (4.6)$$

In this case, we can compute the maximizer in the above Hopf formula explicitly, which can be done numerically using the method of least squares. Alternatively, this LQR problem can also be solved via the Riccati ODEs, which are given by

$$\dot{P}(t) = -P(t)^TB\mathcal{R}^{-1}B^TP(t), \quad \dot{\mathbf{q}}(t) = -P(t)^TB\mathcal{R}^{-1}(B^T\mathbf{q}(t) - \boldsymbol{a}), \quad (4.7)$$

$$\dot{r}(t) = -\frac{1}{2}\left\|\mathcal{R}^{-1/2}(B^T\mathbf{q}(t) - \boldsymbol{a})\right\|_2^2,$$

with initial conditions $P(0) = \mathcal{Q}_f$, $\mathbf{q}(0) = \boldsymbol{b}$, and $r(0) = 0$.

The above Hopf formula (4.6) is related to the learning problem (4.6) with quadratic data fidelity term and quadratic regularization. Specifically, let $t = \lambda$ and $\mathbf{p} = \boldsymbol{\theta}$. Then, solving the above maximization problem (4.6) is equivalent to minimizing the following loss function with respect to $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^T$:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{\lambda}{2} \left\| \mathcal{R}^{-1/2}(B^T\boldsymbol{\theta} - \boldsymbol{a}) \right\|_2^2 + \frac{1}{2} \left\| \mathcal{Q}_f^{-1/2}(\boldsymbol{\theta} - (\boldsymbol{b} + \mathcal{Q}_f\mathbf{x})) \right\|_2^2. \tag{4.8}$$

This loss function corresponds to a one-point linear regression problem, where the regularization term is given by $\frac{1}{2} \left\| \mathcal{Q}_f^{-1/2}(\boldsymbol{\theta} - (\boldsymbol{b} + \mathcal{Q}_f\mathbf{x})) \right\|_2^2$ and the data fitting term is given by $\frac{\lambda}{2} \left\| \mathcal{R}^{-1/2}(B^T\boldsymbol{\theta} - \boldsymbol{a}) \right\|_2^2$; i.e., set $N = 1$ in Figure 4.4. The minimizer $\boldsymbol{\theta}^*$ of $\mathcal{L}$ is related to the solution of the Riccati equation via

$$\boldsymbol{\theta}^*(= \mathbf{p}^*) = \nabla_{\mathbf{x}} S(\mathbf{x}, \lambda) = P(\lambda)\boldsymbol{x} + \mathbf{q}(\lambda), \tag{4.9}$$

where $\mathbf{p}^*$ is the minimizer in the Hopf formula (4.6). Note that if we only need to recover the minimzer $\boldsymbol{\theta}^*$, then we only need to solve two ODEs (i.e., the ODEs for $P, \mathbf{q}$) since (4.9) does not depend on $r$.

### 4.3.3 Connection to multi-point regularized linear regression problems

If the running cost is piecewise quadratic with $\mathcal{Q} = 0, \mathcal{N} = 0$ on each piece and the dynamics are piecewise linear with $A = 0$ on each piece, then we get a more general piecewise LQR problem in the form of (4.3) with piecewise running costs $L_i(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathcal{R}_i\mathbf{u} - \boldsymbol{a}_i^T\mathbf{u}$, piecewise dynamics $f_i(\mathbf{u}) = B_i\mathbf{u}$, terminal cost

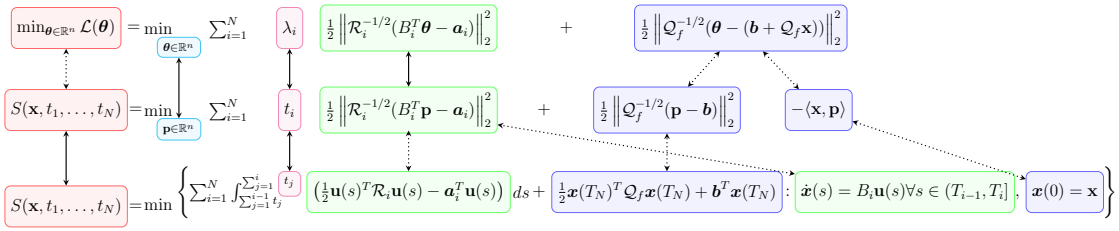



Figure 4.4: (See Section 4.3) Mathematical formulation describing the connection between a regularized linear regression problem (**top**), the multi-time Hopf formula (**middle**), and a piecewise LQR problem with $A = \mathcal{Q} = 0$ and $\mathcal{N} = 0$ on each piece (**bottom**). Note that $T_i = \sum_{j=1}^i t_j$. The content of this illustration is a special case of the connection in Figure 4.3 using quadratic data fitting losses and quadratic regularization. The colors indicate the associated quantities between each problem. The solid-line arrows denote direct equivalences. The dotted arrows represent additional mathematical relations.

$J(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathcal{Q}_f\mathbf{x} + \boldsymbol{b}^T\mathbf{x}$, and terminal time $t = \sum_{j=1}^N t_j$. Recall that this optimal control problem corresponds to the multi-time HJ PDE (4.4) with Hamiltonians $H_i(\mathbf{p}) = \sup_{\mathbf{u}\in\mathbb{R}^n}\{\langle -f_i(\mathbf{u}), \mathbf{p}\rangle - L_i(\mathbf{u})\}$ and initial data $J$. Then, the solution to this LQR problem and corresponding HJ PDE is given by the following multi-time Hopf formula:

$$S(\mathbf{x}, t_1, \ldots, t_N) = \sup_{\mathbf{p}\in\mathbb{R}^n} \left\{ \langle \mathbf{x}, \mathbf{p}\rangle - \sum_{i=1}^N \frac{t_i}{2}\|\mathcal{R}_i^{-1/2}(B_i^T\mathbf{p} - \boldsymbol{a}_i)\|_2^2 - \frac{1}{2}\|\mathcal{Q}_f^{-1/2}(\mathbf{p} - \boldsymbol{b})\|_2^2 \right\}. \tag{4.10}$$

Define $T_m$ to be

$$T_m = \sum_{j=1}^m t_j. \tag{4.11}$$

Then, the multi-time HJ PDE and piecewise LQR problem can also be solved using the following Riccati equation: $S(\mathbf{x}, t_1, \cdots, t_N) = \frac{1}{2}\mathbf{x}^T P(T_N)\mathbf{x} + \mathbf{q}(T_N)^T\mathbf{x} + r(T_N)$, where the function $P : [0, \infty) \to \mathbb{R}^{n\times n}$ takes values in the space of symmetric positive definite matrices and solves the following piecewise Riccati equation:

$$\begin{cases} \dot{P}(s) = -P(s)^T B_i\mathcal{R}_i^{-1}B_i^T P(s) \quad s \in (T_{i-1}, T_i) \\ P(0) = \mathcal{Q}_f, \end{cases} \tag{4.12}$$

the function $\mathbf{q} : [0, \infty) \to \mathbb{R}^n$ solves the following piecewise linear ODE:

$$\begin{cases} \dot{\mathbf{q}}(s) = -P(s)^T B_i \mathcal{R}_i^{-1}(B_i^T \mathbf{q}(s) - \boldsymbol{a}_i) \quad s \in (T_{i-1}, T_i)\,, \\ \mathbf{q}(0) = -\boldsymbol{b}, \end{cases} \tag{4.13}$$

and the function $r : [0, \infty) \to \mathbb{R}$ solves the following piecewise ODE:

$$\begin{cases} \dot{r}(s) = -\dfrac{1}{2} \left\| \mathcal{R}_i^{-1/2}(B_i^T \mathbf{q}(s) - \boldsymbol{a}_i) \right\|_2^2 \quad s \in (T_{i-1}, T_i)\,, \\ r(0) = 0. \end{cases} \tag{4.14}$$

The above multi-time Hopf formula (4.10) can also be regarded as a linear regression problem with multiple data points, as illustrated in Figure 4.4. Let $\frac{1}{2}\|\mathcal{Q}_f^{-1/2}(\mathbf{p} - \boldsymbol{b})\|_2^2$ be the the regularization term and $\frac{t_i}{2}\|\mathcal{R}_i^{-1/2}(B_i^T \mathbf{p} - \boldsymbol{a}_i)\|_2^2$ be the data fitting term at the $i$-th data point. Then, the multi-time LQR problem (4.10) is equivalent to the learning problem $\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$, where the loss function $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{\lambda_i}{2} \left\| \mathcal{R}_i^{-1/2}(B_i^T \boldsymbol{\theta} - \boldsymbol{a}_i) \right\|_2^2 + \frac{1}{2} \left\| \mathcal{Q}_f^{-1/2}(\boldsymbol{\theta} - (\boldsymbol{b} + \mathcal{Q}_f \mathbf{x})) \right\|_2^2. \tag{4.15}$$

In Section 4.4.1, we discuss a specific example of the learning problem (4.15), which is more readily recognizable as the standard linear regression problem. The minimizer $\boldsymbol{\theta}^*$ of the learning problem (4.15) (and $\mathbf{p}^*$ of the Hopf formula (4.10)) is given by

$$\boldsymbol{\theta}^* (= \mathbf{p}^*) = \nabla_{\mathbf{x}} S(\mathbf{x}, t_1, \ldots, t_N) = P\left(T_N\right) \mathbf{x} + \mathbf{q}\left(T_N\right). \tag{4.16}$$

## 4.4 Methodology

In the previous section, we established a novel theoretical connection between certain multi-time HJ PDEs, LQR problems, and regularized linear regression problems. In this section, we leverage this theoretical connection to design novel algorithms for solving various learning problems. The connection between the learning problem of interest, the Hopf formula, and the corresponding optimal control problem is summarized in Figure 4.5 and Table 4.1.

### 4.4.1 Solving the regularized linear regression problem using Riccati ODEs

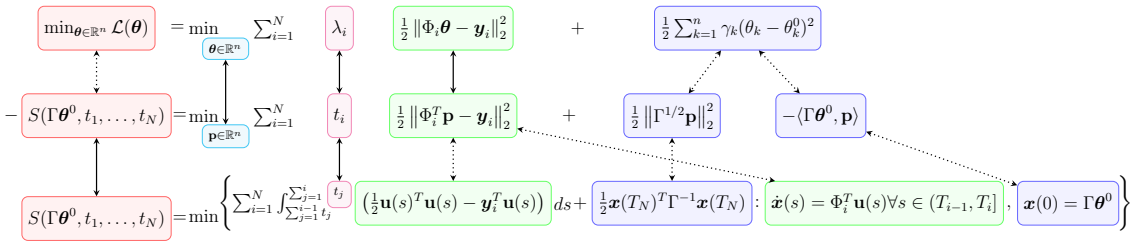



Figure 4.5: (See Section 4.4) Mathematical formulation describing the connection between our model linear regression problem with quadratic regularization (**top**), the multi-time Hopf formula (**middle**), and a piecewise LQR problem (**bottom**). Note that $T_i = \sum_{j=1}^{i} t_j$. The content of this illustration is a special case of the connection in Figure 4.4 using $\mathcal{R}_i = I$, $B_i = \Phi_i^T$, $\boldsymbol{a}_i = \boldsymbol{y}_i$, $\mathcal{Q}_f = \Gamma^{-1}$, $\boldsymbol{b} = 0$, and $\mathbf{x} = \Gamma\boldsymbol{\theta}^0$. We use this connection to develop our Riccati-based methodology (Section 4.4) and perform our numerical examples (Section 4.5). The colors indicate the associated quantities between each problem. The solid-line arrows denote direct equivalences. The dotted arrows represent additional mathematical relations.

A linear regression problem with quadratic data fitting loss and quadratic regularization is formulated as follows. The goal of this learning problem is to fit $N$ data points $(\boldsymbol{z}_i, \boldsymbol{y}_i) \in \mathbb{R}^m \times \mathbb{R}^M$ with the linear prediction model $\Phi_i\boldsymbol{\theta} \approx \boldsymbol{y}_i$, where $\Phi_i = [\phi_1(\boldsymbol{z}_i), ..., \phi_n(\boldsymbol{z}_i)] \in \mathbb{R}^{M\times n}$ is the matrix whose columns are the basis functions

| **Terms in the loss function (4.1)** | **Terms in the Hopf formula** | **Terms in the optimal control problem** | **Learning problems** |
|---|---|---|---|
| $\Phi_i, \boldsymbol{y}_i$ | Hamiltonian | Running cost, dynamics | Continual learning (Section 4.5.1); Post-training calibration (Section 4.5.2) |
| $\lambda_i$ | Time | Time | Post-training calibration (Section 4.5.2) |
| $\gamma_k$ | Hamiltonian; Initial condition | Running cost, dynamics; Terminal cost | Hyper-parameter tuning, flow along the Pareto front (Section 4.5.3) |
| $\theta_k^0$ | Spatial variable | Initial position | Generalized regularization functions (Section 4.5.4) |

Table 4.1: Summary of the learning problems and corresponding numerical examples presented in this chapter. Each row summarizes which terms in the loss function (4.1), the Hopf formula, and the optimal control problem must be changed to match the context of various learning problems.

$\phi_j : \mathbb{R}^m \to \mathbb{R}^M$, $j = 1, \ldots, n$ evaluated at $\boldsymbol{z}_i$ and $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^T \in \mathbb{R}^n$ are unknown trainable coefficients. We learn $\boldsymbol{\theta} \in \mathbb{R}^n$ by minimizing the following loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^{N} \lambda_i \|\Phi_i \boldsymbol{\theta} - \boldsymbol{y}_i\|_2^2 + \frac{1}{2} \sum_{k=1}^{n} \gamma_k (\theta_k - \theta_k^0)^2, \tag{4.1}$$

where $\lambda_i \geq 0, i = 1, ..., N$ are tunable hyper-parameters for the data fitting losses, $\gamma_k > 0, k = 1, ..., n$ are tunable hyper-parameters for the regularization terms, and $\theta_k^0 \in \mathbb{R}$ is a prior on the unknown coefficients that biases $\theta_k$ to be close to $\theta_k^0$. Note that the above loss function (4.1) is strictly convex, and hence it has a unique global minimizer. Since this loss function is quadratic in $\boldsymbol{\theta}$, it can be minimized exactly using the method of least squares. However, we explore a different approach for minimizing (4.1), which, in certain learning contexts, yields computational advantages over conventional approaches like the method of least squares.

Note that this learning problem is in the form of (4.15); i.e., set $\mathcal{R}_i = I_{M \times M}$, $B_i = \Phi_i^T$, $\boldsymbol{a}_i = \boldsymbol{y}_i$, $\mathcal{Q}_f = \Gamma^{-1}$, where $\Gamma \in \mathbb{R}^{n \times n}$ is the diagonal matrix whose $i$-th entry is $\gamma_i$, $\boldsymbol{b} = 0$, and $\mathbf{x} = \Gamma \boldsymbol{\theta}^0$. Then, this learning problem is related to an LQR problem, and we summarize this connection in Figure 4.5. Thus, this learning

problem can alternatively be solved via the following Riccati ODEs:

$$\begin{cases} \dot{P}(t) = -P(t)^T \Phi_i^T \Phi_i P(t) & t \in (T_{i-1}, T_i)\,, \\ \dot{\mathbf{q}}(t) = -P(t)^T \Phi_i^T (\Phi_i \mathbf{q}(t) - \boldsymbol{y}_i) & t \in (T_{i-1}, T_i)\,, \end{cases} \tag{4.2}$$

with initial condition $P(0) = \Gamma^{-1}$, $\mathbf{q}(0) = 0$. Note that here we disregard the ODE for $r$ since, for the learning problem, we are only concerned with the value of the minimizer $\boldsymbol{\theta}^*$, which only requires the values of $P, \mathbf{q}$ (e.g., see (4.9), (4.16)), whereas recovering the minimal objective value $\mathcal{L}(\boldsymbol{\theta}^*)$ (which is generally not needed in the context of learning) would require all three values $P, \mathbf{q}, r$. There are many numerical methods for solving the Riccati ODEs (4.2) in the literature. In this chapter, we use the 4th-order Runge-Kutta method (RK4) [18].

At first glance, solving this linear regression problem via the Riccati ODEs (4.2) may seem unnecessary given the existence of other well-established methods for minimizing (4.1) (e.g., the method of least squares). However, note that (4.2) is actually a sequence of Riccati ODEs. Thus, using our theoretical connection (and hence, these sequential Riccati ODEs to minimize (4.1)) means that we have the flexibility to handle sequential changes to the learning problem. In the remainder of Section 4.4, we identify several applications in learning (summarized in Table 4.1), for which this Riccati-based approach yields computational advantages over traditional learning methods (especially when the number of data points $N$ is large), and we describe how standard Riccati solvers can be adapted for these contexts.

### 4.4.2 Adding or removing data

Since our theoretical connection gives us access to the Riccati ODEs (4.2), which are solved sequentially, we have the flexibility to handle sequential changes to the learning problem (4.1). In this section, we focus on the sequential addition or removal of data. Some related machine learning examples are provided in Sections 4.5.1 and 4.5.2.

First, we discuss the addition of one data point; i.e., we increase the number of data points from $N$ to $N+1$. This case corresponds to updating our learned model as new data is collected, which is crucial for many practical machine learning applications. To add one data point, we adapt the Riccati ODEs (4.2) as follows. Adding the $(N+1)$-th data point corresponds to adding the term $\frac{1}{2}\lambda_{N+1}\|\Phi_{N+1}\boldsymbol{\theta}-\boldsymbol{y}_{N+1}\|_2^2$ in the loss function (4.1) or, equivalently, to adding the Hamiltonian $\frac{1}{2}\|\Phi_{N+1}\boldsymbol{\theta}-\boldsymbol{y}_{N+1}\|_2^2$ to the multi-time HJ PDE and the pieces $L_{N+1}(s,\mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathbf{u} - \boldsymbol{y}_{N+1}^T\mathbf{u}$ and $f(s,\mathbf{u}) = \Phi_{N+1}^T\mathbf{u}, s \in (T_N, T_{N+1})$ to the running cost and dynamics, respectively, of the corresponding piecewise LQR problem. Thus, minimizing this new loss function is equivalent to solving the following Riccati ODE:

$$\begin{cases} \dot{\tilde{P}}(t) = -\tilde{P}(t)^T\Phi_j^T\Phi_j\tilde{P}(t) & t > 0, \\ \dot{\tilde{\mathbf{q}}}(t) = -\tilde{P}(t)^T\Phi_j^T(\Phi_j\tilde{\mathbf{q}}(t) - \boldsymbol{y}_j) & t > 0, \end{cases} \tag{4.3}$$

where the index $j = N+1$ and with initial condition $\tilde{P}(0) = P(T_N)$ and $\tilde{\mathbf{q}}(0) = \mathbf{q}(T_N)$, where $P(T_N)$ and $\mathbf{q}(T_N)$ are obtained from solving the learning problem (4.1) with $N$ data points. Then, the solution to the new learning problem with an additional point is given by

$$\tilde{\boldsymbol{\theta}}^* = \tilde{P}\Gamma\boldsymbol{\theta}^0 + \tilde{\mathbf{q}}, \tag{4.4}$$

where $\tilde{P} = \tilde{P}(\lambda_{N+1})(= P(T_{N+1}))$ and $\tilde{\mathbf{q}} = \tilde{\mathbf{q}}(\lambda_{N+1})(= \mathbf{q}(T_{N+1}))$ are the solution to (4.3); i.e., we have evolved the solution of the new corresponding multi-time HJ PDE in the time variable $t_{N+1}$ from $S(\mathbf{x}, t_1, \ldots, t_N, 0)$ to $S(\mathbf{x}, t_1, \ldots, t_N, \lambda_{N+1})$. Using the same methodology described above, note that we can also interpret adding one data point as solving a one-point linear regression problem (4.8) and its corresponding single-piece LQR problem (4.5).

Removing one data point (i.e., decreasing the number of data points from $N$ to $N-1$) corresponds to calibrating our learned model by removing possible outliers and/or overly noisy data. To remove one data point, we reverse time and solve a terminal value Riccati ODE as follows. Note that removing the $j$-th data point corresponds to removing the term $\frac{1}{2}\lambda_j\|\Phi_j\boldsymbol{\theta} - \boldsymbol{y}_j\|_2^2$ in the loss function (4.1) or, equivalently, removing the Hamiltonian $\frac{1}{2}\|\Phi_j\boldsymbol{\theta} - \boldsymbol{y}_j\|_2^2$ from the multi-time HJ PDE and removing the pieces $L_j(s, \mathbf{u}) = \frac{1}{2}\mathbf{u}^T\mathbf{u} - \boldsymbol{y}_j$ and $f(s, \mathbf{u}) = \Phi_j^T\mathbf{u}$ from the running cost and dynamics, respectively, of the corresponding piecewise LQR problem. Hence, numerically, we can remove the $j$-th data point by solving the following Riccati ODE:

$$\begin{cases} \dot{\tilde{P}}(t) = -\tilde{P}(t)^T\Phi_j^T\Phi_j\tilde{P}(t) & t < \lambda_j, \\ \dot{\tilde{\mathbf{q}}}(t) = -\tilde{P}(t)^T\Phi_j^T(\Phi_j\tilde{\mathbf{q}}(t) - \boldsymbol{y}_j) & t < \lambda_j, \end{cases} \tag{4.5}$$

with terminal condition $\tilde{P}(\lambda_j) = P(T_N)$ and $\tilde{\mathbf{q}}(\lambda_j) = \mathbf{q}(T_N)$, where $P(T_N)$ and $\mathbf{q}(T_N)$ are obtained from solving the learning problem (4.1) with all $N$ data points. Then, the solution to the new learning problem with the $j$-th point removed is given by (4.4), where $\tilde{P} = \tilde{P}(0)$ and $\tilde{\mathbf{q}} = \tilde{\mathbf{q}}(0)$ are the solution to (4.5).

Note that in both cases, the above approach only requires information about the data point to be added or removed and the results of the previous training. Thus, we can add and remove data without retraining on the entire dataset or

requiring access to all of the previous data. In contrast, traditional approaches to minimizing (4.1) (e.g., the method of least squares) would require memory of all previous data and then retraining on the entire updated data set. Hence, our approach provides promising computational and memory savings over conventional methods.

### 4.4.3 Hyper-parameter tuning

In the loss function (4.1), we have three types of hyper-parameters: the weights $\lambda_i$ of the data fitting terms, the weights $\gamma_k$ of the regularization term, and the bias $\boldsymbol{\theta}^0$ on the trainable coefficients $\boldsymbol{\theta}$. In this section, we discuss how we adapt the Riccati ODEs to tune each of these hyper-parameters.

First, we discuss tuning the weights of the data fitting terms; e.g., as in post-training calibration and federated learning (Section 4.5.2). Consider changing the weight on the $i$-th data fitting term from $\lambda_i$ to $\tilde{\lambda}_i$, which corresponds to changing the time $t_i = \lambda_i$ to $t_i = \tilde{\lambda}_i$ in the multi-time HJ PDE and corresponding piecewise LQR problem. Then, the methodology is similar to that in Section 4.4.1.

If $\tilde{\lambda}_i > \lambda_i$, then the solution to the learning problem with the new data fitting weight $\tilde{\lambda}_i$ is given by (4.4), where $\tilde{P} = \tilde{P}(\tilde{\lambda}_i - \lambda_i)$ and $\tilde{\mathbf{q}} = \tilde{\mathbf{q}}(\tilde{\lambda}_i - \lambda_i)$ are the solutions to the Riccati ODEs (4.3) with $j = i$, at time $(\tilde{\lambda}_i - \lambda_i)$, and with initial condition $\tilde{P}(0) = P(T_N)$ and $\tilde{\mathbf{q}}(0) = \mathbf{q}(T_N)$, where $P(T_N), \mathbf{q}(T_N)$ are obtained from solving the original learning problem (4.1) with the original value of $\lambda_i$. Similarly, if $\lambda_i > \tilde{\lambda}_i$, then the solution to the learning problem with the new data fitting weight $\tilde{\lambda}_i$ is given by (4.4), where $\tilde{P} = \tilde{P}(0)$ and $\tilde{\mathbf{q}} = \tilde{\mathbf{q}}(0)$ are the solutions to the time-reversed Riccati ODEs (4.5) with $j = i$, at time 0, and with terminal condition $\tilde{P}(\lambda_i - \tilde{\lambda}_i) = P(T_N)$

and $\tilde{\mathbf{q}}(\lambda_i - \tilde{\lambda}_i) = \mathbf{q}(T_N)$, where $P(T_N), \mathbf{q}(T_N)$ are obtained from solving the original learning problem (4.1) with the original value of $\lambda_i$.

Next, we discuss tuning the weights of the regularization terms; e.g., as in the hyper-parameter tuning example in Section 4.5.3. Note that in the original formulation of the Riccati ODEs (4.2), the regularization weights appear in the initial condition for $P$. Changing a hyper-parameter $\gamma_k$ by changing this initial condition would require re-solving the entire sequence of Riccati ODEs and hence retraining on the entire dataset. However, since we have freedom in how we formulate the corresponding multi-time HJ PDE, we can avoid this retraining as follows. Note that here, instead of sequentially changing each regularization weight $\gamma_k$ one-by-one, we can actually update all of the regularization weights at once.

We consider the case where we change each regularization parameter $\gamma_k$ to $\tilde{\gamma}_k$. This change can be regarded as two steps: first, we change all parameters $\gamma_k$ for the indices $k$ such that $\tilde{\gamma}_k > \gamma_k$, and then we change the other parameters. Define the index set $\mathcal{K}$ to be $\mathcal{K} = \{k \colon \tilde{\gamma}_k > \gamma_k\}$.

The first step is equivalent to adding the term $\sum_{k\in\mathcal{K}} \frac{\tilde{\gamma}_k - \gamma_k}{2}(\theta_k - \theta_k^0)^2$ to the loss function (4.1). We can interpret this as adding an $(N+1)$-th Hamiltonian $\frac{1}{2}\boldsymbol{\theta}^T \Gamma_+ \boldsymbol{\theta}$ with corresponding time variable $t_{N+1} = 1$ to the multi-time HJ PDE, where $\Gamma_+$ is a diagonal matrix whose $k$-th diagonal element is $\tilde{\gamma}_k - \gamma_k$ if $k \in \mathcal{K}$ and 0 otherwise. Therefore, the solution to this new multi-time HJ PDE can be solved by the following Riccati equation:

$$\begin{cases} \dot{P}_+(t) = -P_+(t)^T \Gamma_+ P_+(t) & t \in (0,1), \\ \dot{\mathbf{q}}_+(t) = -P_+(t)^T \Gamma_+ \mathbf{q}_+(t) & t \in (0,1), \end{cases} \tag{4.6}$$

with initial condition $P_+(0)$ and $\mathbf{q}_+(0)$, which are the corresponding solutions to the Riccati equations before changing the weights $\gamma_k, k \in \mathcal{K}$. In other words, we set

$P_+(0) = P(T_N)$ and $\mathbf{q}_+(0) = \mathbf{q}(T_N)$, where $P(T_N), \mathbf{q}(T_N)$ are obtained from solving the original learning problem (4.1) with the original values of $\gamma_k$.

The second step is equivalent to removing the term from the loss function (4.1). This is equivalent to solving a single-time HJ PDE with a terminal condition at time 1 and Hamiltonian $\frac{1}{2}\boldsymbol{\theta}^T\Gamma_-\boldsymbol{\theta}$, where $\Gamma_-$ is a diagonal matrix whose $k$-th diagonal element is $\gamma_k - \tilde{\gamma}_k$ if $k \notin \mathcal{K}$ and 0 otherwise. Then, the solution can be obtained by evolving this HJ PDE backward in time. We do this by solving the following Riccati equation:

$$\begin{cases} \dot{P}_-(t) = -P_-(t)^T\Gamma_-P_-(t) & t \in (0,1)\,, \\ \dot{\mathbf{q}}_-(t) = -P_-(t)^T\Gamma_-\mathbf{q}_-(t) & t \in (0,1)\,, \end{cases} \tag{4.7}$$

with terminal condition $P_-(1) = P_+(1)$ and $\mathbf{q}_-(1) = \mathbf{q}_+(1)$, where $P_+, \mathbf{q}_+$ are obtained from the solution to (4.6).

Finally, the minimizer of the new learning problem after changing all of the weights $\gamma_k$ in the regularization term is given by $P_-(0)\tilde{\Gamma}\boldsymbol{\theta}^0 + \mathbf{q}_-(0)$, where $P_-, \mathbf{q}_-$ are obtained from the solution to (4.7) and $\tilde{\Gamma}$ is a diagonal matrix whose $k$-th diagonal element is the new regularization parameter $\tilde{\gamma}_k$.

Next, we discuss tuning the bias $\boldsymbol{\theta}^0$ on the trainable coefficients $\boldsymbol{\theta}$; e.g., as in the generalized regularization example in Section 4.5.4. In the context of the learning problem, we interpret $\boldsymbol{\theta}^0$ as the center of a Gaussian prior on $\boldsymbol{\theta}$. Using our theoretical connection, changing $\boldsymbol{\theta}^0$ is also equivalent to evaluating the corresponding multi-time HJ PDE and piecewise LQR problem at a different point in space. Specifically, changing $\boldsymbol{\theta}^0$ to $\tilde{\boldsymbol{\theta}}^0$ means that instead of evaluating the multi-time HJ PDE and piecewise LQR problem at $\mathbf{x} = \Gamma\boldsymbol{\theta}^0$, we evaluate them at $\mathbf{x} = \Gamma\tilde{\boldsymbol{\theta}}^0$. Then, the

solution to the learning problem with the new bias $\tilde{\boldsymbol{\theta}}^0$ is given by

$$P(T_N)\Gamma\tilde{\boldsymbol{\theta}}^0 + \mathbf{q}(T_N), \tag{4.8}$$

where $P(T_N)$ and $\mathbf{q}(T_N)$ are obtained from solving the original learning problem (4.1) with the original value of $\boldsymbol{\theta}^0$. In other words, shifting the bias involves neither retraining nor access to any of the data points nor solving Riccati ODEs. Instead, we only require some matrix multiplication and addition involving the results of the previous training and the new bias $\tilde{\boldsymbol{\theta}}^0$.

### 4.4.4 General convex regularization functions

So far, we have only considered quadratic regularizations. In this section, we will consider the linear regression problem with an arbitrary convex regularization term. Specifically, consider the general regularization term $R(\boldsymbol{\theta}) - \langle \mathbf{x}, \boldsymbol{\theta} \rangle$, where $R : \mathbb{R}^n \to \mathbb{R}$ is a convex function. Then, the loss function of the learning problem is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N} \lambda_i \|\Phi_i \boldsymbol{\theta} - \boldsymbol{y}_i\|_2^2 + R(\boldsymbol{\theta}) - \langle \mathbf{x}, \boldsymbol{\theta} \rangle. \tag{4.9}$$

This learning problem corresponds to a multi-time HJ PDE (4.4), where the $i$-th Hamiltonian is $H_i(\mathbf{p}) = \frac{1}{2}\lambda_i \|\Phi_i \mathbf{p} - \boldsymbol{y}_i\|_2^2$ and the initial condition is $J = R^*$, which is the Fenchel-Legendre transform of $R$. The corresponding optimal control problem is given by (4.3), with terminal time $\sum_{i=1}^{N} \lambda_i$, piecewise running costs $L_i(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|^2 - \langle \boldsymbol{y}_i, \mathbf{u} \rangle$ and piecewise dynamics $f_i(\mathbf{u}) = \Phi_i^T \mathbf{u}$ on $(\sum_{j=1}^{i-1} \lambda_j, \sum_{j=1}^{i} \lambda_j]$, and terminal cost $R^*$. Then, the solution $\boldsymbol{\theta}^*$ to the learning problem (4.9) is equivalent to the spatial gradient $\nabla_{\mathbf{x}} S(\mathbf{x}, \lambda_1, \ldots, \lambda_N)$ of the solution to the multi-time HJ PDE, and the minimal value of the loss function (4.9) is equivalent to $-S(\mathbf{x}, \lambda_1, \ldots, \lambda_N)$.

Since the loss function (4.9) is convex, it can be minimized using any appropriate convex optimization algorithm. In this chapter, we demonstrate how our Riccati-based approach can be combined with PDHG [20] to solve this learning problem. Note that PDHG can be applied to this learning problem (4.9) as long as the proximal point of $R$ or $R^*$ is numerically computable. For instance, $R$ could be a quadratic function $\langle \cdot, M\cdot \rangle$, a quadratic norm $\sqrt{\langle \cdot, M\cdot \rangle}$, $\|\cdot\|_1$, $\|\cdot\|_1^2$, $\|\cdot\|_\infty^2$, a polynomial function, or the Fenchel-Legendre transformation of any of these functions [38]. To compute the solution $\boldsymbol{\theta}^*$ of this learning problem, PDHG iterates the following:

$$\begin{cases} \boldsymbol{\theta}^{\ell+1} = \underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\arg\min} \dfrac{1}{2}\displaystyle\sum_{i=1}^{N} \lambda_i \|\Phi_i \boldsymbol{\theta} - \boldsymbol{y}_i\|_2^2 + \dfrac{1}{2\sigma_{\boldsymbol{\theta}}} \left\|\boldsymbol{\theta} - (\boldsymbol{\theta}^\ell - \sigma_{\boldsymbol{\theta}}(\boldsymbol{w}^\ell - \mathbf{x}))\right\|_2^2, \\ \bar{\boldsymbol{\theta}}^{\ell+1} = 2\boldsymbol{\theta}^{\ell+1} - \boldsymbol{\theta}^\ell, \\ \boldsymbol{w}^{\ell+1} = \underset{\boldsymbol{w}\in\mathbb{R}^n}{\arg\min}\, R^*(\boldsymbol{w}) + \dfrac{1}{2\sigma_{\boldsymbol{w}}} \left\|\boldsymbol{w} - (\boldsymbol{w}^\ell + \sigma_{\boldsymbol{w}} \bar{\boldsymbol{\theta}}^{\ell+1})\right\|_2^2, \end{cases} \tag{4.10}$$

where $\sigma_{\boldsymbol{\theta}}$, $\sigma_{\boldsymbol{w}}$ are positive step-size parameters satisfying $\sigma_{\boldsymbol{\theta}}\sigma_{\boldsymbol{w}} < 1$. In each iteration, we need to solve two minimization problems. Updating $\boldsymbol{\theta}^\ell$ is equivalent to solving the learning problem with quadratic regularization. Specifically, updating $\boldsymbol{\theta}^\ell$ is the same as changing the bias $\theta^0$ in the loss function (4.1). Therefore, we only need to solve the Riccati ODEs (4.2) once (to compute $\boldsymbol{\theta}^1$) and every other iterate can be computed using (4.8). Updating $\boldsymbol{w}^\ell$ is equivalent to computing the proximal point of $R^*$. Depending on $R^*$, there may already exist efficient solvers or explicit formulas for computing its proximal point.

Note that with non-quadratic regularization, if we change the weights $\lambda_k$ of the data fitting losses, change $\mathbf{x}$ (which is related to the bias on $\boldsymbol{\theta}$), or add or remove data points, then we will need to restart PDHG to solve the resulting learning problem with these new hyper-parameter values or updated datasets. However, we can reuse some of the computations between runs. Namely, we only need to solve the full

sequence of Riccati ODEs (4.2) once (to compute $\boldsymbol{\theta}^1$ in the first run of PDHG). Then, the solution to those Riccati ODEs can be reused in combination with the methods described in Sections 4.4.2 and 4.4.3 (according to how the learning problem is modified) in subsequent iterations and runs of PDHG.

## 4.5 Numerical examples

In this section, we apply the Riccati-based methodology presented in Section 4.4 to four test problems from machine learning to demonstrate the versatility and potential computational advantages of our new approach. In each example, we use RK4 with double precision to solve the Riccati ODEs when applying our Riccati-based methodology. The connections between the examples presented in this section, the Hopf formula, and the corresponding optimal control problems can be found in Table 4.1.

### 4.5.1 Function approximation in continual learning

In this section, we test our Riccati-based approach (as described in Section 4.5.1) on a pedagogical function approximation example in continual learning [109, 83, 129] to demonstrate its computational and memory advantages. Under the continual learning framework, data is accessed in a stream and the trainable model parameters are updated incrementally as new data becomes available. In some cases, the historical data may also become inaccessible after new data is received, which can often lead to catastrophic forgetting [83, 109], which refers to the abrupt degradation in performance of learned models on previous tasks upon training on new tasks. In

this example, we show how our Riccati-based approach naturally coincides with the continual learning framework, while also inherently avoiding catastrophic forgetting even if the historical data is inaccessible.

Our example set-up is as follows. Our goal is to regress the function $y(\tau) = 0.1\tau + 0.05\sin(10\tau)$ given noisy data $\{(\tau_i, y_i \approx y(\tau_i))\}_{i=1}^N \subset \mathbb{R} \times \mathbb{R}$. Following the continual learning framework, we assume that a new noisy data point is available every $\Delta\tau = 0.01$ (i.e., $\tau_i = (i-1)\Delta\tau \in [0, 10]$, $i = 1, \ldots, N$). We regress $y(\tau)$ using the linear model $y(\tau) = \sum_{k=1}^n \theta_k \phi_k(\tau)$, where $n = 10$ and the basis functions are given by

$$\{\phi_k(\tau)\}_{k=1}^n = \{1, \tau, \tau^2, \tau^3, \sin(\tau), \sin(5\tau), \sin(8\tau), \sin(9\tau), \sin(10\tau), \sin(12\tau)\}. \quad (4.1)$$

The coefficients $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^T$ of our linear model are learned by minimizing the following loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^N \lambda_i \left| \sum_{k=1}^n \theta_k \phi_k(\tau_i) - y_i \right|^2 + \frac{1}{2}\sum_{k=1}^n \gamma_k |\theta_k|^2, \quad (4.2)$$

where $\lambda_i, i = 1, ..., N$ and $\gamma_k, k = 1, ..., n$ are weights for the data loss and $\ell_2$ regularization terms, respectively, and we update our learned coefficients every time a new data point is available. In our numerical experiments, we use additive Gaussian noise with zero mean and standard deviation 0.01, and we set $\lambda_i = 1, \forall i$ and $\gamma_k = 0.1, \forall k$.

Although this loss function (4.2) could be minimized using conventional machine learning techniques (e.g., the method of least squares), these methods typically require access to and training on the entire dataset $\{(\tau_i, y_i)\}_{i=1}^N$, which conflicts with the assumptions in continual learning. However, note that this loss function (4.2) is of the form (4.1). Thus, we can instead apply our Riccati-based approach (as

described in Sections 4.4.1 and 4.4.2) to solve this learning problem. In particular, the methodology described in Section 4.4.2 matches the data streaming paradigm of continual learning; we incrementally update our learned coefficients by considering each new data point as the time evolution of a corresponding multi-time HJ PDE. As a result, in contrast to conventional machine learning methods, our Riccati-based approach does not require storage of previous data points and its memory and computational complexity is constant for each added data point. As such, our Riccati-based methodology may be well-suited to online learning applications. Additionally, since this time evolution of the HJ PDE (i.e., the addition of a new data point) requires knowledge about the solution to the HJ PDE at the previous time (i.e., the results of the previous training), our Riccati-based approach also inherently avoids catastrophic forgetting.

The results of applying our Riccati-based method to solve this learning problem (4.4) are shown in Figure 4.6 and Table 4.2. Figure 4.6(a) depicts the evolution of the coefficients $\{\theta_k\}_{k=1}^n$ as more data points become available. We observe that the learned coefficients $\theta_k, k = 1, ..., n$ converge to their true values as more data points are incorporated into the model. Figure 4.6(b) displays three inferences at different times $\tau = 2, 4, 8$, which demonstrate that our Riccati-based approach is capable of real-time inferences without catastrophic forgetting, even though each inference is made using only one data point. Note that due to an appropriate choice of basis functions, our learned model is able to provide accurate extrapolations as well.

Table 4.2 displays the numerical errors of the minimizer $\boldsymbol{\theta}^*$ of the loss function (4.2) obtained using the Riccati-based methodology from Section 4.4 after the last data point becomes available at $\tau = 10$. We observe that the accuracy of our approach increases as we decrease the step size $h$ of RK4, which indicates that the errors of $\boldsymbol{\theta}^*$ stem from the accuracy of RK4 in solving the corresponding Riccati

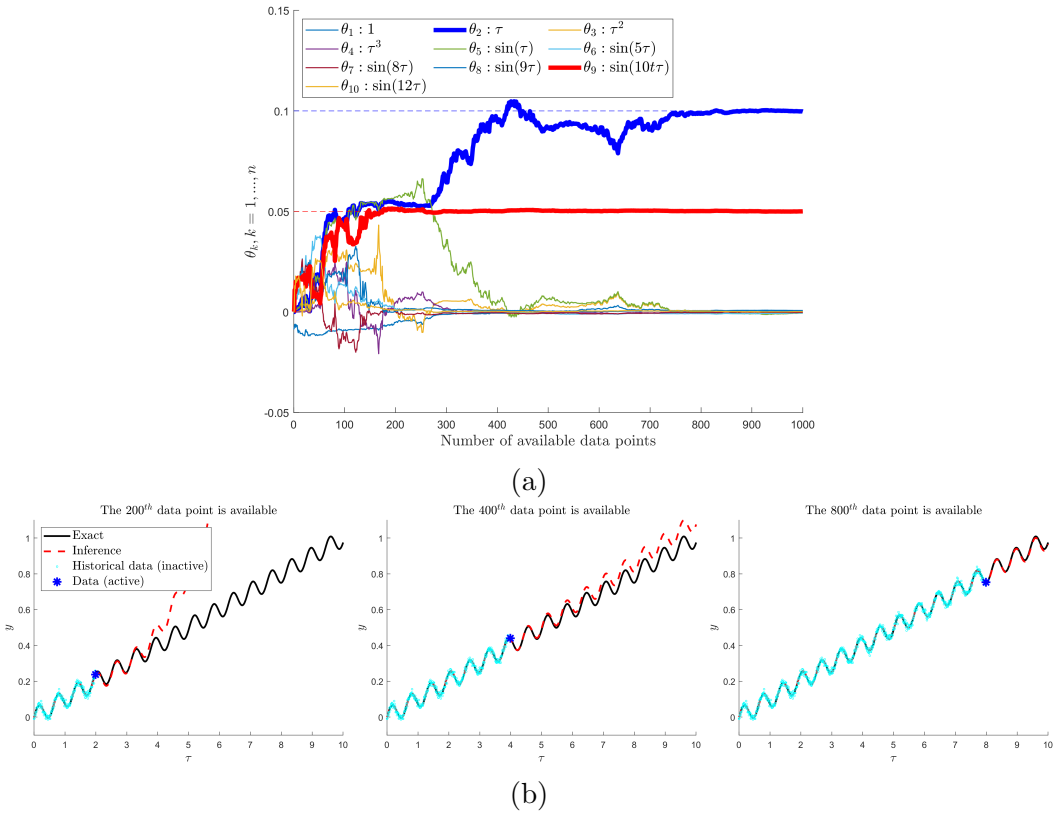



Figure 4.6: Evolution and continual learning of the function approximation learned using our Riccati-based approach as more data becomes available. (a) shows the evolution of the learned coefficients $\theta_k, k = 1, ..., n$ as more data is incorporated into the model; the horizontal dotted lines denote the exact reference values. (b) shows the inferences of $y$ after the 200th, 400th, and 800th noisy data point becomes available. Our Riccati-based approach allows us to incrementally update the learned coefficients as more data becomes available without requiring access to the previous data or re-training on the entire dataset, which provides advantages in both memory and computations over conventional learning methods.

ODEs. The reference solution is obtained by minimizing (4.2) directly using the method of least squares and assuming that all $N = 1000$ data points are accessible.

| | $h = 0.2$ | $h = 0.1$ | $h = 0.01$ |
|---|---|---|---|
| $\ell_1$ error of $\boldsymbol{\theta}^*$ | $1.1210 \times 10^{-6}$ | $6.0924 \times 10^{-9}$ | $4.2222 \times 10^{-12}$ |
| $\ell_1$ relative error of $\boldsymbol{\theta}^*$ | $7.3622 \times 10^{-6}$ | $4.0012 \times 10^{-8}$ | $2.7729 \times 10^{-11}$ |

Table 4.2: Errors in computing the minimizer $\boldsymbol{\theta}^*$ of the function approximation loss (4.2) using our Riccati-based approach. We use RK4 to solve the Riccati ODEs (4.2) and (4.3) with double precision and various step sizes $h$. The reference is obtained by using the method of least squares to minimize the loss function (4.2) directly.

### 4.5.2 1D steady-state reaction-diffusion equation and post-training calibration

In this example, we use our Riccati-based approach to apply post-training calibrations when solving a PDE. Specifically, we leverage the methodology in Section 4.4.2 to add or remove data and the methodology in Sections 4.4.3 to enforce the boundary conditions of the PDE without retraining the entire learned model. Consider the following 1D steady-state reaction-diffusion equation:

$$\begin{cases} D\dfrac{\partial^2 u}{\partial x^2}(x) + \kappa u(x) = f(x), x \in [0,1], \\ u(0) = u(1) = 0, \end{cases} \tag{4.3}$$

where $D = 0.01$ is the diffusion coefficient, $\kappa = -1$, and $f(x)$ is the source term of which noisy measurements $\{(x_i, f_i \approx f(x_i))\}_{i=1}^N \subset \mathbb{R} \times \mathbb{R}$ are available. We consider the scenario where regular training has been employed but with either insufficient data or sufficient data with outliers, both of which yield inaccurate inferences of the solution. We further assume that extra information is provided after the regular training and seek to perform post-training calibrations to incorporate this extra information into the already-trained models without losing the information from the

original training. In the literature, it is well-established that post-training calibration can significantly increase the performance of deployed machine learning methods [115, 136]. However, designing computationally efficient methods for performing these post-calibrations is still of great interest.

In this example, we solve this PDE (4.3) by reformulating the PDE as an optimization problem [116, 126, 62]. We use a linear model to approximate the solution, i.e. $u(x) = \sum_{k=1}^{n} \theta_k \phi_k(x)$, where $n = 21$ and $\{\phi_k(x)\}_{k=1}^{n}$ are the truncated Fourier basis functions on $[0, 1]$, $\{1\} \cup \{\sin(2l\pi x), \cos(2l\pi x)\}_{l=1}^{(n-1)/2}$. We learn the coefficients $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^T$ of the linear model by minimizing the following loss:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N} \lambda_i \left| D\sum_{k=1}^{n} \theta_k \frac{\partial^2 \phi}{\partial x^2}(x_i) + \kappa \sum_{k=1}^{n} \theta_k \phi_k(x_i) - f_i \right|^2 \\ + \frac{1}{2}\lambda_b \left| \sum_{k=1}^{n} \theta_k \phi_k(0) - 0 \right|^2 + \frac{1}{2}\lambda_b \left| \sum_{k=1}^{n} \theta_k \phi_k(1) - 0 \right|^2 + \frac{1}{2}\sum_{k=1}^{n} \gamma_k |\theta_k|^2,\end{aligned} \tag{4.4}$$

where $\lambda_i, i = 1, ..., N$, $\lambda_b$, and $\gamma_k, k = 1, .., n$ are balancing weights for the PDE residual, the boundary conditions, and the regularization term, respectively. In our numerical experiments, we assume the exact solution to be $u(x) = \sin^3(2\pi x)$ (and $f$ to be defined by (4.3), accordingly) and the noise to be additive Gaussian with zero mean and standard deviation 0.1. For the regular training, we set $\lambda_i = 1$, $\lambda_b = 1$, and $\gamma_k = 1$ and apply our Riccati-based method (see Section 4.4.1) to get our original estimate of the minimizer $\boldsymbol{\theta}^*$ of the loss function (4.4). Using our Riccati-based method yields an $\ell_1$ error of $8.9154 \times 10^{-10}$ and relative $\ell_1$ error of $1.4162 \times 10^{-9}$ in $\boldsymbol{\theta}^*$, where the reference is obtained by minimizing (4.4) directly using the method of least squares.

In the leftmost column of Figure 4.7, we see that the accuracy of both $u$ and $f$ as inferred by the regular training is impaired by a lack of data around the highest

peak and lowest valley of the exact functions. To compensate, we first calibrate our model by adding some new noisy measurements of $f$ in these regions where the data points are sparse. This calibration uses the methodology described in Section 4.4.2, and the results are shown in the middle column of Figure 4.7. Next, we note that the inferred $u$ still disagrees with the exact solution at the boundary points. Hence, we further calibrate our model by increasing the value of the boundary weight $\lambda_b$ from 1 to 10 to enforce the boundary conditions of the PDE. This calibration is done using the methodology described in Section 4.4.3 and the results of this second calibration are presented in the rightmost column of Figure 4.7. In both cases, we observe that the calibrations successfully improve the accuracy of the learned model.

Note that our Riccati-based approach allows us to perform each of these calibration steps using only the new or changed values in that step and the results of the previous training step. In other words, each step of this training process (including the original training and each subsequent calibration step) is done without sharing data between training steps, which exactly matches the framework of federated learning [93]. Thus, our methodology may be relevant to distributed training or collaborative learning applications, where data privacy is of concern.

Next, we discuss another post-training calibration technique. In this case, we assume the data for the regular training is sufficient but contains outliers due to large noise. We again assume the regular training is performed using the Riccati-based approach from Section 4.4.1. Then, we eliminate these outliers using the methodology described in Section 4.4.2, which only requires knowledge about the outliers to be removed and the results of the regular training. In Figure 4.8, we show the results of this post-training outlier removal. We observe that successively removing these outlier points successfully improves the accuracy of the learned models.

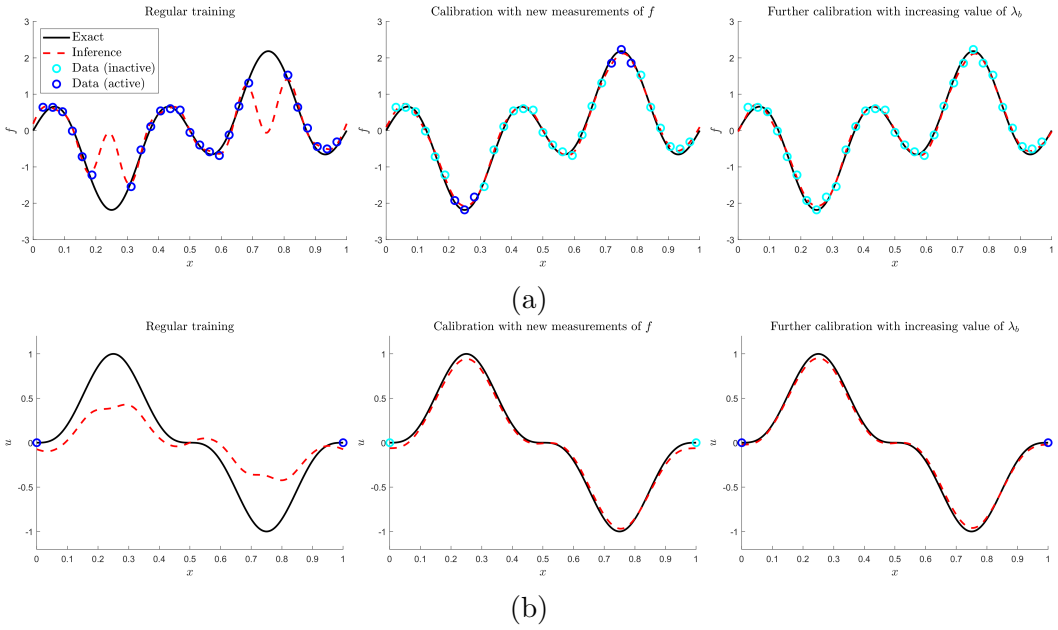
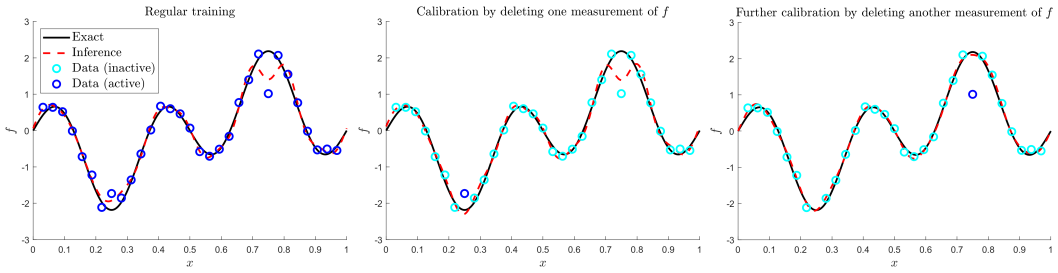


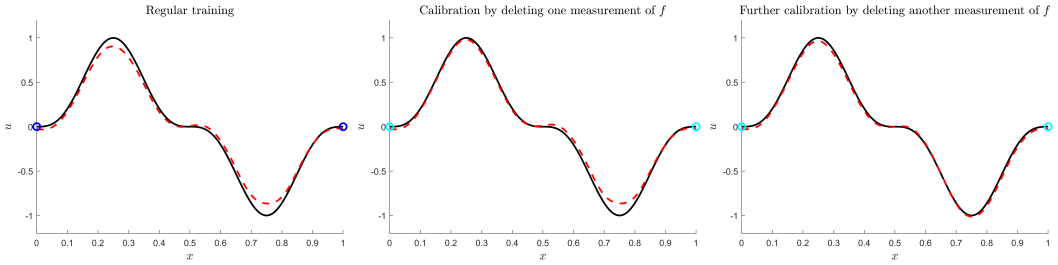

Figure 4.7: Results of solving the 1D steady-state reaction-diffusion equation (4.3) with noisy measurements of the source term $f$ in the domain and noiseless measurements of the solution $u$ on the boundary. (a) results for $f$; (b) results for $u$. **Left**: results of regular training; **middle**: calibrating the results of regular training with some additional noisy measurements of $f$; **right**: calibrating the results further by enforcing the boundary conditions by increasing the value of $\lambda_b$ in the loss function (4.4). The regular training uses our Riccati-based method in Section 4.4.1 to minimize (4.4), while the calibrations use the adaptations of our method in Section 4.4.2. Calibrations are employed without re-training or access to the data from the previous training, which demonstrates the advantages in both memory storage and computational complexity of our Riccati-based approach over conventional machine learning methods.

### 4.5.3 Poisson equation using PINNs and transfer learning

In this example, we demonstrate the versatility of our Riccati-based method by combining it with existing machine learning techniques to fit the last layer of a PINN. We also show that when we perform hyper-parameter tuning by solving the associated Riccati ODEs (Section 4.4.3), we not only provide the solution to the updated problem, but also a continuum of solutions along a 1D curve on the Pareto front of the data fitting losses and regularization. Consider the 2D Poisson equation




(a)




(b)

Figure 4.8: Results of solving the 1D steady-state reaction-diffusion equation (4.3) with noisy measurements of the source term $f$ in the domain and noiseless measurements of the solution $u$ on the boundary. (a) results for $f$; (b) results for $u$. **Left**: results of regular training using our Riccati-based method in Section 4.4.1; **middle** and **right**: calibrating the results of regular training by eliminating two outlier measurements of $f$ using the methodology described in Section 4.4.2. Calibrations are employed without re-training or access to the data from the previous training.

with Dirichlet boundary conditions is given by

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2}(x,y) + \dfrac{\partial^2 u}{\partial y^2}(x,y) = f(x,y) & (x,y) \in \Omega, \\ u(x,y) = 0 & (x,y) \in \partial\Omega, \end{cases} \tag{4.5}$$

where $\Omega := [-1,1]^2$ and $f$ is a source term. In this example, we solve this equation using transfer learning and PINNs. Consider the scenario where we only have access to measurements $\{(x_i, y_i, f_i = f(x_i, y_i))\}_{i=1}^N$ of $f$ at limited sampling points. Transfer learning compensates for this lack of knowledge, by transferring the knowledge from models pre-trained on similar problems to solve this new problem of interest. In this case, we learn the linear model $u(x,y) = \sum_{k=1}^n \theta_k \phi_k(x,y)$, where each basis function $\phi_k(x,y), k = 1, \ldots, n$ is the PINN solution of a neural network pre-trained to solve

the 2D Poisson equation (4.5) with similar source terms. Transfer learning using pre-trained neural networks as basis functions, as we do here, has recently grown in popularity in the scientific machine learning community [45, 135, 61] and has been shown to provide efficient yet accurate inferences even given very limited data [135]. We then learn the coefficients $\boldsymbol{\theta}$ of our linear model by minimizing the following PINN-type loss:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N} \lambda_i \left| \sum_{k=1}^{n} \theta_k \left( \frac{\partial^2 \phi_k}{\partial x^2} + \frac{\partial^2 \phi_k}{\partial y^2} \right)(x_i, y_i) - f_i \right|^2 + \frac{1}{2}\sum_{k=1}^{n} \gamma_k |\theta_k|^2, \tag{4.6}$$

where $\lambda_i = 1, i = 1, ..., N$ and $\gamma_k = \gamma, k = 1, ..., n$ are weights for the data fitting and $\ell_2$-regularization terms, respectively. Note that minimizing (4.6) with respect to $\boldsymbol{\theta}$ is equivalent to fitting the last layer of a neural network given pre-trained previous nonlinear layers as the basis functions.

In our numerical experiments, we use transfer learning to solve the 2D Poisson equation (4.5) with source term $f(x, y) = \sin(2.5\pi x)\sin(2.5\pi y)$ using $n = 100$ basis functions and $N = 100$ random measurements of $f$. We use the multi-head PINN method [135] to obtain the basis functions, which correspond to the shared nonlinear hidden layers of the pre-trained PINN solutions to (4.5) with source terms $f(x, y) = \sin(k\pi x)\sin(k\pi y), k = 1, 2, 3, 4$. We solve the learning problem (4.6) using our Riccati-based method from Section 4.4.1.

In Table 4.3, we compare the errors of the minimizer $\boldsymbol{\theta}^*$ of (4.6) and the solution $u$ of the PDE (4.5) as we decrease the weight $\gamma(= \gamma_k, \forall k)$ of the regularization term from 1 to 1e-5. The reference for $\boldsymbol{\theta}^*$ is obtained from minimizing (4.6) directly for each value of $\gamma$ using the method of least squares. The reference for $u$ is computed using a finite difference method with a five-point stencil to solve (4.5). Note that we originally minimize (4.6) using $\gamma = 1$ and the Riccati-based method in Section 4.4.1.

We then compute the solutions for the other values of $\gamma$ by incrementally decreasing $\gamma$ by a factor of 10 and using the methodology for hyper-parameter tuning described in Section 4.4.3 to reuse the results of training with the previous value $\gamma$ to compute the solution for the new value of $\gamma$. Consequently, we see that the error in $\boldsymbol{\theta}^*$ increases as we decrease $\gamma$ due to error accumulation from repeated applications of RK4. However, we also see that the error of $u$ generally decreases as we decrease $\gamma$ with the lowest error being achieved when $\gamma =$ 1e-4.

| | $\gamma = 1$ | $\gamma = 0.1$ | $\gamma = 0.01$ | $\gamma = 0.001$ | $\gamma = 0.0001$ | $\gamma = 0.00001$ |
|---|---|---|---|---|---|---|
| $\ell_1$ error of $\boldsymbol{\theta}^*$ | $5.9707 \times 10^{-10}$ | $4.2905 \times 10^{-8}$ | $1.0725 \times 10^{-6}$ | $1.7235 \times 10^{-5}$ | $2.0490 \times 10^{-4}$ | $5.8749 \times 10^{-3}$ |
| $L^2$ relative error of $u$ | 6.5247% | 5.6793% | 3.0390% | 1.7280% | 1.1662% | 1.4736% |

Table 4.3: Errors of the minimizer $\boldsymbol{\theta}^*$ of (4.6) and the solution $u$ to the 2D Poisson equation (4.5) using transfer learning and our Riccati-based approach. The reference for $\boldsymbol{\theta}^*$ is given by minimizing (4.6) with the corresponding value of $\gamma$ directly using the method of least squares, and the reference for $u$ is given by solving (4.5) using a finite difference method. Since we decrease $\gamma$ incrementally from 1 to 1e-5, the error of $\boldsymbol{\theta}^*$ accumulates due to successive applications of RK4.

From the results in Table 4.3, we see that our choice of the hyper-parameter $\gamma$ can greatly influence the accuracy of our learned model. Note that since we fix $\lambda_i = 1, \forall i$ and $\gamma_k = \gamma, \forall k$, we can view (4.6) as a bi-objective loss, where the two objectives are the weighted data fitting term $\frac{1}{2}\sum_{i=1}^{N}\left|\sum_{k=1}^{n}\theta_k\left(\frac{\partial^2\phi_k}{\partial x^2}+\frac{\partial^2\phi_k}{\partial y^2}\right)(x_i,y_i)-f_i\right|^2$ and the regularization term $\frac{1}{2}\sum_{k=1}^{n}|\theta_k|^2$. To better understand the effects of tuning $\gamma$, in Figure 4.9, we explore the Pareto front of these two objectives. Traditional scalarization-based approaches for computing the Pareto front typically rely on discrete samplings of the Pareto front corresponding to discrete choices of $\gamma$ [76]. While our Riccati-based methodology from Section 4.4.3 also recovers discrete points on the Pareto front corresponding to particular choices of $\gamma$, note that when we change $\gamma$, e.g., from $\gamma = 1$ to $\gamma = 0.1$, we also recover a one-dimensional curve along the Pareto front corresponding to every $\gamma \in [0.1, 1]$. We obtain this 1D curve theoretically via the flow of solutions obtained from the corresponding Riccati ODEs and numerically via the intermediate steps of RK4. The left plot of Figure 4.9 shows the 1D curve

along the Pareto front recovered by our Riccati-based approach (although note that in this example, the Pareto front is also one-dimensional and hence is equivalent to the exposed 1D curve), where the flow of solutions corresponding to decreasing $\gamma$ is represented by the arrows. Thus, although in general our methodology cannot compute the entire Pareto front, every time we change the value of the hyper-parameters, our approach recovers a continuous 1D curve along the Pareto front. In the right plot of Figure 4.9, we also visualize how the $L^2$ error of our learned solution $u$ changes as we decrease $\gamma$.

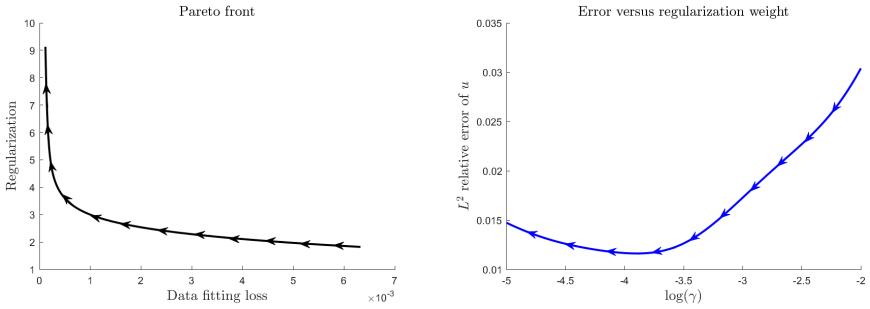



Figure 4.9: Results of changing the regularization weight $\gamma$ when solving the 2D Poisson equation using PINNs and transfer learning. We incrementally decrease the value of $\gamma$ by evolving the corresponding Riccati ODEs backwards in time. As a result, we obtain a flow of solutions, the direction of which is represented by the arrows in each figure. In the left figure, this flow of solutions gives us that every change in the value of $\gamma$ from $\hat{\gamma}$ to $\tilde{\gamma}$ results in the recovery of every point of the Pareto front along the one-dimensional curve parameterized by $\gamma \in [\hat{\gamma}, \tilde{\gamma}]$.

### 4.5.4 Identifying the dynamics of the Kraichnan-Orszag system from data

In this example, we demonstrate the versatility of our Riccati-based approach by showing how it can be combined with existing methods to solve more general prob-

lems (see Section 4.4.4). Consider the Kraichnan-Orszag (K-O) system [130, 136]:

$$\begin{cases} \dfrac{dx_1}{d\tau} = x_2 x_3, \\ \dfrac{dx_2}{d\tau} = x_1 x_3, \\ \dfrac{dx_3}{d\tau} = -2x_1 x_2, \end{cases} \tag{4.7}$$

with initial conditions $x_1(0) = 1, x_2(0) = 0.8, x_3(0) = 0.5$. Our goal is to identify the dynamics of the K-O system (i.e. the right-hand side of (4.7)) using measurements of $x_i$ and $\frac{dx_i}{d\tau}, i = 1, 2, 3$ at different times. We identify the dynamics by learning the linear models $\frac{dx_i}{d\tau} = \sum_{k=1}^{n} \theta_k^i \phi_k, i = 1, 2, 3$. Following the general framework of the SINDy method [16], we use the following quadratic basis functions ($n = 10$) for the dynamics:

$$\{\phi_k(x_1, x_2, x_3)\}_{k=1}^{n} = \{1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_2x_3, x_1x_3\}, \tag{4.8}$$

and impose $\ell_1$-regularization on $\boldsymbol{\theta}$ to promote sparse identification of the dynamics. Then, we learn each $\boldsymbol{\theta}^i$ independently and in parallel by minimizing the loss functions

$$\mathcal{L}_i(\boldsymbol{\theta}^i) = \frac{1}{2} \sum_{j=1}^{N} \lambda_j \left[ \left( \frac{dx_i}{d\tau} \right)_j - \sum_{k=1}^{n} \theta_k^i \phi_k((x_1)_j, (x_2)_j, (x_3)_j) \right]^2 + \sum_{k=1}^{n} \gamma_k |\theta_k^i|, \tag{4.9}$$

where $\mathcal{L}_i$ denotes the loss function for equation $i$, $(x_i)_j$ and $(\frac{dx_i}{dt})_j$ denote the measurements of $x_i$ and $\frac{dx_i}{d\tau}$, respectively, at time $\tau_j$, $i = 1, 2, 3, j = 1, ..., N$. Note that this loss function (4.9) corresponds to setting $R = \|\cdot\|_1$ and $\mathbf{x} = 0$ in the linear regression problem (4.9). Thus, solving this learning problem is equivalent to evaluating the solution to the corresponding multi-time HJ PDE at $(0, \lambda_1, \ldots, \lambda_n)$. In our numerical experiments, we generate data points for training and testing by solving (4.7) numerically for $x_1, x_2, x_3 \in [0, 10]$ using a central finite difference scheme to approximate the time derivatives, and set $\lambda_j = 1, \forall j$ and $\gamma_k = 0.1, \forall k$.

Instead of the sparse regression techniques employed by SINDy, we use PDHG to minimize (4.9) (see Section 4.4.4). Each iteration of PDHG involves minimizing a loss function of the form (4.9), but with $\ell_2$-regularization instead of $\ell_1$. Hence, this sub-problem can be solved using our Riccati-based methods. As discussed in Section 4.4.4, note that we only need to apply RK4 for the first iteration of PDHG, and every subsequent iteration can be solved using a change of bias. As such, we do not suffer from any error accumulation related to repeated applications of RK4. In Table 4.4, we see that we do indeed recover a sparse identification of the dynamics. However, we also incorrectly identify non-zero coefficients for the basis functions $x_2$ and $x_3$. We note that this misidentification may be the result of a lack of unique identifiability of the system from the data points sampled. In fact, Table 4.5 shows that the errors in the solution $x_1, x_2, x_3$ of the identified system versus the solution of the true system (4.7) are relatively small, which corroborates that identifiability may have been an issue.

| | 1 | $x_1$ | $x_2$ | $x_3$ | $x_1^2$ | $x_2^2$ | $x_3^2$ | $x_1x_2$ | $x_2x_3$ | $x_1x_3$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $\boldsymbol{\theta}^{1,*}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.9931 | 0 |
| $\boldsymbol{\theta}^{2,*}$ | 0 | 0 | 0 | 0.0160 | 0 | 0 | 0 | 0 | 0 | 0.9761 |
| $\boldsymbol{\theta}^{3,*}$ | 0 | 0 | −0.0165 | 0 | 0 | 0 | 0 | −1.9777 | 0 | 0 |

Table 4.4: Results of sparse identification of the K-O system (4.7) using PDHG to minimize the $\ell_1$-regularized losses (4.9). The true solution is 0 for all entries, except $\boldsymbol{\theta}^{1,*} = 1$ for $x_2x_3$, $\boldsymbol{\theta}^{2,*} = 1$ for $x_1x_3$, and $\boldsymbol{\theta}^{3,*} = -2$ for $x_1x_2$. We recover the dynamics reasonably well, albeit with some slight misidentification of $\boldsymbol{\theta}^{2,*}, \boldsymbol{\theta}^{3,*}$.

| | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| relative $L^2$ error (%) | 0.2144 | 0.3718 | 0.3152 |

Table 4.5: Errors of the solution $x_1, x_2, x_3$ of the system identified using PDHG compared to the true solution of the K-O system. The reference is obtained by numerically solving the true system (4.7) using a central finite difference scheme. These errors indicate that the errors in the system identification in Table 4.4 may be due to a lack of unique identifiability of the system using the given data points.

## 4.6 Summary

In this chapter, we established a novel theoretical connection between certain learning problems and the multi-time Hopf formula. In doing so, we showed that when we solve certain learning problems, we actually solve certain multi-time HJ PDEs and their corresponding optimal control problems. In this work, we focused on the development of the connection between regularized linear regression and the LQR problem. By leveraging this novel connection, we developed new methodology based on solving Riccati ODEs that allows us to design novel training approaches for certain machine learning applications, including continual learning, post-training calibration, hyper-parameter tuning and exploration of the associated Pareto front, and sparse dynamics identification. We also showed that our Riccati-based approach yields some promising computational advantages over conventional learning methods; after the original training, the models learned using our approach can be continually updated without having to retrain the entire model or having access to all of the previous data, which could be particularly useful in continual learning [109, 83, 129] and federated learning [93, 77].

Thus, our novel theoretical connection and our Riccati-based methodology present many exciting opportunities. Some possible future directions are as follows. While our Riccati-based methodology allows us to alter the hyper-parameters and data points used in the learning problem without having to retrain on all previous data, it also requires that the original training be done using our Riccati-based methodology. It would allow for increased versatility if we could more easily combine our Riccati-based approach with other training methods. Additionally, in Sections 4.4.4 and 4.5.4, we showed that our Riccati-based approach allows computations to be reused when using non-quadratic regularizations. However, in this

case, the training process still had to be restarted if the hyper-parameters, dataset, or regularization type is changed. It would allow for more flexibility if we could develop more adaptive processes for changing these aspects of the learning problem when the regularization is not quadratic.

In Section 4.3, we focused on LQR problems, where the dynamics are independent of the trajectory, but it would be worthwhile to investigate what connections LQR problems with general linear dynamics may yield (e.g., LQR problems with general linear dynamics may be reformulated as LQR problems with state-independent dynamics using a change of variable [38]). Another natural extension would be to consider nonlinear models, which currently pose challenges in both scientific machine learning and optimal control and hence, connections drawn in this case would benefit both fields. Alternatively, if we relax our assumption on convexity by allowing for nonconvex loss functions (or equivalently, nonconvex Hamiltonians), we could also extend our connection to be between learning problems and differential games [51] instead of optimal control. However, even with convex losses, there are already many interesting potential applications in machine learning to pursue. For example, the theoretical connection presented in Section 4.2.2 provides a formula for the optimal learning parameters $\boldsymbol{\theta}^*$ in terms of the hyper-parameters. This connection could be leveraged to simplify the bi-level optimization in meta-learning applications to a single-level, constrained optimization problem. As another example, we could generalize our theory to instead consider the viscous HJ PDE, which adds a Laplacian term to the right-hand side of the HJ PDEs (4.1) and (4.4). Then, by leveraging the recently established connection between viscous HJ PDEs and Bayesian modeling [33], we could extend our work to applications in Bayesian inference.

# Chapter Five

# Conclusion

In Part I, we derived new representation formulas for certain classes of optimal control problems (and their corresponding HJ PDEs) with state-dependent running costs (or equivalently, state-dependent Hamiltonians) or non-smooth constraints on the control. We then leveraged these representation formulas to develop efficient numerical solvers for these problems in high dimensions. To demonstrate the potential of these solvers for real-time optimal control applications, we presented FPGA implementations of our solvers that showed promising performance boosts over our already efficient CPU implementations, while also tracking how many logic and memory resources were consumed. Thus, in Part I, we both developed new analytical results and corresponding efficient numerical solvers and gained insight on potential FPGA implementation methodology for real-time optimal control applications.

However, currently our analytical results only apply to the specific classes of HJ PDEs and optimal control problems considered. Thus, it would greatly increase the scope of our results if we could combine our proposed algorithms with other building blocks, such as the solver in [48] and/or the LQR solver, to handle more general classes of problems. Additionally, while our FPGA results yield promising advantages over those of the CPU, we also encountered some challenges in developing the FPGA implementations. For example, the issue of negative slack complicated our FPGA implementation of ADMM in terms of both programming and resource usage. While we proposed some solutions that allowed us to successfully get rid of the slack, the success of these fixes was not always predictable. As such, another important future direction would be to continue exploring FPGA implementations of scientific computing methods more broadly in order to establish more robust methodology for FPGA implementation development.

In Part II, we established a novel theoretical connection between the multi-time Hopf formula and certain optimization problems arising in machine learning. As

such, we showed that when we solve these learning problems, we actually solve a multi-time HJ PDE and its associated optimal control problem. We then leveraged this connection to develop a new Riccati-based approach to solving regularized linear regression problems that shows potential computational and memory advantages over conventional learning approaches. Thus, in Part II, we used existing HJ PDE theory and optimal control techniques to develop new theory and training approaches for machine learning.

In Section 4.6, we listed several exciting future directions for this work. To reiterate, some possible future directions include exploring how our new connection could allow existing machine learning methods to be reused to solve high-dimensional multi-time HJ PDEs or their corresponding optimal control problems, developing potential advantages of our new connection in the case of non-convex Hamiltonians or nonlinear learning models, and extending our connection to consider viscous HJ PDEs or differential games.

# Bibliography


[1] M. Akian, R. Bapat, and S. Gaubert, *Max-plus algebra*, Handbook of linear algebra, 39 (2006).

[2] M. Akian, S. Gaubert, and A. Lakhoua, *The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis*, SIAM Journal on Control and Optimization, 47 (2008), pp. 817–848.

[3] A. Alla, M. Falcone, and L. Saluzzi, *An efficient DP algorithm on a tree-structure for finite horizon optimal control problems*, SIAM Journal on Scientific Computing, 41 (2019), pp. A2384–A2406.

[4] A. Alla, M. Falcone, and S. Volkwein, *Error analysis for POD approximations of infinite horizon problems via the dynamic programming approach*, SIAM Journal on Control and Optimization, 55 (2017), pp. 3091–3115.

[5] S. A. Aĭpanov and Z. N. Murzabekov, *Analytical solution of a linear-quadratic optimal control problem with constraints on the value of the control*, Izv. Ross. Akad. Nauk Teor. Sist. Upr., (2014), pp. 87–94.

[6] A. Bachouch, C. Huré, N. Langrené, and H. Pham, *Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications*, Methodol. Comput. Appl. Probab., 24 (2022), pp. 143–178.

[7] S. Bansal and C. Tomlin, *DeepReach: A Deep Learning Approach to High-Dimensional Reachability*, 2020.

[8] M. Bardi and I. Capuzzo-Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, Systems & Control: Foundations & Applications, Birkhäuser Boston, Inc., Boston, MA, 1997. With appendices by Maurizio Falcone and Pierpaolo Soravia.

[9] M. Bardi and F. Da Lio, *On the Bellman equation for some unbounded control problems*, NoDEA Nonlinear Differential Equations and Applications, 4 (1997), pp. 491–510.

[10] J.-L. Basdevant, *Variational Principles in Physics*, Springer, 2007.

[11] R. E. Bellman, *Adaptive control processes: a guided tour*, Princeton university press, 1961.

[12] D. P. Bertsekas, *Reinforcement learning and optimal control*, Athena Scientific, Belmont, Massachusetts, (2019).

[13] O. Bokanowski, J. Garcke, M. Griebel, and I. Klompmaker, *An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton-Jacobi Bellman equations*, Journal of Scientific Computing, 55 (2013), pp. 575–605.

[14] D. A. Borton, E. M. D. van Rijn, M. T. Harrison, N. R. Provenza, P. Chen, T. Kim, and J. Darbon, *Period-based artifact reconstruction and removal for deep brain stimulation*, 2022.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, Found. Trends Mach. Learn., 3 (2011), p. 1–122.

[16] S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proceedings of the National Academy of Sciences, 113 (2016), pp. 3932–3937.

[17] R. S. Burachik, C. Y. Kaya, and S. N. Majeed, *A Duality Approach for Solving Control-Constrained Linear-Quadratic Optimal Control Problems*, SIAM Journal on Control and Optimization, 52 (2014), pp. 1423–1456.

[18] J. C. Butcher, *Numerical methods for ordinary differential equations*, John Wiley & Sons, 2016.

[19] M. Cannon, W. Liao, and B. Kouvaritakis, *Efficient MPC Optimization using Pontryagin's Minimum Principle*, in Proceedings of the 45th IEEE Conference on Decision and Control, 2006, pp. 5459–5464.

[20] A. Chambolle and T. Pock, *A first-order primal-dual algorithm for convex problems with applications to imaging*, Journal of Mathematical Imaging and Vision, 40 (2011), pp. 120–145.

[21] J. Chen, W. Zhan, and M. Tomizuka, *Constrained iterative LQR for on-road autonomous driving motion planning*, in 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017, pp. 1–7.

[22] ——, *Autonomous Driving Motion Planning With Constrained Iterative LQR*, IEEE Transactions on Intelligent Vehicles, 4 (2019), pp. 244–254.

[23] M. Chen, Q. Hu, J. F. Fisac, K. Akametalu, C. Mackin, and C. J. Tomlin, *Reachability-Based Safety and Goal Satisfaction of Unmanned Aerial Platoons on Air Highways*, Journal of Guidance, Control, and Dynamics, 40 (2017), pp. 1360–1373.

[24] P. Chen, J. Darbon, and T. Meng, *Lax-Oleinik-type formulas and efficient algorithms for certain high-dimensional optimal control problems*, arXiv preprint arXiv:2109.14849, (2021).

[25] ——, *Hopf-type representation formulas and efficient algorithms for certain high-dimensional optimal control problems*, arXiv preprint arXiv:2110.02541, (2023).

[26] P. Chen, T. Kim, E. Dastin-van Rijn, N. R. Provenza, S. A. Sheth, W. K. Goodman, D. A. Borton, M. T. Harrison, and J. Darbon, *Periodic artifact removal with applications to deep brain stimulation*, IEEE Transactions on Neural Systems and Rehabilitation Engineering, 30 (2022), pp. 2692–2699.

[27] P. Chen, T. Meng, Z. Zou, J. Darbon, and G. E. Karniadakis, *Leveraging Multi-time Hamilton-Jacobi PDEs for Certain Scientific Machine Learning Problems*, arXiv preprint arXiv:2303.12928, (2023).

[28] Y. T. Chow, J. Darbon, S. Osher, and W. Yin, *Algorithm for overcoming the curse of dimensionality for state-dependent hamilton-jacobi equations*, Journal of Computational Physics, 387 (2019), pp. 376–409.

[29] J. Cong, J. Lau, G. Liu, S. Neuendorffer, P. Pan, K. Vissers, and Z. Zhang, *FPGA HLS Today: Successes, Challenges, and Opportunities*, ACM Transactions on Reconfigurable Technology and Systems, 15 (2022).

[30] M. Coupechoux, J. Darbon, J. Kélif, and M. Sigelle, *Optimal Trajectories of a UAV Base Station Using Lagrangian Mechanics*, in IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2019, pp. 626–631.

[31] J. Darbon, *On convex finite-dimensional variational methods in imaging sciences and Hamilton–Jacobi equations*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 2268–2293.

[32] J. Darbon, P. M. Dower, and T. Meng, *Neural network architectures using min-plus algebra for solving certain high-dimensional optimal control problems and Hamilton–Jacobi PDEs*, Mathematics of Control, Signals, and Systems, (2022), pp. 1–44.

[33] J. Darbon and G. Langlois, *On Bayesian Posterior Mean Estimators in Imaging Sciences and Hamilton–Jacobi Partial Differential Equations*, J. Math Imaging Vis., 63 (2021), p. 821–854.

[34] J. Darbon, G. P. Langlois, and T. Meng, *Overcoming the curse of dimensionality for some Hamilton-Jacobi partial differential equations via neural network architectures*, Res. Math. Sci., 7 (2020), p. 20.

[35] J. Darbon and T. Meng, *On Decomposition Models in Imaging Sciences and Multi-time Hamilton–Jacobi Partial Differential Equations*, SIAM Journal on Imaging Sciences, 13 (2020), pp. 971–1014.

[36] J. Darbon and T. Meng, *On some neural network architectures that can represent viscosity solutions of certain high dimensional Hamilton–Jacobi partial differential equations*, Journal of Computational Physics, 425 (2021), p. 109907.

[37] J. Darbon, T. Meng, and E. Resmerita, *On Hamilton–Jacobi PDEs and Image Denoising Models with Certain Nonadditive Noise*, Journal of Mathematical Imaging and Vision, 64 (2022), pp. 408–441.

[38] J. Darbon and S. Osher, *Algorithms for overcoming the curse of dimensionality for certain Hamilton-Jacobi equations arising in control theory and elsewhere*, Res Math Sci Research in the Mathematical Sciences, 3 (2016), pp. 1–26.

[39] D. Davis and W. Yin, *Faster Convergence Rates of Relaxed Peaceman-Rachford and ADMM Under Regularity Assumptions*, Mathematics of Operations Research, 42 (2017), pp. 783–805.

[40] M. A. de Abreu de Sousa, R. Pires, and E. Del-Moral-Hernandez, *Somprocessor: A high throughput fpga-based architecture for implementing self-organizing maps and its application to video processing*, Neural Networks, 125 (2020), pp. 349–362.

[41] J. de Fine Licht, G. Kwasniewski, and T. Hoefler, *Flexible communication avoiding matrix multiplication on fpga with high-level synthesis*, in The 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'20), 2020.

[42] D. Delahaye, S. Puechmorel, P. Tsiotras, and E. Feron, *Mathematical models for aircraft trajectory design: A survey*, in Air Traffic Management and Systems, Tokyo, 2014, Springer Japan, pp. 205–247.

[43] W. Deng and W. Yin, *On the global and linear convergence of the generalized alternating direction method of multipliers*, J. Sci. Comput., 66 (2016), pp. 889–916.

[44] J. Denk and G. Schmidt, *Synthesis of a walking primitive database for a humanoid robot using optimal control techniques*, in Proceedings of IEEE-RAS International Conference on Humanoid Robots, 2001, pp. 319–326.

[45] S. Desai, M. Mattheakis, H. Joy, P. Protopapas, and S. Roberts, *One-shot transfer learning of physics-informed neural networks*, arXiv preprint arXiv:2110.11286, (2021).

[46] B. Djeridane and J. Lygeros, *Neural approximation of PDE solutions: An application to reachability computations*, in Proceedings of the 45th IEEE Conference on Decision and Control, Dec 2006, pp. 3034–3039.

[47] S. Dolgov, D. Kalise, and K. Kunisch, *Tensor Decomposition Methods for High-dimensional Hamilton–Jacobi–Bellman Equations*, SIAM Journal on Scientific Computing, 43 (2021), pp. A1625–A1650.

[48] P. M. Dower, W. M. McEneaney, and M. Cantoni, *Game representations for state constrained continuous time linear regulator problems*, arXiv preprint arXiv:1904.05552, (2019).

[49] P. M. Dower, W. M. McEneaney, and H. Zhang, *Max-plus fundamental solution semigroups for optimal control problems*, in 2015 Proceedings of the Conference on Control and its Applications, SIAM, 2015, pp. 368–375.

[50] A. El Khoury, F. Lamiraux, and M. Taïx, *Optimal motion planning for humanoid robots*, in 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 3136–3141.

[51] L. C. Evans and P. E. Souganidis, *Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations*, Indiana University mathematics journal, 33 (1984), pp. 773–797.

[52] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. P. D'Arpino, R. Deits, M. DiCicco, D. Fourie, et al., *An architecture for online affordance-based perception and whole-body planning*, Journal of Field Robotics, 32 (2015), pp. 229–254.

[53] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, *Optimization based full body control for the atlas robot*, in 2014 IEEE-RAS International Conference on Humanoid Robots, 2014, pp. 120–127.

[54] Feng Lin and R. D. Brandt, *An optimal control approach to robust control of robot manipulators*, IEEE Transactions on Robotics and Automation, 14 (1998), pp. 69–77.

[55] W. Fleming and W. McEneaney, *A max-plus-based algorithm for a Hamilton–Jacobi–Bellman equation of nonlinear filtering*, SIAM Journal on Control and Optimization, 38 (2000), pp. 683–710.

[56] K. Fujiwara, S. Kajita, K. Harada, K. Kaneko, M. Morisawa, F. Kanehiro, S. Nakaoka, and H. Hirukawa, *An optimal planning of falling motions of a humanoid robot*, in 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 456–462.

[57] J. Garcke and A. Kröner, *Suboptimal feedback control of PDEs by solving HJB equations on adaptive sparse grids*, Journal of Scientific Computing, 70 (2017), pp. 1–28.

[58] S. Gaubert, W. McEneaney, and Z. Qu, *Curse of dimensionality reduction in max-plus based approximation methods: Theoretical estimates and improved pruning algorithms*, in 2011 50th IEEE Conference on Decision and Control and European Control Conference, IEEE, 2011, pp. 1054–1061.

[59] R. Glowinski, *On Alternating Direction Methods of Multipliers: A Historical Perspective*, Springer Netherlands, Dordrecht, 2014, pp. 59–82.

[60] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[61] S. Goswami, K. Kontolati, M. D. Shields, and G. E. Karniadakis, *Deep transfer operator learning for partial differential equations under conditional shift*, Nature Machine Intelligence, (2022), pp. 1–10.

[62] J. Han, A. Jentzen, and W. E, *Solving high-dimensional partial differential equations using deep learning*, Proceedings of the National Academy of Sciences, 115 (2018), pp. 8505–8510.

[63] J.-B. Hiriart-Urruty and C. Lemarechal, *Convex Analysis and Minimization Algorithms I: Fundamentals*, vol. 305, Springer-Verlag Berlin Heidelberg, 1993.

[64] M. Hofer, M. Muehlebach, and R. D'Andrea, *Application of an approximate model predictive control scheme on an unmanned aerial vehicle*, in 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 2952–2957.

[65] E. Hopf, *Generalized Solutions of non-linear Equations of First Order*, Journal of Mathematics and Mechanics, 14 (1965), pp. 951–973.

[66] M. B. Horowitz, A. Damle, and J. W. Burdick, *Linear Hamilton Jacobi Bellman equations in high dimensions*, in 53rd IEEE Conference on Decision and Control, IEEE, 2014, pp. 5880–5887.

[67] C. Hu and C.-W. Shu, *A Discontinuous Galerkin Finite Element Method for Hamilton–Jacobi Equations*, SIAM Journal on Scientific Computing, 21 (1999), pp. 666–690.

[68] C. Huré, H. Pham, A. Bachouch, and N. Langrené, *Deep neural networks algorithms for stochastic control problems on finite horizon: convergence analysis*, SIAM J. Numer. Anal., 59 (2021), pp. 525–557.

[69] C. Huré, H. Pham, and X. Warin, *Deep backward schemes for high-dimensional nonlinear PDEs*, Math. Comp., 89 (2020), pp. 1547–1579.

[70] H. Jaddu, *Spectral method for constrained linear–quadratic optimal control*, Mathematics and Computers in Simulation, 58 (2002), pp. 159 – 169.

[71] F. Jiang, G. Chou, M. Chen, and C. J. Tomlin, *Using neural networks to compute approximate and guaranteed feasible Hamilton-Jacobi-Bellman PDE solutions*, arXiv preprint arXiv:1611.03158, (2016).

[72] G. Jiang and D. Peng, *Weighted ENO Schemes for Hamilton–Jacobi Equations*, SIAM Journal on Scientific Computing, 21 (2000), pp. 2126–2143.

[73] L. Jin, S. Li, J. Yu, and J. He, *Robot manipulator control using neural networks: A survey*, Neurocomputing, 285 (2018), pp. 23 – 34.

[74] P. Jin, Z. Zhang, I. G. Kevrekidis, and G. E. Karniadakis, *Learning Poisson Systems and Trajectories of Autonomous Systems via Poisson Neural Networks*, IEEE Transactions on Neural Networks and Learning Systems, (2022), pp. 1–13.

[75] P. Jin, Z. Zhang, A. Zhu, Y. Tang, and G. E. Karniadakis, *SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems*, Neural Networks, 132 (2020), pp. 166–179.

[76] Y. Jin and B. Sendhoff, *Pareto-based multiobjective machine learning: An overview and case studies*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38 (2008), pp. 397–415.

[77] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., *Advances and open problems in federated learning*, Foundations and Trends® in Machine Learning, 14 (2021), pp. 1–210.

[78] D. Kalise, S. Kundu, and K. Kunisch, *Robust feedback control of nonlinear PDEs by numerical approximation of high-dimensional Hamilton–Jacobi–Isaacs equations*, SIAM Journal on Applied Dynamical Systems, 19 (2020), pp. 1496–1524.

[79] D. Kalise and K. Kunisch, *Polynomial approximation of high-dimensional Hamilton–Jacobi–Bellman equations and applications to feedback control of semilinear parabolic PDEs*, SIAM Journal on Scientific Computing, 40 (2018), pp. A629–A652.

[80] W. Kang and L. C. Wilcox, *Mitigating the curse of dimensionality: sparse grid characteristics method for optimal feedback control and HJB equations*, Computational Optimization and Applications, 68 (2017), pp. 289–315.

[81] R. Kastner, J. Matai, and S. Neuendorffer, *Parallel Programming for FPGAs*, ArXiv e-prints, (2018).

[82] Y. H. Kim, F. L. Lewis, and D. M. Dawson, *Intelligent optimal control of robotic manipulators using neural networks*, Automatica, 36 (2000), pp. 1355 – 1364.

[83] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., *Overcoming catastrophic forgetting in neural networks*, Proceedings of the National Academy of Sciences, 114 (2017), pp. 3521–3526.

[84] V. N. Kolokoltsov and V. P. Maslov, *Idempotent analysis and its applications*, vol. 401 of Mathematics and its Applications, Kluwer Academic Publishers Group, Dordrecht, 1997. Translation of ıt Idempotent analysis and its application in optimal control (Russian), "Nauka" Moscow, 1994 [MR1375021 (97d:49031)], Translated by V. E. Nazaikinskii, With an appendix by Pierre Del Moral.

[85] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, *Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot*, Autonomous Robots, 40 (2016), pp. 429–455.

[86] K. Kunisch, S. Volkwein, and L. Xie, *HJB-POD-based feedback design for the optimal control of evolution problems*, SIAM Journal on Applied Dynamical Systems, 3 (2004), pp. 701–722.

[87] P. Lambrianides, Q. Gong, and D. Venturi, *A new scalable algorithm for computational optimal control under uncertainty*, J. Comput. Phys., 420 (2020), pp. 109710, 19.

[88] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521 (2015), pp. 436–444.

[89] D. Lee and C. J. Tomlin, *A Computationally Efficient Hamilton-Jacobi-based Formula for State-Constrained Optimal Control Problems*, arXiv e-prints, (2021), p. arXiv:2106.13440.

[90] D. Lee and C. J. Tomlin, *A Hopf-Lax Formula in Hamilton–Jacobi Analysis of Reach-Avoid Problems*, IEEE Control Systems Letters, 5 (2021), pp. 1055–1060.

[91] F. L. Lewis, D. M. Dawson, and C. T. Abdallah, *Robot Manipulator Control: Theory and Practice*, Control engineering, Marcel Dekker, 2004.

[92] A. Li, S. Bansal, G. Giovanis, V. Tolani, C. Tomlin, and M. Chen, *Generating Robust Supervision for Learning-Based Visual Navigation Using Hamilton-Jacobi Reachability*, in Proceedings of the 2nd Conference on Learning for Dynamics and Control, A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, eds., vol. 120 of Proceedings of Machine Learning Research, The Cloud, 10–11 Jun 2020, PMLR, pp. 500–510.

[93] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, *Federated learning: Challenges, methods, and future directions*, IEEE Signal Processing Magazine, 37 (2020), pp. 50–60.

[94] W. Li and E. Todorov, *Iterative linear quadratic regulator design for nonlinear biological movement systems*, in ICINCO (1), 2004, pp. 222–229.

[95] P. L. Lions and J.-C. Rochet, *Hopf Formula and Multitime Hamilton-Jacobi Equations*, Proceedings of the American Mathematical Society, 96 (1986), pp. 79–84.

[96] J. Ma, Z. Cheng, X. Zhang, M. Tomizuka, and T. H. Lee, *Alternating Direction Method of Multipliers for Constrained Iterative LQR in Autonomous Driving*, IEEE Transactions on Intelligent Transportation Systems, (2022), pp. 1–12.

[97] W. McEneaney, *A Curse-of-Dimensionality-Free Numerical Method for Solution of Certain HJB PDEs*, SIAM Journal on Control and Optimization, 46 (2007), pp. 1239–1276.

[98] W. M. McEneaney, *Max-plus methods for nonlinear control and estimation*, Systems & Control: Foundations & Applications, Birkhäuser Boston, Inc., Boston, MA, 2006.

[99] W. M. McEneaney, A. Deshpande, and S. Gaubert, *Curse-of-complexity attenuation in the curse-of-dimensionality-free method for HJB PDEs*, in 2008 American Control Conference, IEEE, 2008, pp. 4684–4690.

[100] W. M. McEneaney and L. J. Kluberg, *Convergence rate for a curse-of-dimensionality-free method for a class of HJB PDEs*, SIAM Journal on Control and Optimization, 48 (2009), pp. 3052–3079.

[101] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*, MIT press, 2018.

[102] J. J. Moreau, *Proximité et dualité dans un espace hilbertien*, Bulletin de la Société Mathématique de France, 93 (1965), pp. 273–299.

[103] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, *Adaptive deep learning for high-dimensional Hamilton-Jacobi-Bellman equations*, SIAM Journal on Scientific Computing, 43 (2021), pp. A1221–A1247.

[104] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, *QRnet: Optimal Regulator Design With LQR-Augmented Neural Networks*, IEEE Control Systems Letters, 5 (2021), pp. 1303–1308.

[105] K. N. Niarchos and J. Lygeros, *A Neural Approximation to Continuous Time Reachability Computations*, in Proceedings of the 45th IEEE Conference on Decision and Control, Dec 2006, pp. 6313–6318.

[106] A. R. Omondi and J. C. Rajapakse, eds., *FPGA Implementations of Neural Networks*, Springer, New York, 2010.

[107] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto, *A Neural Network Approach for High-Dimensional Optimal Control Applied to Multiagent Path Finding*, IEEE Transactions on Control Systems Technology, (2022), pp. 1–17.

[108] S. Osher and C.-W. Shu, *High-Order Essentially Nonoscillatory Schemes for Hamilton-Jacobi Equations*, SIAM Journal on Numerical Analysis, 28 (1991), pp. 907–922.

[109] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, *Continual lifelong learning with neural networks: A review*, Neural networks, 113 (2019), pp. 54–71.

[110] J. H. Park, S. Han, and W. H. Kwon, *LQ tracking controls with fixed terminal states and their application to receding horizon controls*, Systems & Control Letters, 57 (2008), pp. 772 – 777.

[111] M. Parker, *Digital Signal Processing 101: Everything You Need to Know to Get Started*, Newnes, 1 ed., 2010.

[112] C. Parzani and S. Puechmorel, *On a Hamilton-Jacobi-Bellman approach for coordinated optimal aircraft trajectories planning*, in CCC 2017 36th Chinese Control Conference, Control Conference (CCC), 2017 36th Chinese, Dalian, China, July 2017, IEEE, pp. ISBN: 978–1–5386–2918–5.

[113] S. O. Popescu, G. Budura, and A. S. Gontean, *Review of psk and qam — digital modulation techniques on fpga*, in 2010 International Joint Conference on Computational Cybernetics and Technical Informatics, 2010, pp. 327–332.

[114] S. K. Prakash, *Managing HBM's bandwidth in Multi-Die FPGAs using Overlay NoCs*, Master's thesis, University of Waterloo, 2021.

[115] A. F. Psaros, X. Meng, Z. Zou, L. Guo, and G. E. Karniadakis, *Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons*, Journal of Computational Physics, (2023), p. 111902.

[116] M. Raissi, P. Perdikaris, and G. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics, 378 (2019), pp. 686–707.

[117] C. Reisinger and Y. Zhang, *Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems*, Anal. Appl. (Singap.), 18 (2020), pp. 951–999.

[118] J.-C. Rochet, *The taxation principle and multi-time Hamilton-Jacobi equations*, Journal of Mathematical Economics, 14 (1985), pp. 113–128.

[119] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*, vol. 317 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Springer-Verlag, Berlin, 1998.

[120] V. Rubies-Royo and C. Tomlin, *Recursive regression with neural networks: Approximating the HJI PDE solution*, arXiv preprint arXiv:1611.02739, (2016).

[121] A. Rucco, P. B. Sujit, A. P. Aguiar, J. B. de Sousa, and F. L. Pereira, *Optimal Rendezvous Trajectory for Unmanned Aerial-Ground Vehicles*, IEEE Transactions on Aerospace and Electronic Systems, 54 (2018), pp. 834–847.

[122] S. J. Russell, *Artificial intelligence: A modern approach*, Pearson Education, Inc., 2010.

[123] D. Russo, *Adaptation of High Performance and High Capacity Reconfigurable Systems to OpenCL Programming Environments*, Master's thesis, Universitat Politècnica de València, 2020.

[124] A. Shiri and G. K. Khosroshahi, *An fpga implementation of singular value decomposition*, in 2019 27th Iranian Conference on Electrical Engineering (ICEE), 2019, pp. 416–422.

[125] A. Sideris and J. E. Bobrow, *An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems*, in Proceedings of the 2005, American Control Conference, 2005., vol. 4, 2005, pp. 2275–2280.

[126] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, Journal of Computational Physics, 375 (2018), pp. 1339 – 1364.

[127] E. Todorov, *Efficient computation of optimal actions*, Proceedings of the National Academy of Sciences, 106 (2009), pp. 11478–11483.

[128] H. L. Trentelman, A. A. Stoorvogel, and M. Hautus, *Control Theory for Linear Systems*, Springer-Verlag London, 2001.

[129] G. M. Van de Ven and A. S. Tolias, *Three scenarios for continual learning*, arXiv preprint arXiv:1904.07734, (2019).

[130] X. Wan and G. E. Karniadakis, *Multi-element generalized polynomial chaos for arbitrary probability measures*, SIAM Journal on Scientific Computing, 28 (2006), pp. 901–928.

[131] S. Weisberg, *Applied linear regression*, vol. 528, John Wiley & Sons, 2005.

[132] I. Yegorov and P. M. Dower, *Perspectives on characteristics based curse-of-dimensionality-free numerical approaches for solving Hamilton–Jacobi equations*, Applied Mathematics & Optimization, (2017), pp. 1–49.

[133] M. Zhou, J. Han, and J. Lu, *Actor-Critic Method for High Dimensional Static Hamilton–Jacobi–Bellman Partial Differential Equations based on Neural Networks*, SIAM Journal on Scientific Computing, 43 (2021), pp. A4043–A4066.

[134] L. Zhuo and V. Prasanna, *High performance linear algebra operations on reconfigurable systems*, in SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, 2005, pp. 2–2.

[135] Z. Zou and G. E. Karniadakis, *L-HYDRA: Multi-Head Physics-Informed Neural Networks*, arXiv preprint arXiv:2301.02152, (2023).

[136] Z. Zou, X. Meng, A. F. Psaros, and G. E. Karniadakis, *NeuralUQ: A comprehensive library for uncertainty quantification in neural differential equations and operators*, arXiv preprint arXiv:2208.11866, (2022).