Novel optimization algorithms for evaluating solution to certain high dimension (multi-time) Hamilton-Jacobi PDEs arising from optimal control

by

Taewoo Kim

B.Sc., Seoul National University, Seoul, Republic of Korea, 2015

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 2022

This dissertation by Taewoo Kim 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_____ _____

Jerome Darbon, Ph.D., Advisor

Recommended to the Graduate Council

Date_____ _____

Chi-Wang shu, Ph.D., Reader

Date_____ _____

Johnny Guzman, Ph.D., Reader

Approved by the Graduate Council

Date_____ _____

Andrew G. Campbell, Dean of the Graduate School

Vita

Taewoo Kim was born in Seoul, Republic of Korea in Sep 1990. He went to Seoul National University in 2009 for the undergraduate study in Mathematics, and graduated in August 2015.

In September 2015, he enrolled as a Ph.D. student in the Division of Applied Mathematics at Brown University. While at Brown, he has worked under the supervision of Prof. Jerome Darbon.

Acknowledgments

I would first like to thank my advisor, Prof. Jerome Darbon, for being supportive. His guidance and advice carried me through all the stages of my research. I would like to thank Prof. Chi-Wang Shu and Prof. Johnny Guzman in my thesis committee for their valuable guidance throughout my studies. I would also like to thank my collegues Tingwei Meng , Gabriel Provencher Langlois and Paula Chen.

Last, but not the least, I want to thank my family for the long-time support.

Abstract of "Novel optimization algorithms for evaluating solution to certain high dimension (multi-time) Hamilton-Jacobi PDEs arising from optimal control", by Taewoo Kim, Ph.D., Brown University, May 2022

In this thesis, we study optimization algorithms for evaluating solutions to certain high dimension (multi-time) Hamilton-Jacobi Partial Differential equations arising from optimal control. There are three main chapters to cover these topics.

The first chapter describes known connections between optimal control problems and Hamilton-Jacobi PDEs and the generalized Hopf formula that corresponds to the viscosity solution of certain Hamilton-Jacobi PDEs. We then focus on an important class of optimal control problem where the control has to remain in a full ellipsoid and the terminal cost is a convex function. We show that these problems can be solved using standard convex optimization methods. The efficiency of these optimization methods rely on the performance of computing the projection on the Minkowsky sum of ellipsoids. We also describe connections between the generalized Hopf formula and multi-time Hamilton-Jacobi PDEs.

The second chapter presents a novel and efficient Newton based algorithm to compute the projection onto the Minkowski sum of full ellipsoids.

The third chapter extends the class of linear optimal control problems considered in this thesis by allowing the control to live in a degenerate ellipsoid. Several possible algorithms to cope with these degenerate cases are presented.

Finally, the last chapter focuses on developing the algorithm for computing the proximal point of a convex polyhedral function whose domain is a polyhedral set. This algorithm finds applications in optimal control and multi-time Hamilton Jacobi PDEs.

Contents

Chapter One

Introduction

In this chapter, we first describe the main challenge of solving high dimensional Hamilton-Jacobi Partial differential equations (HJ PDEs): it is known as the "curse of dimensionality". Then we present classical and fundamental connections between optimal control problems and viscosity solutions to HJ PDEs in section 1.2. Section 1.3 describes the class of optimal control problems that we consider in this manuscript and its associated HJ PDE. The considered class of optimal control problems is relevant to the US Navy [58, 60]. Following [58, 60], we also present the generalized Hopf formula which corresponds to a variational representation formula of the viscosity solution to these HJ PDEs. Section 1.4 considers discretizations of the generalized Hopf formula which yield finite dimensional convex optimization problems. Section 1.5 considers standard state of the art convex optimization methods for solving convex optimization problems that arise from the discretization of the generalized Hopf formula associated to the class of optimal control problems considered in this thesis. We show that the performance of these optimization algorithms crucially depend on the efficiency of numerical algorithms to compute the projection on the Minkowski sum of ellipsoids. In section 1.6 we show that discretizations of the generalized Hopf formula corresponds to solving multi-time HJ PDEs and present some numerical results. Finally, section 1.7 describes the content and contributions of the other chapters of this thesis.

1.1 Context and motivations: Mitigating the curse of dimensionality

It is well known that time-dependant Hamilton-Jacobi partial differential equations (HJ PDEs), play an important role in analyzing continuous dynamic games and control theory problems [32]. Unfortunately, standard grid-based numerical algorithms for PDEs are generally computationally impractical when the dimension is higher than 5. Such algorithms employ grids to discretize the spatial and time domain, and the number of grid points required to evaluate accurately solutions of PDEs grows exponentially with the dimension. It is therefore essentially impossible in practice to numerically solve PDEs in high dimension using grid-based algorithms, even with sophisticated high-order accuracy methods for HJ PDEs such as ENO [78], WENO [52], and DG [48]. This problem is known as the curse of dimensionality [10].

Several numerical approaches have been proposed to overcome or mitigate the curse of dimensionality arising from certain classes of HJ PDEs. These include, but are not limited to, max-plus algebra methods [1, 2, 30, 36, 39, 70, 71, 72, 73], dynamic programming and reinforcement learning [3, 11], tensor decomposition techniques [29, 47, 89], sparse grids [12, 38, 56], model order reduction [4, 62], polynomial approximation [54, 55], optimization methods [19, 23, 25, 90] and neural networks [22, 24, 6, 28, 51, 41, 49, 50, 66, 77, 81, 84, 86, 88].

In this thesis, we propose novel numerical algorithms for solving large classes of high dimensional HJ PDEs that arise from certain optimal control problems without the use of grids or numerical approximations. The class of optimal control problems considered in this thesis are relevant to the US Navy [58, 60].

1.2 Optimal control problems and Hamilton-Jacobi PDEs

First, we introduce some mathematical notations that will be used in this thesis. Then, we present well-known connections between optimal control problems and viscosity solutions to Hamilton-Jacobi PDEs.

1.2.1 Mathematical notations

We denote by $\mathbb{R}^n$ the n-dimensional Euclidean space. The Euclidean scalar product and Euclidean norm on $\mathbb{R}^n$ are denoted by $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$, respectively. We also use $\| \cdot \|_1$, $\| \cdot \|_\infty$, and $\| \cdot \|_E$ to denote the ℓ_1-norm, ℓ_∞-norm, and the norm induced by a symmetric positive definite matrix E, respectively. They are defined as follows for all $x = (x_1, \dots, x_n) \in \mathbb{R}^n$

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|, \ \|x\|_\infty = \max\{|x_i|, i = 1, \dots, n\}, \ \|x\|_E = \sqrt{\langle x, Ex \rangle}.$$

We denote the set of real matrices with m rows and n columns by $\mathcal{M}_{m,n}(\mathbb{R})$, and we identify any vector in $\mathbb{R}^n$ with a matrix with n rows and 1 column. We also list some common notations and definitions used throughout this thesis in Table 1.1. Other definitions and theorems from convex analysis that are used in this thesis are given in Appendix A.1.

Table 1.1: Notation used in this thesis. In this table, unless otherwise specified, we use C to denote a set in $\mathbb{R}^n$, f to denote a function from $\mathbb{R}^n$ to $\mathbb{R} \cup \{+\infty\}$, x to denote a vector in $\mathbb{R}^n$ and M to denote an $m \times n$ matrix in $\mathcal{M}_{m,n}(\mathbb{R})$. For simplicity, we omit the assumptions in the definition.

Notation	**Meaning**	**Definition**
$\langle \cdot, \cdot \rangle$	Euclidean scalar product in $\mathbb{R}^n$	$\langle x, y \rangle := \sum_{i=1}^n x_i y_i$
$\|\cdot\|_2$	Euclidean norm in $\mathbb{R}^n$	$\|x\|_2 := \sqrt{\langle x, x \rangle}$
$\|\cdot\|_A$	Vector norm in $\mathbb{R}^n$	$\|x\|_A := \sqrt{\langle Ax, x \rangle}$
$\|\cdot\|_2$	Matrix norms in $\mathbb{R}^n \times \mathbb{R}^n$ induced by Euclidean norm	$\|A\|_2 := \sup \{\|Ax\|_2 \text{ with } \|x\|_2 = 1\}$
$\|\cdot\|_F$	Frobenius norm in $\mathbb{R}^n \times \mathbb{R}^n$	$\|A\|_F := \sqrt{\sum_{i,j=1,\cdots,n} \|a_{ij}\|^2}$
dom f	The domain of f	$\{x \in \mathbb{R}^n : f(x) < +\infty\}$
$\Gamma_0(\mathbb{R}^n)$	A useful and standard class of convex functions	The set containing all proper, convex, lower semicontinuous functions from $\mathbb{R}^n$ to $\mathbb{R} \cup \{+\infty\}$
f^*	Fenchel–Legendre transform of f	$f^*(p) := \sup_{x \in \mathbb{R}^n} \{\langle p, x \rangle - f(x)\}$
$f \square g$	Inf-convolution of f and g	$f \square g(x) = \inf_{u \in \mathbb{R}^n} \{f(u) + g(x - u)\}$
M^T	Matrix transpose of M	$[M^T]_{ij} = [M]_{ji}$
I_C	Indicator function of C	If $x \in C$, then define $I_C(x) := 0$. Otherwise, define $I_C(x) := +\infty$.
$\Pi_C(x)$	Projection of x onto C	$\Pi_C(x) := \arg\min_{u \in C} \|x - u\|$
$\partial f(x)$	Subdifferential of f at x	$\{p \in \mathbb{R}^n : f(y) \geqslant f(x) + \langle p, y - x \rangle \ \forall y \in \mathbb{R}^n\}$
$f'(x; d)$	Directional derivative of f at x along the direction d	$\lim_{h \to 0^+} \frac{1}{h}(f(x + hd) - f(x))$
cone C	Convex cone generated by C	The set containing all conic combinations of elements of C.
conv C	Convex hull of C	The set containing all convex combinations of the elements of C.
C°	Polar cone of a closed convex cone C	$C^\circ := \{y \in \mathbb{R}^n : \langle y, x \rangle \leq 0 \ \forall x \in C\}$
epi f	Epigraph of f	$\{(x, t) \in \mathbb{R}^n \times \mathbb{R} : x \in \text{dom } f, t \geq f(x)\}$

1.2.2 Value function in optimal control theory and Hamilton-Jacobi PDEs

We now briefly describe classical theoretical connections between optimal control theory and HJ PDEs [32]. Suppose we are given a fixed terminal time $T \in \mathbb{R}$, an initial time $t < T$ with an initial data $x \in \mathbb{R}^n$. We consider the solution $y : [t, T] \to \mathbb{R}^n$ of the following ordinary differential equation

$$\begin{cases} \dfrac{dy}{ds}(s) = f(y(s), u(s)) & \text{in } (t, T), \\ y(t) = x. \end{cases}$$

where $f : \mathbb{R}^n \times C \to \mathbb{R}^n$ is Lipschitz and C is a compact set of $\mathbb{R}^m$. The solution of the above ordinary differential equation depends on the measurable function $u : (-\infty, T] \to C$ which is called control. For each given control u, we consider a "running cost" along the trajectory that reads

$$\int_t^T L(y(s), u(s))ds$$

where $L : \mathbb{R}^n \times C$ is referred to as Lagrangian and a "terminal cost" that only depends on the final state $y(T)$ which reads

$$J(y(T)).$$

Optimal control problems consist of finding controls u that minimize the sum of the running cost plus the terminal cost given the initial state $x \in \mathbb{R}^n$ and initial

time t. Therefore we can define the value function $V : \mathbb{R}^n \times (-\infty, T) \to \mathbb{R}$ as follows

$$V(x,t) = \inf_{u(\cdot)} \underbrace{\int_t^T L(y(s), u(s))ds}_{\text{Running cost}} + \underbrace{J(\mathrm{y}(T))}_{\text{Terminal cost}}$$

It is well known (see e.g. [32]) the value function V satisfies the following HJ PDE (in the viscosity sense) with a terminal condition

$$\begin{cases} \dfrac{\partial V}{\partial t}(x,t) - H(\nabla_x V(x,t), x) = 0, \\ V(x,T) = J(x), \end{cases}$$

where the Hamiltonian $H(p,x)$ is defined by

$$H(p,x) = \max_{v \in C} \left(-\langle p, f(x,v) \rangle - L(x,a) \right).$$

In order to consider HJ PDEs with initial data, we can "reverse" time and consider the function $S(x,t) = V(x, T-t)$. Then, the function S is the viscosity solution of the following HJ PDE with initial value

$$\begin{cases} \dfrac{\partial S}{\partial t}(x,t) + H(\nabla_x S(x,t), x) = 0 \\ S(x,0) = J(x). \end{cases}$$

Our goal is to compute the value of the viscosity solution $S(x,t)$ and the spacial gradient $\nabla_x S(x,t)$ (when it exists) for each given (x,t). Then, we can recover the optimal control at (x,t) by solving the following optimization problem

$$\max_{a \in C} \left(-\langle \nabla_x S(x,t), f(x,a) - L(x,a) \rangle \right) = H(\nabla_x S(x,t), x).$$

Therefore, to numerically solve optimal control problems, we need to be able to compute the solution of the HJ PDE but also its spatial gradient in order to recover the optimal control.

1.3 Optimal control of certain linear dynamical system and the generalized Hopf formula

In this thesis, we consider a special class of optimal control problems that are relevant to the US Navy [58, 60]. It corresponds to controlling certain linear dynamical systems where the running cost takes a specific form. This class of optimal control problems is associated to HJ PDEs that enjoy a variational representation formula known as the generalized Hopf formula. We now present the class of optimal control problems considered in this thesis and present the generalized Hopf representation formula as described in [58, 60].

As in the previous section, we suppose we are given a fixed terminal time $T \in \mathbb{R}$, an initial time $t < T$ with an initial data $x \in \mathbb{R}^n$. Now we consider $f(y(s), u(s)) = My(s) + N(s)u(s)$ where M is $n \times n$ matrix with real entries and $N(s)$ is a $n \times m$ matrix with real entries for each $s \in (t, T)$ and is uniformly bounded, i.e, for all $s \in (t, T)$ we have $||N(s)|| \leq a$ for some $a \in \mathbb{R}$. In other words, we consider the following ordinary differential equation

$$\begin{cases} \dfrac{dy}{ds}(s) = My(s) + N(s)u(s) & \text{in } (t, T), \\ y(t) = x. \end{cases} \tag{1.1}$$

We also specialize the running cost so that it only depends on the control. We also

suppose that the terminal cost J is convex. Specifically, the value function reads

$$V(x,t) = \inf_{u(\cdot)} \int_t^T L(u(s))ds + J(y(T)),$$

where $L : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ is a proper, lower semicontinuous, convex and its domain of definition reads dom $L = C$, where we recall that $C \subset \mathbb{R}^m$ is compact.

Now we consider the following change of variable $w(s) = e^{-sM}y(s)$. Using this change of variable, the ordinary differential equation (1.1) reads

$$\begin{cases} \dfrac{dw}{ds}(s) = e^{-sM}N(s)u(s) & \text{in } (t,T), \\ w(t) = e^{tM}x = z. \end{cases} \tag{1.2}$$

After this change of variable, the resulting value function for the optimal control problem becomes

$$\varphi(z,t) = \inf_{u(\cdot)} \int_t^T L(u(s))ds + J(e^{TM}z)$$

under the constraint that the trajectory the ordinary differential equation (1.2) (and where z denotes the initial state in (1.2)). Now define $\widetilde{J}(x) = J(e^{TM}x)$. Then we have that φ satisfies the following HJ PDE

$$\begin{cases} \dfrac{\partial \varphi}{\partial t}(z,\bar{t}) + H(\nabla_z \varphi(z,\bar{t}),\bar{t}) = 0 & \text{in } \mathbb{R}^n \times (0,+\infty), \\ \varphi(z,0) = \widetilde{J}(z) & \forall z \in \mathbb{R}^n, \end{cases} \tag{1.3}$$

where the Hamiltonian $H : \mathbb{R}^n \times [0,+\infty) \to \mathbb{R} \cup \{+\infty\}$ is defined by

$$H(p,s) = -\inf_{c\in\mathbb{R}^n}\left(\left\langle e^{-sM}N(s)c, p\right\rangle + L(c)\right) \tag{1.4}$$

$$= -\min_{c\in\mathbb{R}^n}\left(\left\langle e^{-sM}N(s)c, p\right\rangle + L(c)\right). \tag{1.5}$$

where we have use the fact dom L is compact.

Recall that we assume that the terminal cost J is convex. Therefore we also have that $\widetilde{J}$ is convex. Then, using [63, Section 5.3.2, p.215] we have that the viscosity solution to (1.3) can be represented as follows

$$\varphi(x,t) = -\min_{p\in\mathbb{R}^n}\left(\widetilde{J}^*(p) + \int_0^t H(p,s)\,ds - \langle x, p\rangle\right), \tag{1.6}$$

where we recall that $\widetilde{J}^*$ denotes the Fenchel-Legendre transform of $\widetilde{J}$. The above formula is known as the generalized Hopf formula. In addition, if (1.6) has a unique minimizer for a given (x,t), then, using variational analysis techniques [13], it can be shown shown that φ is differential with respect to its first variable at (x,t) and its gradient (i.e, $\nabla\varphi(x,t)$) is the minimizer of (1.6).

Following [58, 60], we now also assume that the map L that appears in the running cost is the characteristic function $I_{\mathcal{E}}$ of the ellipsoid $\mathcal{E}$ which is defined as follows

$$\mathcal{E} = \{x | x^\top E x \leq 1\}$$

where the matrix E is symmetric positive definite. This is exactly the set up considered in [58, 60].

We now wish to obtain a formula for the Hamiltonian $H(p,s)$ given in (1.5) when L is the characteristic function of an ellipsoid. Let s be given and denote by

A the matrix $e^{-sM}N(s)$ to shorten notations, i.e., $A = e^{-sM}N(s)$. Let $g(c) = \frac{1}{2}\langle c, Ec\rangle - \frac{1}{2}$. Using this notations we need to solve for c the following optimization problem $\min_c \langle A^\top p, c\rangle$ under the constraint $g(c) \leq 0$ (which is non empty since E is symmetric and positive definite). If $A^\top p = 0$ then $\min_{c\in C} \langle Ac, p\rangle = 0$. Therefore if $A^\top p = 0$ then $H(p, s) = 0$. Now suppose that $A^\top p \neq 0$. Let $\bar{c}$ be a global minimizer (which is guaranteed since we are minimizing a continuous function on compact set). Using Karush-Kuhn-Tucker (KKT) theorem, we obtain that there exist $\bar{\lambda}$ such that

(a) $A^\top p + \bar{\lambda} E\bar{c} = 0$ with $\bar{\lambda} \geq 0$,

(b) $g(\bar{c}) = \frac{1}{2}\langle \bar{c}, E\bar{c}\rangle - \frac{1}{2} = 0$.

Thereore we deduce that $1 = \langle \bar{c}, E\bar{c}\rangle = \left\langle -\frac{1}{\bar{\lambda}} E^{-1}A^\top p, E\left(-\frac{1}{\bar{\lambda}} E^{-1}A^\top p\right)\right\rangle$ which implies that the $\bar{\lambda} = \sqrt{p^\top AE^{-1}A^\top p}$ and $\bar{c} = \frac{1}{\sqrt{p^\top AE^{-1}A^\top p}} E^{-1}A^\top p$. Therefore, we have

$$\min_{c\in C} \langle Ac, p\rangle = -\sqrt{p^\top AE^{-1}A^\top p}.$$

Now let the matrix $Q(s) = e^{-sM}N(s)E^{-1}N(s)^\top e^{-sM^\top}$. Then the Hamiltonian reads as follows when $A^topp \neq 0$

$$\begin{aligned} H(p, s) &= -\min_{c\in C} \left\langle e^{-sM}N(s)c, p\right\rangle \\ &= \sqrt{p^\top e^{-sM}N(s)E^{-1}N(s)^\top e^{-sM^\top} p} \\ &= \left\|p\right\|_{Q(s)} . \end{aligned}$$

Therefore the Hamiltonian $H(p, s)$ has an explicit formula which reads as follows

$$H(p, s) = \begin{cases} 0 & \text{if } (e^{-sM}N(s))^\top p = 0, \\ \|p\|_{Q(s)} & \text{otherwise.} \end{cases}$$

Our goal is now to leverage these formulas to create numerical algorithms for solving these HJ PDEs and associated optimal controls.

1.4 Discretization of the generalized Hopf formula and projection on Minkowski sum of ellipsoids

So far we have obtained a variational representation formula (known as the generalized Hopf formula) for the viscosity solution to certain HJ PDEs that arise from the optimal control problems considered in this thesis. In general, the generalized Hopf representation formula does not have an explicit form. Therefore, in general, we need to consider numerical approximations. Therefore we consider discretizations of the generalized Hopf formula and resulting finite dimensional convex optimization problems.

So, the idea is to approximate the integral involved in the generalized Hopf formula using quadrature rules. It formally consists of the performing the following approximation of $\varphi(x,t)$ as follows

$$\begin{aligned}
\varphi(x,t) &= -\min_{p\in\mathbb{R}^n}\left(\widetilde{J}^*(p) + \int_0^t H(p,s)ds - \langle x,p\rangle\right)\\
&\approx -\min_{p\in\mathbb{R}^n}\left(\widetilde{J}^*(p) + \sum_{k=1}^{N} H\left(p, \sum_{j=1}^{k} t_j\right)c_k t_k - \langle x,p\rangle\right)\\
&= -\min_{p\in\mathbb{R}^n}\left(\widetilde{J}^*(p) + \sum_{k=1}^{N} \left\|p\right\|_{Q\left(\sum_{j=1}^{k} t_j\right)} c_k t_k - \langle x,p\rangle\right).
\end{aligned}$$

where $c_k \in \mathbb{R}$ and $t_k > 0$, such that $\sum_{k=1}^{N} t_k = t$, come from quadrature rule approximations. Let us introduce the matrix $Q_k^{-1} = (c_k t_k)^2\, Q\left(\sum_{j=1}^{k} t_j\right)$. We have that

$\left\|p\right\|_{Q(\frac{tk}{N})\,\frac{t}{N}} = \sqrt{\langle x, Q_k^{-1}x\rangle}$. So our approximation of $\varphi(x,t)$ reads

$$\varphi(x,t) \approx -\min_{p\in\mathbb{R}^n}\left\{-\langle x,p\rangle + \widetilde{f}^*(p) + \sum_{k=1}^{N}\left\|p\right\|_{Q_k^{-1}}\right\}.$$

1.5 Convex optimization algorithms and the projection on the Minkowski sum of ellipsoids

We now briefly describe our to compute solution the following optimization problem

$$-\min_{p\in\mathbb{R}^n}\left\{-\langle x,p\rangle + \widetilde{f}^*(p) + \sum_{k=1}^{N}\left\|p\right\|_{Q_k^{-1}}\right\} \tag{1.7}$$

which corresponds to a discretization of the generqalized Hopf formula. We consider the standard ADMM/split-Bregman approach [25] which consists of iterating the following

$$v_{k+1} = \arg\min_{v\in\mathbb{R}^n}\left\{\widetilde{f}^*(v) - \langle x,v\rangle + \frac{\lambda}{2}\left\|d_k - v - b_k\right\|_2^2\right\}, \tag{1.8}$$

$$d_{k+1} = \arg\min_{d\in\mathbb{R}^n}\left\{\sum_{j=1}^{N}\|d\|_{Q_j^{-1}} + \frac{\lambda}{2}\left\|d - v_{k+1} - b_k\right\|_2^2\right\}, \tag{1.9}$$

$$b_{k+1} = b_k + v_{k+1} - d_{k+1}. \tag{1.10}$$

For simplicity, we set the parameter$\lambda = 1$ and consider the following initial condition $v_0 = x$, $d_0 = x$ and $b_0 = 0$. The sequences $\{v_k\}$ and $\{d_k\}$ both converge to the minimizer of (1.7). We need to compute (1.8) and (1.9) to implement this algorithm. For optimal control applications, standard initial data J reads as follows

- $J = \frac{1}{2}\left\|\cdot\right\|_p^2$ for $p = 1, 2, \infty$,

- $J = \frac{1}{2}\langle \cdot, A\cdot \rangle$ with a positive definite diagonal matrix A,

for which efficient algorithms to compute (1.8) are available (see [25]).

The difficult part consists of computing (1.9). We shall see below that it amounts to be able to compute efficiently the projection onto the Minkowsky sum of ellipsoids.

1.5.1 Projection on Minkowski sum of ellipsoids

Note that (1.9) can be written as

$$\underset{d\in\mathbb{R}^n}{\arg\min}\left\{\alpha\sum_{j=1}^{N}\|d\|_{Q_j^{-1}}+\frac{1}{2}\|d-w\|_2^2\right\} \tag{1.11}$$

for a given vector $w \in \mathbb{R}^n$ and $\alpha > 0$. Using classical Fenchel-Legendre calculus we have

$$\begin{aligned}\left\{\sum_{j=1}^{N}\|d\|_{Q_j^{-1}}\right\}^* &= \|d\|^*_{Q_1^{-1}} \square \cdots \square \|d\|^*_{Q_N^{-1}} \\ &= \mathcal{I}_{\mathcal{E}}(0,Q_1)\square\cdots\square\mathcal{I}_{\mathcal{E}}(0,Q_N) \\ &= \mathcal{I}_{\sum_{j=1}^{N}\mathcal{E}(0,Q_j)}\end{aligned}$$

where we recall that $\square$ denotes the infimal-convolution operator (see Tab. 1.1). Let C be the set of Minkowski sum of ellipsoids $\sum_{j=1}^{N}\mathcal{E}(0,Q_j)$. Using this notation the optimization problem (1.11) reads

$$\underset{d\in\mathbb{R}^n}{\arg\min}\left\{\alpha\mathcal{I}_C^*+\frac{1}{2}\|d-w\|_2^2\right\} \tag{1.12}$$

Then we can invoke Moreau's identity to obtain

$$\arg\min_{d\in\mathbb{R}^n}\left\{\alpha\mathcal{I}_C^*(d)+\frac{1}{2}\|d-w\|_2^2\right\}+\alpha\arg\min_{d\in\mathbb{R}^n}\left\{\frac{1}{\alpha}\mathcal{I}_C(d)+\frac{1}{2}\left\|d-\frac{w}{\alpha}\right\|_2^2\right\}=w$$

Therefore to compute (1.12), it is enough to compute $\arg\min_{d\in\mathbb{R}^n}\left\{\frac{1}{\alpha}\mathcal{I}_C(d)+\frac{1}{2}\left\|d-\frac{w}{\alpha}\right\|_2^2\right\}$.

$$\begin{aligned}\arg\min_{d\in\mathbb{R}^n}\left\{\frac{1}{\alpha}\mathcal{I}_C(d)+\frac{1}{2}\left\|d-\frac{w}{\alpha}\right\|_2^2\right\}&=\arg\min_{d\in C}\left\{\frac{1}{2}\left\|d-\frac{w}{\alpha}\right\|_2^2\right\}\\&=\Pi_C\left(\frac{w}{\alpha}\right)\end{aligned}$$

where we recall that $\Pi_C(\frac{w}{\alpha})$ denotes the projection of $\frac{w}{\alpha}$ onto C. Recall that C is the Minkowsky sum of ellipsoids. Therefore the performance of ADMM essentially relies on the efficiency of the algorithm to compute the projection onto the Minkowski sum of ellipsoids. We will propose in chapter 2 an efficient algorithm to compute this projection.

1.6 Connections to multi-time Hamilton-Jacobi PDEs

In this section, we draw connections between a class of multi time Hamilton-Jacobi Partial Differential Equation and discretization of the integral that appears in the general Hopf formula. The discretization can be written as follows

$$\phi(x,t)=-\min_{p\in\mathbb{R}^n}J^*(p)+\sum_{i}^{N}t_iH_i(p)-\langle x,p\rangle$$

Using [23] we can deduce that ψ satisfies the following multi-time Hamilton-Jacobi PDE

$$\begin{cases} \dfrac{\partial \psi}{\partial t_j}(x, t_1, \cdots, t_N) + H_j(\nabla_x \psi) = 0 & x \in \mathbb{R}^n, t_1, \cdots, t_N > 0, \\ \psi(x, 0, \cdots, 0) = J(x) & x \in \mathbb{R}^n. \end{cases}$$

where $\frac{\partial \psi}{\partial t_j}$ denote the partial derivative with respect to t_j and $\nabla_x \psi$ denote the gradient vector with respect to x. Under proper assumptions (see [23]), there exists a unique solution given by Hopf formula and Lax formula. These are Hopf formula and Lax formula respectively [82] [69],

$$\psi(x, t_1, \cdots, t_N) = \left(J^* + \sum_{j=1}^{N} t_j H_j \right)^* (x) \tag{1.13}$$

$$= \left(J \square \left(\sum_{j=1}^{N} t_j H_j \right)^* \right)^* (x). \tag{1.14}$$

This representation formula for the multi-time HJ PDEs can be seen as a discretization of the generalized Hopf formula presented in the previous subsection.

Now, we present several numerical examples that depicts the solution to these multi-time HJ PDEs using the algorithms developed chapter two. For simplicity, we only present two dimensional problems. The representation formula reads

$$\psi(x, t_1, t_2) = (J^* + t_1 H_1 + t_2 H_2)^* (x) \tag{1.15}$$

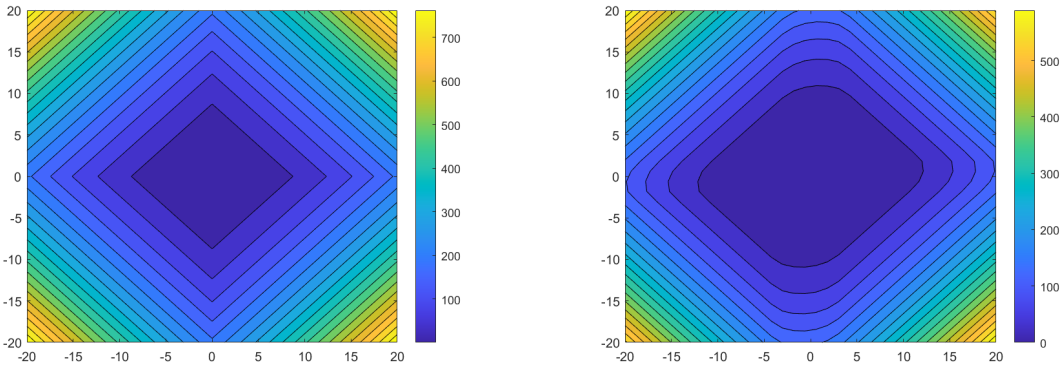
$$= (J \square (t_1 H_1)^* \square (t_2 H_2)^*)(x). \tag{1.16}$$

Define each Hamiltonian $H_j(x) = \sqrt{\langle x, E_j^{-1} x \rangle}$ for $j = 1, 2$ where the matrices E_1, E_2 are defined as

$$E_1 = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}, \; E_2 = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}.$$

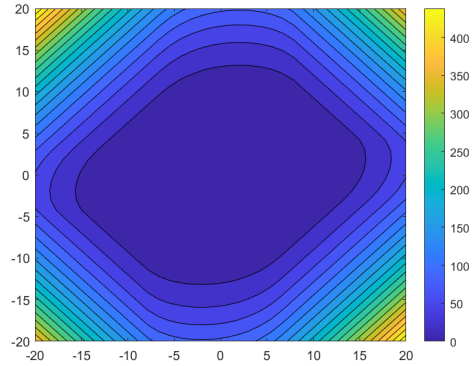
Eigenvectors of E_1 and E_2 are the basis vectors of each ellipsoid and the square root of corresponding eigenvalues are the semi axis of each ellipsoid. Eigenvalues and the corresponding eigenvectors of E_1 are $\left\{4, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right\}$ and $\left\{1, \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right\}$. Eigenvalues and the corresponding eigenvectors of E_2 are $\left\{3, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{bmatrix}\right\}$ and $\left\{1, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}\right\}$.

Figure 1.1 depicts the solutions for several times as well as certain level lines of the solution for the initial data $J = \frac{1}{2}\|\cdot\|_\infty^2$. Figure 1.2 depicts the solutions for several times as well as certain level lines of the solution for the initial data $J = \frac{1}{2}\|\cdot\|_1^2$. Figure 1.3 depicts the solutions for several times as well as certain level lines of the solution for the initial data $J = \frac{1}{2}\langle D\cdot, \cdot\rangle$ where $D = \begin{bmatrix} 1 & 0 \\ 0 & 0.25 \end{bmatrix}$.

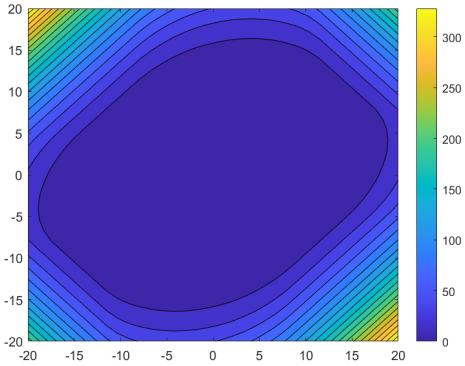


(a) $t_1 = 0, t_2 = 0$

(b) $t_1 = 2.5, t_2 = 2.5$

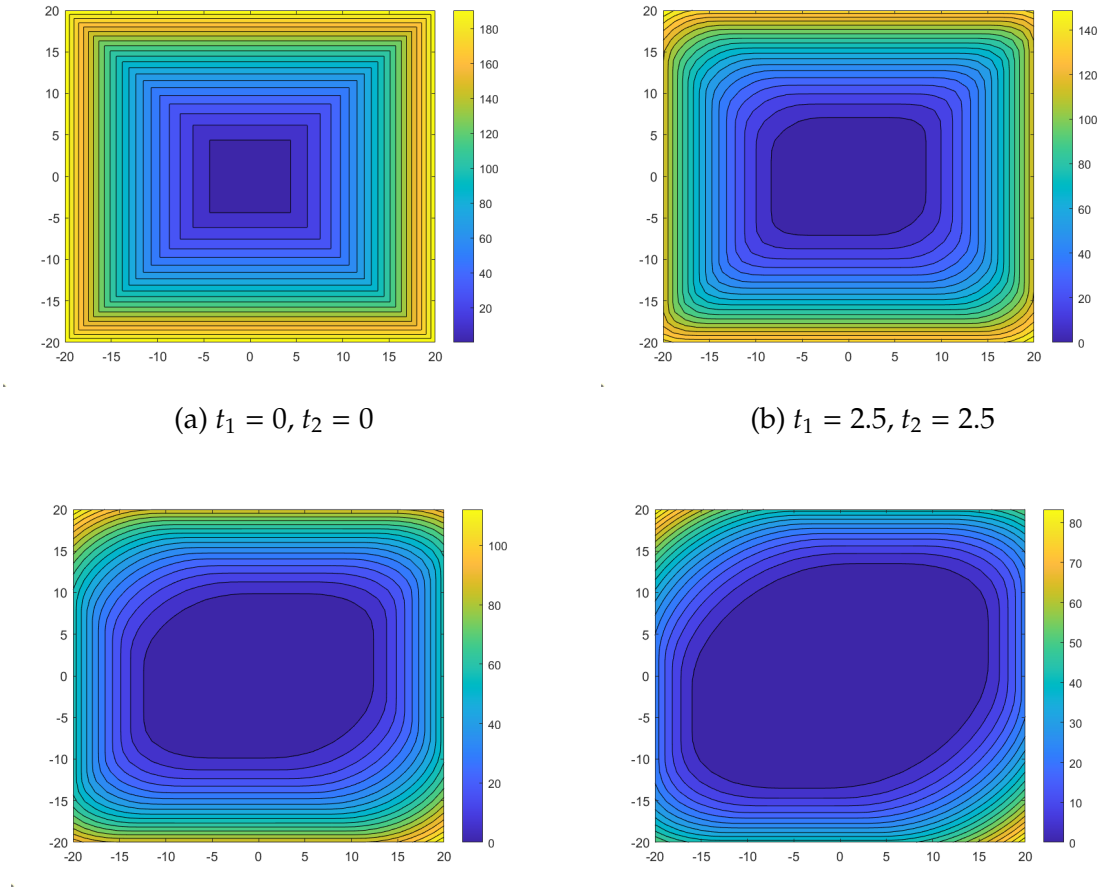


(c) $t_1 = 5, t_2 = 5$



(d) $t_1 = 5, t_2 = 10$

Figure 1.1: Evaluation of the solution $\psi(x, t_1, t_2)$ of the Hamilton Jacobi PDE with initial data $J = \frac{1}{2}\|\cdot\|_\infty^2$ and Hamiltonians $H_1(x) = \sqrt{\langle x, E_1^{-1}x\rangle}$, $H_2(x) = \sqrt{\langle x, E_2^{-1}x\rangle}$



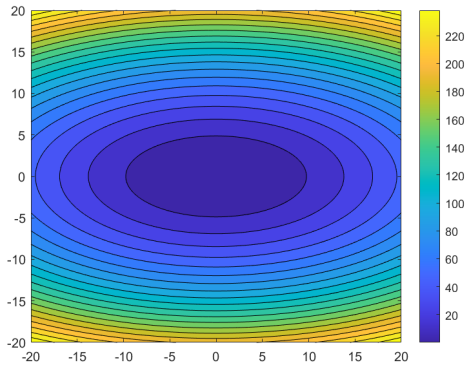
(a) $t_1 = 0, t_2 = 0$

(b) $t_1 = 2.5, t_2 = 2.5$

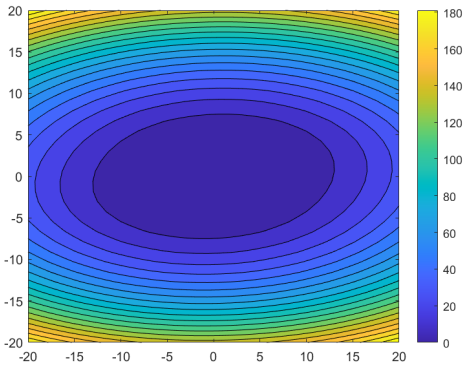
(c) $t_1 = 5, t_2 = 5$

(d) $t_1 = 5, t_2 = 10$

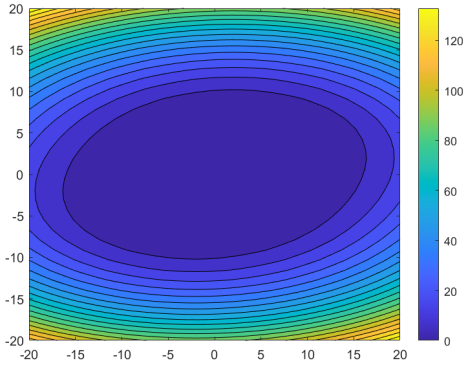
Figure 1.2: Evaluation of the solution $\psi(x, t_1, t_2)$ of the Hamilton Jacobi PDE with initial data $J = \frac{1}{2}\|\cdot\|_1^2$ and Hamiltonians $H_1(x) = \sqrt{\langle x, E_1^{-1}x\rangle}$, $H_2(x) = \sqrt{\langle x, E_2^{-1}x\rangle}$



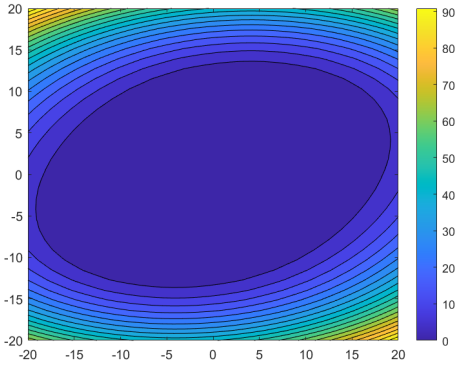
(a) $t_1 = 0, t_2 = 0$



(b) $t_1 = 2.5, t_2 = 2.5$



(c) $t_1 = 5, t_2 = 5$



(d) $t_1 = 5, t_2 = 10$

Figure 1.3: Evaluation of the solution $\psi(x, t_1, t_2)$ of the Hamilton Jacobi PDE with initial data $J = \frac{1}{2}\langle D\cdot, \cdot\rangle$ where $D = \begin{bmatrix} 1 & 0 \\ 0 & 0.25 \end{bmatrix}$ and Hamiltonians $H_1(x) = \sqrt{\langle x, E_1^{-1}x\rangle}$, $H_2(x) = \sqrt{\langle x, E_2^{-1}x\rangle}$

1.7 A brief overview of the chapters of this thesis

We now briefly present some important ideas used in this thesis. In chapter 2, we describe a novel Newton based algorithm for the projection onto Minkowski sum of ellipsoids. As mentioned in this chapter, this algorithm is crucial to efficiently compute generalized Hopf formulas where the running cost function in the optimal problem is a characteristic function of a convex ellipsoid and the dynamics are linear. In particular, this novel algorithm can be used for solving an important class of optimal control problems that is used by the US Navy [58, 60, 59].

In chapter 3, we present an extension of the algorithm described in chapter 2 to cover more general optimal control problems. Indeed, while we consider non-degenerate ellipsoids in Chapter 2, we allow the control to live in a degenerate ellipsoids in this chapter. This extension is important because most of linear optimal control problems in a real application yield the projection onto Minkowski sum of degenerate ellipsoid.

In Chapter 4, we propose a novel algorithm for computing the proximal point of a convex polyhedral function whose domain is a polyhedral set. This algorithm serves as a building block for solving certain multi-time Hamilton-Jacobi PDEs that arise from certain optimal control problems.

Chapter Two

Projection on Minkowski sum of ellipsoids

2.1 Introduction

As mentioned in the first chapter, variational representation formulas that solves Hamilton-Jacobi PDEs are very appealing because it suggests that they can be computed using optimization techniques. This chapter focuses on computing the projection onto a Minkowski sum full ellipsoids. Several algorithms have been proposed to compute the projection on the closed convex sets such as Gilbert Method [40],[8] , Primal-Dual method [58],[60] and Nesterov-based algorithms [80], [76], [75]. However, none of them are fast enough in practice to solve the class of optimal control problems considered in the first chapter. In addition none of them provide good enough theoretical guarantees. As mentioned in the first chapter, we focus on computing the projection onto a Minkowski sum of full ellipsoids. We present a Newton's method to compute the projection. It is well known that Newton's method has a quadratic convergence and is one of the best algorithms in practice. However, the initial guess has to be close to the solution to achieve the convergence of Newton's method. We propose to use Gilbert Method to get a good initial guess for Newton's Method. We also present a straightforward parallel version of the proposed algorithm for computing the projection.

The main contribution of this chapter is the development of a fast algorithm for real-time computing projections on a Minkowski sum of ellipsoids. We show that it outperforms the standard Gilbert method, Primal-Dual method and Nesterov algorithm. Our algorithm paves the way for real-time applications arising from certain optimal control problem. Since we focus on the projection of the Minkowski sum of full ellipsoids, we have an access to the second order information which is used in Newton's method. Note that other proposed methods, such Primal-Dual method and Nesterov-based algorithm, only use first order information.

The remainder of this chapter is as follows. In Section 2, we define the problem and introduce several definitions and results of convex analysis. In Section 3, we describe Gilbert method, Modified Gilbert method and Nesterov method. In particular we show how we can leverage the Ellipsoidal Toolbox[65] to implement efficiently Gilbert methods when the projection on the Minkowsky sum of ellipsoid is considered. Section 4 presents our algorithm. In Section 5, we present some numerical results.

2.2 State of the art algorithms

In this chapter, we present Gilbert Algorithm, Modified Gilbert Algorithm and Nesterov Algorithm to compute the projection on the Minkowski sum of non-degenerate ellipsoids. Let us first clarify what the Minkowski sum of sets is and define our problem. The Minkowski sum $A+B$ of sets A and B is defined as follows

$$A + B = \{a + b : a \in A, b \in B\}.$$

There is no confusion for a finite Minkowski sum because $A + B = B + A$ and $A+(B+C) = (A+B)+C$ for sets, A, B and C. Now consider an ellipsoid, $\mathcal{E}(q, Q) \subseteq \mathbb{R}^n$ where $q \in \mathbb{R}^n$ and $Q \in \mathbb{R}^n \times \mathbb{R}^n$ is symmetric positive definite, which is defined by

$$\mathcal{E}(q, Q) \coloneqq \left\{x \in \mathbb{R}^n : \left\langle (x - q), Q^{-1}(x - q) \right\rangle \leq 1\right\}.$$

Consider the Minkowski sum of non-degenerate ellipsoids $\mathcal{E}(q_1, Q_1), \cdots, \mathcal{E}(q_k, Q_k) \subseteq \mathbb{R}^n$. We want to calculate projection on the Minkowski sum of non-degenerate el-

lipsoids from the origin which corresponds to

$$\arg\min_{x\in\mathbb{R}^n}\left\{\|x\|_2 : x \in \sum_{i=1}^{k}\mathcal{E}(q_i, Q_i)\right\}.$$

Here we formulate the specific projection of 0 onto the closed convex set, sum of $\mathcal{E}(q_i, Q_i)$. There is no loss of generality since a the projection of a point z simply corresponds to change of variable

$$\arg\min_{x\in\mathbb{R}^n}\left\{\|x-z\|_2 : x \in \sum_{i=1}^{k}\mathcal{E}(q_i, Q_i)\right\} = \arg\min_{x\in\mathbb{R}^n}\left\{\|x\|_2 : x \in \sum_{i=1}^{k}\mathcal{E}(q_i^*, Q_i)\right\}$$

where $q_1^* = q_1 - z$ and $q_i^* = q_i$ for $i \geq 2$. We now introduce several definitions and results of convex analysis that will be used in this chapter.

Definition 2.2.1. Projection on the set A from the point s is defined by

$$\Pi_A(s) := \arg\min_{x\in\mathbb{R}^n}\{\|x-s\|_2 : x \in A\}$$

Let A be a nonempty set in $\mathbb{R}^n$ and K be a compact and convex set in $\mathbb{R}^n$. The following definitions are given in [40] and necessary to define Gilber algorithms.

Definition 2.2.2. A function $\sigma_A(y) : \mathbb{R}^n \to \mathbb{R}^n \cup \{+\infty\}$ is a support function of A defined by

$$\sigma_A(y) := \sup\{\langle s, y\rangle : s \in A\}$$

Since K is a compact, $\sigma_K(y) = \max\{\langle s, y\rangle : s \in K\}$ and $\sigma_K(y) \in \mathbb{R}$ for any $y \in \mathbb{R}^n$. $\sigma_K(y)$ is a convex function on $\mathbb{R}^n$ and this implies that $\sigma_K(y)$ is continuous on $\mathbb{R}^n$. Let $P(y), y \neq 0$ be the hyperplane $\{x|\langle x, y\rangle = \sigma_K(y)\}$. $P(y)$ is the unique support hyperplane of K because $\sigma_K(y) = \langle s^*, y\rangle$ for some $s^* \in K$ and $\langle s, y\rangle \leq \sigma_K(y)$ for any $s \in K$. Figure 2.1 depicts an illustration of $P(y)$.

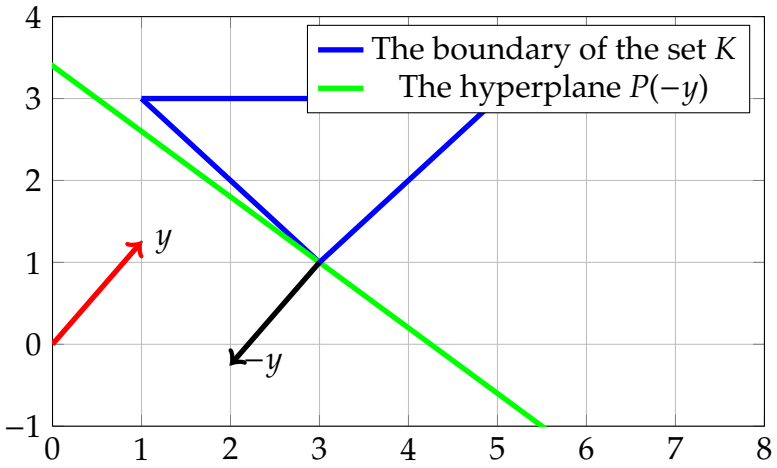


Figure 2.1: Example of a set K and its hyperplane $P(-y)$ for a given vector $y \in K$

Definition 2.2.3. A set function $S(y)$, defined on $\mathbb{R}^n$ is a contact set of K defined by

$$S(y) := P(y) \cap K.$$

Note that $S(y)$ is not empty, $S(y) \in \partial K$, $S(wy) = S(y)$ for $w > 0$.

Definition 2.2.4. A function $s(y)$, defined on $\mathbb{R}^n$ is a contact function of K if $s(y) \in S(y)$, $y \neq 0$, and $s(0) \in K$.

We now introduce convergence rates to clarify to compare algorithms.

Definition 2.2.5. A sequence $\{z_k\}_{k\in\mathbb{N}}$ converging to z^* converges linearly, if there exists $r \in (0, 1)$ such that

$$\limsup \frac{\|z_{k+1} - z^*\|_2}{\|z_k - z^*\|_2} = r$$

Definition 2.2.6. A sequence $\{z_k\}_{k\in\mathbb{N}}$ converging to z^* converges quadratically, if there exists $r > 0$ such that

$$\limsup \frac{\|z_{k+1} - z^*\|_2}{\|z_k - z^*\|_2^2} = r$$

2.2.1 Gilbert Algorithm

We now present Gilbert algorithm [40]. Let K be a compact and convex set and z^* be a point that satisfies $z^* = \arg\min_{z \in K} \|z\|_2$. z^* is a projection of the origin onto K and z^* is unique. Let $s(\cdot)$ be a contact function of K which is compact and convex. We assume that we can compute the contact function, $s(\cdot)$. Consider

$$\beta(z) = \begin{cases} \|z - s(-z)\|_2^{-2} \langle z, z - s(-z) \rangle & \text{if } z - s(-z) \neq 0, \\ 0 & \text{if } z - s(-z) = 0. \end{cases}$$

$$\gamma(z) = \begin{cases} \|z\|_2^{-2} \langle z, s(-z) \rangle & \text{if } \|z\|_2 > 0, \langle z, s(-z) \rangle > 0, \\ 0 & \text{if } z = 0 \text{ or } \|z\|_2 > 0, \langle z, s(-z) \rangle \leq 0, \end{cases}$$

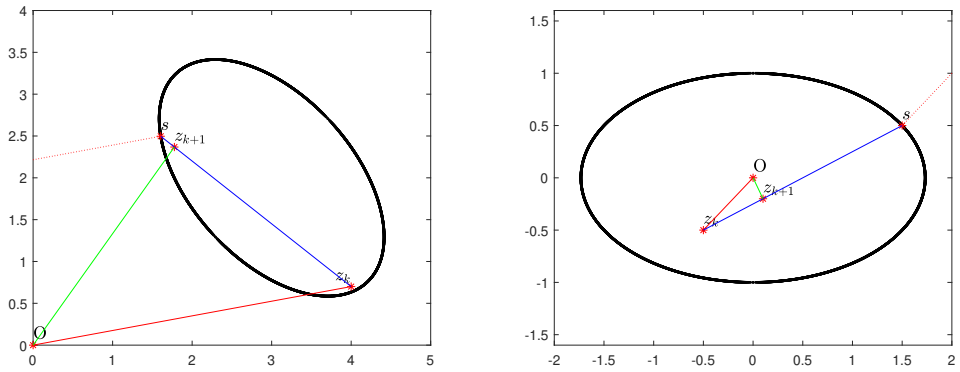
Gilbert algorithm corresponds to the following iterative procedure

$$z_{k+1} = z_k + \alpha_k (s(-z_k) - z_k), \quad k = 0, 1, 2, \cdots$$

where z_0 is an arbitrary point in K and α_k is chosen arbitrarily from the interval $I_\delta(z_k)$ defined as follows

$$I_\delta(z) = [\min\{\delta\beta(z), 1\}, \min\{(2 - \delta)\beta(z), 1\}], \quad 0 < \delta \leq 1.$$

The sequence $\{z_k\}_{k \in \mathbb{N}}$ converges to z^*. Since we do not have access to the solution, z^*, we can use the following stopping criterion $\|z_k\|_2 - \|z_k\|_2 \gamma(z_k)$. This is



(a) When K doesn't contain the origin (b) When K contains the origin

Figure 2.2: Demonstration of how Gilbert method works. z_{k+1} is the point with the smallest distance between the origin and blue line when $\delta = 1$.

a reasonable choice because

$$\|z_k\|_2 - \|z^*\|_2 \leq \|z_k\|_2 - \|z_k\|_2 \, \gamma(z_k)$$

$$\|z_k\|_2 \, \gamma(z_k) \longrightarrow \|z^*\|_2 \text{ as } k \longrightarrow \infty.$$

For $\|z^*\|_2 > 0$ and $k \geq 1$, the rate of convergence is as follows

$$\|z_k\|_2 - \|z^*\|_2 < 2\mu^2 \, \|z^*\|_2^{-1} \, \delta^{-1} k^{-1}$$

$$\|z_k - z^*\|_2 < 2\mu\delta^{-1/2} k^{-1/2}$$

where $\mu = \max_{a,b \in K} \|a - b\|_2$ which is the diameter of K. This is not a desirable result because much faster algorithms are required for computing solutions to optimal control problems in real time. However, in practice, it can perform much better than the above rate of convergence.

Denote $z = (z^1, z^2, \cdots, z^n)$ and let $\delta = 1$,

$$K = \left\{ z | z^1 \geq \nu + \frac{1}{2} \sum_{i=2}^{n} (z^i)^2 \lambda_i^{-1}, z^i \leq 2\nu \right\}$$

where $\nu, \lambda_1, \cdots, \lambda_n > 0$. We get $z^* = (\nu, 0, \cdots, 0)$ and in the neighborhood of z^*, ∂K is the elliptic hyperparaboloid where z^1 reads

$$z^1 = \nu + \frac{1}{2} \sum_{i=2}^{n} (z^i)^2 \lambda_i^{-1}$$

The rate of convergence is

$$\|z_k\|_2 - \|z^*\|_2 < \frac{1}{2} \nu^{-1} \theta_0 \left(1 - \frac{1}{2} \underline{\beta} \delta\right)^k$$
$$\|z_k - z^*\|_2 < \sqrt{\theta_0} \left(1 - \frac{1}{2} \underline{\beta} \delta\right)^{k/2},$$

where the constants θ_0 and $\underline{\beta}$ are

$$\theta_0 = \|z_0\|_2^2 - \|z_*\|_2^2$$
$$\underline{\beta} = \frac{1}{1 + \frac{5}{2} \bar{\lambda} \nu^{-1} + \bar{\lambda}^{-2} \nu^{-2}}$$

where $\bar{\lambda} = \max_{i=1,2,\cdots,n} \lambda_i$. Since $\theta_0 > 0$ and $\underline{\beta} > 0$, the convergence rate is linear. However, if $\underline{\beta} \ll 1$, the convergence rate will be much slower. $\underline{\beta} \ll 1$ means that ν is significantly smaller than $\bar{\lambda}$, the biggest λ_i.

For many convex sets, in the neighborhood of z^*, ∂K can be closely approximated by an elliptic hyperparaboloid. Gilbert shows that if z^* can be closely approximated by an elliptic hyperparaboloid, the convergence rate is linear. However, if z^* has at least one principal radius of curvature large compared with $\|z^*\|_2$, it will converge slowly. Figure 2.3 depicts an example of the cases of slow convergence. It is an ellipsoid with a bad condition number and 0 is close to the ellipsoid.

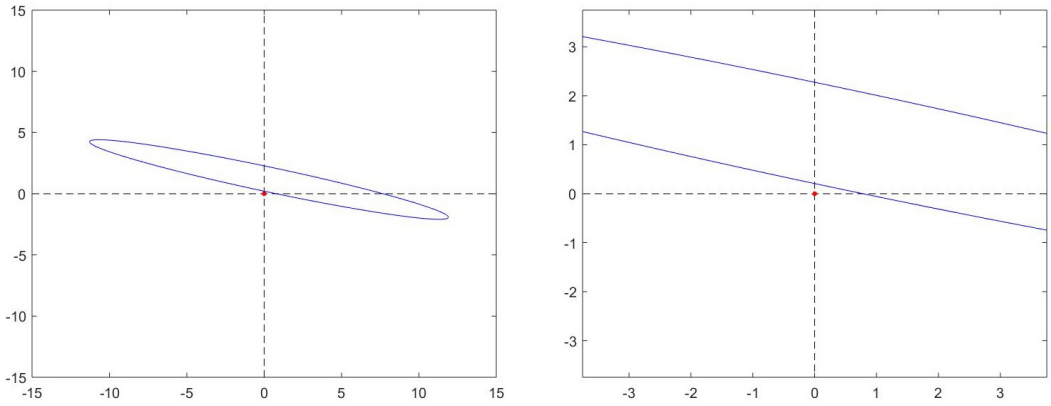


Figure 2.3: A two-dimensional ellipsoid which corresponds to a case where Gilbert algorithm converges slowly. The left figure and right figure depicts the same ellipsoid at different scale.

We now show that we can use the Ellipsoidal Toolbox[65] to define the internal ellipsoid and compute the contact function that is need to implement Gilbert method. Consider the Minkowski sum of non-degenerate ellipsoids, $\mathcal{E}(q_1, Q_1), \cdots, \mathcal{E}(q_k, Q_k) \subseteq \mathbb{R}^n$. It is not an ellipsoid in general, but we can approximate it by the parameterized families of internal ellipsoids.

Let the parameter l be a nonzero vector in $\mathbb{R}^n$. Define $q = \sum_{i=1}^{k} q_i$. For a given vector $l \in \mathbb{R}^n \setminus \{0\}$, define Q_l as

$$Q_l = \left(Q_1^{\frac{1}{2}} + S_2 Q_2^{\frac{1}{2}} + \cdots S_k Q_k^{\frac{1}{2}}\right)^T \left(Q_1^{\frac{1}{2}} + S_2 Q_2^{\frac{1}{2}} + \cdots S_k Q_k^{\frac{1}{2}}\right)$$

where S_i, $i = 1, \cdots k$ are orthogonal matrices and make the vectors $Q_1^{\frac{1}{2}} l$, $S_2 Q_2^{\frac{1}{2}} l$, $\cdots$ $S_k Q_k^{\frac{1}{2}} l$ parallel. Note that S_i, $i = 1, \cdots k$ are not necessarily unique. Then the internal matrix $\mathcal{E}_{in}(l) = \mathcal{E}(q, Q_l)$ has following properties.

$$\mathcal{E}_{in}(l) \subseteq \mathcal{E}(q_1, Q_1) + \cdots + \mathcal{E}(q_k, Q_k)$$

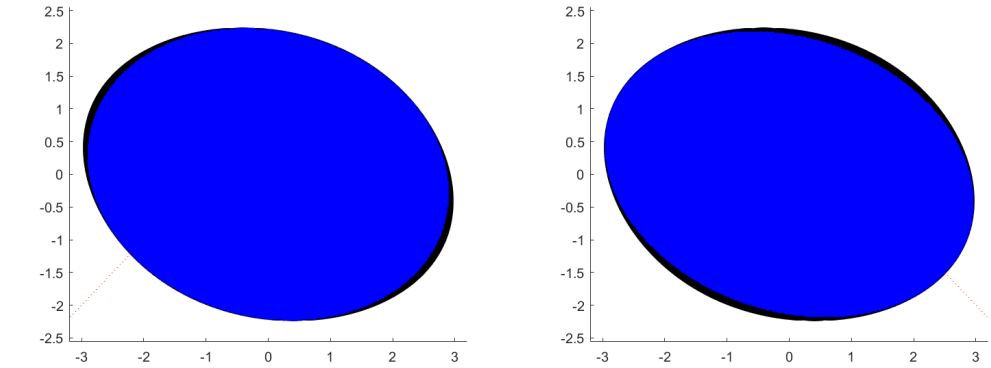
$$\sigma_{\mathcal{E}_{in}(l)}(l) = \sigma_{\mathcal{E}(q_1, Q_1)}(l) + \cdots + \sigma_{\mathcal{E}(q_k, Q_k)}(l).$$

Figure 2.4 illustrates the concept of internal ellipsoids.

Remark 2.2.7. We know that $\mathcal{E}_{in}(l)$ is not the best internal ellipsoid. This means that there may exist ellipsoids which are strictly bigger than the internal ellipsoid and have the same properties. However, in practice, $\mathcal{E}_{in}(l)$ is a good approximation [65].

Therefore, the contact function $s(\cdot)$ can be efficiently computed using the following formula

$$s(l) = \frac{Q_l l}{\langle l, Q_l l \rangle^{\frac{1}{2}}} + q.$$



(a) $\ell = [-1; -1]$ (b) $\ell = [1; -1]$

Figure 2.4: Black ball represents a Minkowski sum of two ellipsoids, $Q_1 = [3, 0; 0, 1]$ and $Q_2 = [1.5, -0.5; -0.5, 1.5]$. Blue ball represents the internal ellipsoid with each ℓ.

Gilbert algorithm works well in most cases because the boundary of Minkowski sum of ellipsoids can be closely approximated by an elliptic hyperparaboloid. If z^* has at least one principal radius of curvature large compared with $\|z^*\|$ which implies that at least one of ellipsoids has a bad condition number and the origin is close to K.

2.2.2 Modified Gilbert Algorithm

We now present modified Gilbert algorithm that improves the convergence rate for computing the projection on an ellipsoid with a bad condition number: it relies on using the projection of z onto the internal ellipsoid $\mathcal{E}_{in}(\cdot)$ defined in the previous section. The idea to improve Gilbert method is motivated by Barr [8] who proposed modified Gilbert algorithm that converges faster than Gilbert algorithm. Note that there is a extra computational cost which depends on the dimension of the problem. However, this algorithm does not depend on the dimension of the problem. This is one of the main contributions in this chapter.

Define $\bar{z}_{k+1} = z_k + \alpha_k(s(-z_k) - z_k), \quad k = 0, 1, 2, \cdots$ which corresponds to the general iteration of Gilbert Algorithm. The modified iterative procedure is as follows

$$z_{k+1} = \begin{cases} \Pi_{\mathcal{E}_{in}(-z_k)}(0) & \text{if } \|\bar{z}_{k+1}\|_2 \geq \left\|\Pi_{\mathcal{E}_{in}(-z_k)}(0)\right\|_2, \\ \bar{z}_{k+1} & \text{if } \|\bar{z}_{k+1}\|_2 < \left\|\Pi_{\mathcal{E}_{in}(-z_k)}(0)\right\|_2. \end{cases}$$

We wish to prove that the modified Gilbert Algorithm is a convergence scheme. The proof is motivated by the paper [40]. Let $\Gamma(z) = \|z\|_2^2 - \|z^*\|_2^2$. We have that

$$\Gamma(z_k) - \Gamma(\bar{z}_{k+1}) \geq \min\left\{\frac{1}{4}\mu^{-2}\delta\Gamma^2(z_k), \frac{1}{2}\Gamma(z_k)\right\}$$

where $\mu = \max_{x,y\in K}\left\|x - y\right\|_2$. Since $\Gamma(\bar{z}_{k+1}) \geq \Gamma(z_{k+1})$, we have that $\Gamma(z_k) - \Gamma(z_{k+1}) \geq \Gamma(z_k) - \Gamma(\bar{z}_{k+1}) \geq \min\{\frac{1}{4}\mu^{-2}\delta\Gamma^2(z, \frac{1}{2}\Gamma(z_k)\}$. Thus, $\Gamma(z_k)$ is decreasing and bounded from below by 0. Let $\lim_{k\to\infty}\Gamma(z_k) = a \geq 0$.

$$\begin{aligned} 0 = \lim_{k\to\infty}\Gamma(z_k) - \Gamma(z_{k+1}) &\geq \lim_{k\to\infty}\min\left\{\frac{1}{4}\mu^{-2}\delta\Gamma^2(z_k), \frac{1}{2}\Gamma(z_k)\right\} \\ &= \min\left\{\frac{1}{4}\mu^{-2}\delta a^2, \frac{1}{2}a\right\} \end{aligned} \tag{2.1}$$

The inequality (2.1) is true because $\frac{1}{4}\mu^{-2}\delta\Gamma^2(z_k)$ and $\frac{1}{2}\Gamma(z_k)$ converges to $\frac{1}{4}\mu^{-2}\delta a^2$ and $\frac{1}{2}a$ respectively. Thus, we have $a = 0$ and we can conclude that our iterative procedure is a convergence scheme.

The Modified Gilbert Algorithm described above converges at least linearly. We do not have a proof that the modified Gilbert Algorithm is better than Gilbert Algorithm in some cases. We expect that modified Gilbert Algorithm will converge much faster than Gilbert Algorithm in case of ellipsoids with a bad condition number. The intuition for this expected behavior relies on geometry. Indeed we are using more information than Gilber algorithm and the internal ellipsoid is a good approximation of Minkowski sum of ellipsoids if we use a normal vector which is close to the actual vector, $z - z^*$. We will present a numerical results which show that modified Gilbert Algorithm is useful.

Since we know that Newton's method converges quadratically, we expect that algorithms based on second order information will also converge quadratically.

2.2.3 Nesterov Algorithm

We can change the problem into a dual problem by using Moreau Identity. Let the set $C = \sum_{i=1}^{k} \mathcal{E}(0, Q_i)$ and the vector $q = -\sum_{i=1}^{k} q_i$. We apply a translation $T_q : p \to p + q$ on the set $\sum_{i=1}^{k} \mathcal{E}(q_i, Q_i)$ to obtain

$$\arg\min_{x\in\mathbb{R}^n} \left\{ \|x\|_2 : x \in \sum_{i=1}^{k} \mathcal{E}(q_i, Q_i) \right\} = \arg\min_{x\in\mathbb{R}^n} \left\{ \left\|x - q\right\|_2 : x \in \sum_{i=1}^{k} \mathcal{E}(0, Q_i) \right\}.$$

By definition, this equals to

$$\arg\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\left\|x-q\right\|_2^2+I_C(x)\right\}.$$

By Moreau's Identity, we get that the above equals to

$$q-\arg\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\left\|x-q\right\|_2^2+I_C^*(x)\right\}.$$

Since we have $I_C^*(x)=\sup_{y\in\mathbb{R}^n}\{\langle x,y\rangle-I_C(y)\}$, we have that $I_C^*(x)=\sup_{y\in C}\{\langle x,y\rangle\}$. Thus, the above equals to

$$q-\arg\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\left\|x-q\right\|_2^2+\sup_{c\in C}\langle x,c\rangle\right\}.$$

Since we have the equation $\sup_{c\in C}\langle x,c\rangle=\sup_{c_1\in\mathcal{E}(0,Q_1),\cdots,c_k\in\mathcal{E}(0,Q_k)}\langle x,\sum_{i=1}^k c_i\rangle$, we have that $\sup_{c\in C}=\sum_{i=1}^k\sup_{c\in\mathcal{E}(0,Q_i)}\langle x,c\rangle$. Therefore the above equals to

$$q-\arg\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\left\|x-q\right\|_2^2+\sum_{i=1}^k\sup_{c\in\mathcal{E}(0,Q_i)}\langle x,c\rangle\right\}$$

If the vector x is equal to 0, we have that $\sup_{c\in\mathcal{E}(0,Q_i)}\langle x,c\rangle=\sqrt{x^TQ_ix}$. If the vector x is not 0, we can achieve the supremum when the normal vector of $\mathcal{E}(0,Q_i)$ at the boundary point c is the vector αx where $\alpha>0$. Such a point c is $\frac{Q_ix}{\sqrt{x^TQ_ix}}$ and we get $\sup_{c\in\mathcal{E}(0,Q_i)}\langle x,c\rangle=\left\langle x,\frac{Q_ix}{\sqrt{x^TQ_ix}}\right\rangle$ which equals to $\sqrt{x^TQ_ix}$. Thus, the above equals to

$$q-\arg\min_{x\in\mathbb{R}^n}\left\{\frac{1}{2}\left\|x-q\right\|_2^2+\sum_{i=1}^k\sqrt{x^TQ_ix}\right\}.$$

Consider the function $f(x)=\frac{1}{2}\left\|x-q\right\|_2^2+\sum_{i=1}^k\sqrt{x^TQ_ix}$. It is twice differentiable on $\mathbb{R}^n\setminus\{0\}$. However, it is non-smooth function on $\mathbb{R}^n$. Now define $f_\mu(x)=$

$\frac{1}{2}\left\|x-q\right\|_2^2 + \sum_{i=1}^{k} \sup_{c\in\mathcal{E}(0,Q_i)} \left\{\langle x,c\rangle - \frac{\mu}{2}\|x\|_2^2\right\}$. It is a differentiable function and its gradient is Lipschitz. The following holds

$$f_\mu(x) = \frac{1}{2}\left\|x-q\right\|_2^2 + \frac{k\|x\|_2}{2\mu} - \sum_{i=1}^{k}\frac{\mu}{2} d\left(\frac{x}{\mu};\mathcal{E}(0,Q_i)\right)^2,$$
$$\nabla f_\mu(x) = x - q + \sum_{i=1}^{k} \Pi_{\mathcal{E}(0,Q_i)}\left(\frac{x}{\mu}\right).$$

In addition the Lipschitz constant of the spacial gradient ∇f_μ is $L_\mu = \frac{k}{\mu} + 1$.

We apply the Nesterov first-order method for the function $f_\mu(x)$. First, we define in initial conditions u_0, v_0 and a parameter μ. The procedure is as follows

$$u_{k+1} = v_k - \frac{1}{L_\mu}\nabla f_\mu(v_k),$$
$$v_{k+1} = u_{k+1} + \frac{\sqrt{L_\mu} - \sqrt{2}}{\sqrt{L_\mu} + \sqrt{2}}\left(u_{k+1} + v_k\right).$$

To get an ε-solution which corresponds to have u_k such that $f_\mu(u_k) - f(u^*) < \varepsilon$, we need to perform $O\left(\frac{1}{\sqrt{\varepsilon}}\log\frac{1}{\varepsilon}\right)$ iterations when the parameter $\mu \sim \frac{\varepsilon}{\sum\|Q_i\|_2^2}$. This means that we need a small parameter μ to achieve a high precision solution. This behavior is certainly not adequate for computing the Minkowski sum of full ellipsoids for practical applications.

2.2.4 Primal-Dual algorithm

We now present the primal-dual algorithm described [16] to solve

$$\arg\min_{x\in\mathbb{R}^n}\left\{G(x) + F(Kx)\right\},$$

where the matrix $K \in \mathbb{R}^{m\times k}$, the functions $G \in \Gamma_0(\mathbb{R}^n)$ and $F \in \Gamma_0(\mathbb{R}^{mk})$. The primal-dual algorithm is an iterative procedure which reads as follows

$$
\begin{aligned}
u_{k+1} &= (I + \sigma\partial F^*)^{-1}(u_k - \sigma K v_k),\\
v_{k+1} &= (I + \tau\partial G)^{-1}(v_k + \tau K^\top u_{k+1}),\\
\bar{v}_{k+1} &= v_{k+1} + \theta(v_{k+1} - v_k),
\end{aligned}
$$

where the parameters $\tau > 0$, $\sigma > 0$ and $\theta \in [0, 1]$.

For the projection problem of interest, we define $G(x) = \frac{1}{2}\left\|x - q\right\|_2^2$. We also defined the function F as follows $F\left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}\right) = \sum_{i=1}^{k} \sqrt{y_i^T Q_i y_i}$ and the matrix $K = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}$ where the vectors $y_i \in \mathbb{R}^n$ for $i = 1, \cdots, k$ and I stands for a $n \times n$ identity matrix. We can change the functions G, F and the matrix K depending on the problem. We refer the reader to [58] for examples relevant to optimal control problems. We will also present possible variants in chapter 3.

2.3 Our Proposed Algorithm based on Newton's Method

In this section, we propose our algorithm based on Newton's method. First, we introduce general Newton's Method when the function f is twice differentiable. We denote the spacial gradient of f by $\nabla_x f(x)$ and the Hessian matrix of f by $\mathbf{H}(x)$. Newton's method is defined as follows

1 Newton Method generates a sequence $\{z_k\}_{k\in\mathbb{N}}$ by the recurrence formula,

$$z_{k+1} = z_k + d_k$$

where d_k is defined as

$$\nabla f(z_k) + \mathbf{H}(z_k)d_k = 0$$

To show that the sequence z_k converges in a some neighborhood of x^*, we need the Hessian matrix $\mathbf{H}(z_k)$ is non-singular on some neighborhood $\Omega \subset \mathbb{R}^n$ and is Lipschitz continuous on $\Omega \subset \mathbb{R}^n$ [14].

2.3.1 Quadratic Convergence of Newton's Method

In this section, we show that our Newton's Method converges quadratically. We use the dual problem in the previous section. The function, $f(x) = \frac{1}{2}\left\|x - q\right\|^2 + \sum_{i=1}^{k} \sqrt{x^T Q_i x}$ is twice differentiable on $\mathbb{R}^n \setminus \{0\}$. We have that on $\mathbb{R}^n \setminus \{0\}$, and

$$\nabla f(x) = x - q + \sum_{i=1}^{k} \frac{Q_i x}{\sqrt{x^T Q_i x}},$$
$$\mathbf{H}(x) = \left(I + \sum_{i=1}^{k} \frac{Q_i}{\sqrt{x^T Q_i x}} - \sum_{i=1}^{k} \frac{Q_i x (Q_i x)^T}{\sqrt{x^T Q_i x^3}} \right).$$

We need to consider two cases.

- The point q is inside of the Minkowski sum of ellipsoids,i.e., $q \in \sum_{i=1}^{k} \mathcal{E}(0, Q_i)$. In this case, we know that $\arg\min_{x\in\mathbb{R}^n} \{f(x)\} = 0$. We cannot use Newton's method because $\nabla f(x)$ does not exist at $x = 0$. We have to use (or modified) Gilbert method.

- The point q is outside of the Minkowski sum of ellipsoids. $q \notin \sum_{i=1}^{k} \mathcal{E}(0, Q_i)$

In this case, we know that $\arg\min_{x\in\mathbb{R}^n}\{f(x)\} \neq 0$. We wish to use Newton's method for this case to obtain quadratic convergence.

Since the function $f(x)$ is 0 - coercive and continuous function, the function $f(x)$ has a minimizer. Since the function $f(x)$ is strictly convex, the function $f(x)$ has at most one minimizer. Thus, the function $f(x)$ has a unique minimizer x^* and the spacial gradient $\nabla f(x)$ is an increasing function and has only one zero.

We want to show that Newton's method $\{z_k\}$ is converging quadratically to x^* with an initial data z_0 which in a some neighborhood of x^*. To show that the sequence z_k is converging in a some neighborhood of x^*, we need the Hessian matrix $\mathbf{H}(z_k)$ is non-singular on some neighborhood $\Omega \subset \mathbb{R}^n$ and is Lipschitz continuous on $\Omega \subset \mathbb{R}^n$ [14].

First, we will show that the Hessian matrix $\mathbf{H}(x)$ is non-singular. Assume that the vectors $x \neq 0$ and $v \in \mathbb{R}^n$ and we have

$$\begin{aligned} v^T\mathbf{H}(x)v &= \left(v^Tv + \sum_{i=1}^{k}\frac{v^TQ_iv}{\sqrt{x^TQ_ix}} - \sum_{i=1}^{k}\frac{v^TQ_ix(Q_ix)^Tv}{\sqrt{x^TQ_ix}^3}\right), \\ &= \left(v^Tv + \sum_{i=1}^{k}\left(\frac{(v^TQ_iv)(x^TQ_ix) - (v^TQ_ix)^2}{\sqrt{x^TQ_ix}^3}\right)\right). \end{aligned}$$

By Cauchy–Schwarz inequality, $(v^TQ_iv)(x^TQ_ix) - (v^TQ_ix)^2 \geq 0$. Thus, the Hessian matrix $\mathbf{H}(x)$ is symmetric positive definite which implies that the Hessian matrix $\mathbf{H}(x)$ is non-singular.

Next, we will show that the Hessian matrix $\mathbf{H}(z)$ is Lipschitz continuous on $\Omega \subset \mathbb{R}^n$. Assume that the neighborhood Ω is bounded and there exists an open set $O \ni 0$ such that $O \cap \Omega = \emptyset$. For given vectors $x, y \in \Omega$, we want to show

that there exists $K > 0$ such that $\left\|\mathbf{H}(x) - \mathbf{H}(y)\right\| \leq K \left\|x - y\right\|$. It is enough to show that the functions $g(x) = \frac{Q}{\sqrt{x^T Q x}}$ and $h(x) = \frac{Qx(Qx)^T}{\sqrt{x^T Q x^3}}$ are Lipschitz continuous on the neighborhood $\Omega \subset \mathbb{R}^n$ where the matrix Q is symmetric positive definite. To show the function $g(x)$ is Lipschitz continuous, we note that

$$\begin{aligned}\left\|g(x) - g(y)\right\|_2 &= \left\| \frac{Q}{\sqrt{x^T Q x}} - \frac{Q}{\sqrt{y^T Q y}} \right\|_2, \\ &= \|Q\|_2 \left| \frac{\sqrt{x^T Q x} - \sqrt{y^T Q y}}{\sqrt{x^T Q x}\sqrt{y^T Q y}} \right|,\end{aligned}$$

and we have that $\|Q\|_2$ and $\sqrt{x^T Q x}\sqrt{y^T Q y}$ are bounded. Thus, we have

$$\left\|g(x) - g(y)\right\|_2 \leq C_1 \left| \|x\|_Q - \left\|y\right\|_Q \right|.$$

By the triangular inequality, we get

$$\left\|g(x) - g(y)\right\|_2 \leq C_1 \left\|x - y\right\|_Q.$$

Since the matrix Q norm and 2-norm are equivalent, we have

$$\left\|g(x) - g(y)\right\|_2 \leq C_2 \left\|x - y\right\|_2,$$

which completes the proof. To show that the function $h(x)$ is Lipschitz continuous, we break the following $h(x) - h(y)$ into two parts,

$$h(x) - h(y) = \left(\frac{Qx(Qx)^T - Qy(Qy)^T}{\sqrt{x^T Q x}^3} \right) - \left(\frac{Qy(Qy)^T}{\sqrt{y^T Q y}^3} - \frac{Qy(Qy)^T}{\sqrt{x^T Q x}^3} \right).$$

For the first part, since we have that $\sqrt{x^TQx}$ is bounded, we have

$$\left\| \frac{Qx(Qx)^T - Qy(Qy)^T}{\sqrt{x^TQx}^3} \right\|_2 \leq C_1 \left\| Q\left(xx^T - yy^T\right)Q^T \right\|_2 .$$

Since we know that 2-norm is a sub-multiplicative matrix norm, we get

$$\left\| \frac{Qx(Qx)^T - Qy(Qy)^T}{\sqrt{x^TQx}^3} \right\|_2 \leq C_2 \left\| xx^T - yy^T \right\|_2 .$$

Since we know that Frobenius norm is greater than or equal to 2-norm,

$$\left\| \frac{Qx(Qx)^T - Qy(Qy)^T}{\sqrt{x^TQx}^3} \right\|_2 \leq C_2 \left\| xx^T - yy^T \right\|_F .$$

By the definition of Frobenius norm, we have

$$\begin{aligned} \left\| \frac{Qx(Qx)^T - Qy(Qy)^T}{\sqrt{x^TQx}^3} \right\|_2 &= C_2 \sqrt{\operatorname{tr}\left(\left(xx^T - yy^T\right)^T \left(xx^T - yy^T\right)\right)}, \\ &= C_2 \sqrt{\|x\|_2^4 + \left\|y\right\|_2^4 - 2\langle x, y\rangle^2}. \end{aligned}$$

By Cauchy–Schwarz inequality, we have

$$\begin{aligned} \left\| \frac{Qx(Qx)^T - Qy(Qy)^T}{\sqrt{x^TQx}^3} \right\|_2 &\leq C_2 \sqrt{\|x\|_2^4 + \left\|y\right\|_2^4 + 2\left\|x\right\|_2^2 \left\|y\right\|_2^2 - 4\langle x, y\rangle^2}, \\ &= C_2 \sqrt{\left\|x - y\right\|_2^2 \left\|x + y\right\|_2^2}. \end{aligned}$$

We have that the norm $\left\|x + y\right\|_2$ is bounded and we get the following,

$$\left\| \frac{Qx(Qx)^T - Qy(Qy)^T}{\sqrt{x^TQx}^3} \right\|_2 \leq C_3 \left\| x - y \right\|_2 .$$

The second part uses a similar idea which we used to show that the function $g(x)$

is Lipschitz continuous. We have

$$\left\| \frac{Qy(Qy)^T}{\sqrt{y^TQy}^3} - \frac{Qy(Qy)^T}{\sqrt{x^TQx}^3} \right\|_2 = \left\| Qy(Qy)^T \right\|_2 \left| \frac{\sqrt{x^TQx}^3 - \sqrt{y^TQy}^3}{\sqrt{x^TQx}^3 \sqrt{y^TQy}^3} \right|.$$

We have that the norms $\left\| Qy(Qy)^T \right\|_2$, $\left\| \sqrt{x^TQx} \right\|_2$ and $\left\| \sqrt{y^TQy} \right\|_2$ are bounded. Therefore we get

$$\begin{aligned} \left\| \frac{Qy(Qy)^T}{\sqrt{y^TQy}^3} - \frac{Qy(Qy)^T}{\sqrt{x^TQx}^3} \right\|_2 &\leq C_1 \left| \|x\|_Q^3 - \left\|y\right\|_Q^3 \right|, \\ &\leq C_2 \left| \|x\|_Q - \left\|y\right\|_Q \right|. \end{aligned}$$

Similar to the function $g(x) - g(y)$, we use the triangular inequality and the fact that the matrix Q norm and 2-norm are equivalent to obtain

$$\begin{aligned} \left\| \frac{Qy(Qy)^T}{\sqrt{y^TQy}^3} - \frac{Qy(Qy)^T}{\sqrt{x^TQx}^3} \right\|_2 &\leq C_2 \left\| x - y \right\|_Q, \\ &\leq C_3 \left\| x - y \right\|_2. \end{aligned}$$

We can now show that $h(x)$ is Lipschitz continuous. The following holds

$$\begin{aligned} \left\| h(x) - h(y) \right\|_2 &= \left\| \left(\frac{Qx(Qx)^T - Qy(Qy)^T}{\sqrt{x^TQx}^3} \right) - \left(\frac{Qy(Qy)^T}{\sqrt{y^TQy}^3} - \frac{Qy(Qy)^T}{\sqrt{x^TQx}^3} \right) \right\|_2, \\ &\leq \left\| \frac{Qx(Qx)^T - Qy(Qy)^T}{\sqrt{x^TQx}^3} \right\|_2 + \left\| \frac{Qy(Qy)^T}{\sqrt{y^TQy}^3} - \frac{Qy(Qy)^T}{\sqrt{x^TQx}^3} \right\|_2, \\ &\leq C \left\| x - y \right\|_2. \end{aligned}$$

Therefore, the Hessian $\mathbf{H}(z)$ is Lipschitz continuous on the neighborhood $\Omega \subset \mathbb{R}^n$ and we have showed that Newton's Method converges quadratically when the vector $q \notin \sum_{i=1}^{k} \mathcal{E}(0, Q_i)$. The problem of Newton's Method is that the initial guess has to be close enough to the solution and actually the maximum distance between

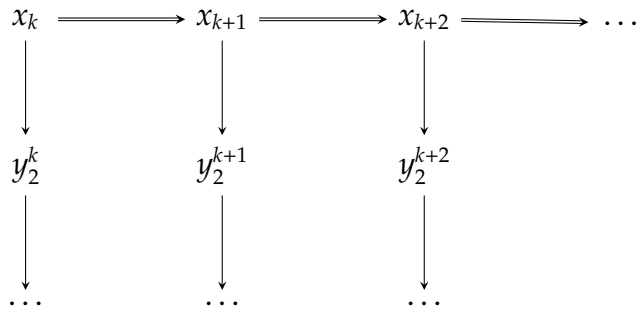
the solution and the initial guess is related to the condition number of the matrix. We propose to use Gilbert Method as an initial guess.

2.3.2 Algorithm for Newton's method

Here we propose our algorithm which combines Gilbert-based convergent scheme with Newton's Method. Our algorithm works as follows

- We run (modified) Gilbert method and denote by $\{x_k\}_{k\in\mathbb{N}}$ its sequence. We know that x_k converges to the projection of the Minkowski sum of ellipsoids.
- We try to perform of few iterations of Newton's method with initial guess x_k. We denote theses sequences by $\{y_l^k\}_{l\in\mathbb{N}}$. If Newton's method succeeds for some starting point x_k then we stop the whole procedure; if not we start again Newton's method with x_{k+1}.

The following figure illustrates our proposed algorithm.



If we can find k^* such that the sequence $\{y_l^{k^*}\}_{l\in\mathbb{N}}$ converges, the algorithm would be $\{ x_1, x_2, \cdots, x_{k^*}, y_{l+1}^{k^*}, y_{l+2}^{k^*}, \cdots \}$. Since it is not easy to find such k^* analytically,

we use each x_k sequentially. If $\{y_l^k\}_{l\in\mathbb{N}}$ does not converge to z^*, we keep going to $\{y_l^{k+1}\}_{l\in\mathbb{N}}$. Here, we have to tell whether $\{y_l^k\}_{l\in\mathbb{N}}$ converges to z^* or not.

The optimality condition reads $\nabla f(z^*) = 0$. Numerically we can use the following stopping criteria $\left\|\nabla f(z^*)\right\|_2 < \epsilon$ since $f(x)$ is a convex function and $\nabla f(x)$ is a continuous function. If $\{y_l^k\}_{l\in\mathbb{N}}$ does not converge to z^*, $\left\|\nabla f\left(y_l^k\right)\right\|_2 > C$ for some constant $C > 0$. Therefore, in practice, we choose a suitable C and $\bar{l}$ such that if $\left\|\nabla f\left(y_{\bar{l}}^k\right)\right\|_2 > C$, we assume that $\{y_l^k\}_{l\in\mathbb{N}}$ doesn't converge to z^* and move on to the next step.

When the point q is inside of the Minkowski sum of ellipsoids, $q \in \sum_{i=1}^k \mathcal{E}(0, Q_i)$, we know that Newton's method is not converging. To avoid slow convergence of Gilbert Method, we use the inner approximation ellipsoid to tell whether q is inside of the inner approximation ellipsoid for each step or not.

2.3.3 Parallel Algorithm

Since the sequences $\{x_k\}_{k\in\mathbb{N}\setminus\{1,\cdots,j\}}$ and $\{y_l^j\}_{l\in\mathbb{N}}$ are independent of each other, we can compute the sequence $\{x_k\}_{k\in\mathbb{N}\setminus\{1,\cdots,j\}}$ and the sequence $\{y_l^j\}_{l\in\mathbb{N}}$ in parallel. Assume that we have $m \geq 2$ threads. We use the main thread to calculate the sequence $\{x_k\}_{k\in\mathbb{N}}$ and $m-1$ subordinate threads to compute the sequence $\{y_l^k\}_{l\in\mathbb{N}}$. If one of $k-1$ threads succeeds to converges to z^*, we stop the whole procedure and get the result. If $\{y_l^j\}_{l\in\mathbb{N}}$ doesn't converge, we get the current x_k from the main thread and the subordinate thread compute the sequence $\{y_l^k\}_{l\in\mathbb{N}}$.

More generally our proposed algorithm can implemented parallel as follows: one threads computes the (modified) Gilbert sequence (x_k) while other threads

perform Newton's methods. Let $\{x_k\}_{k\in\mathbb{N}}$ be the convergence scheme such as Gilbert algorithm or Modified Gilbert algorithm. Let $\{y_l^k\}_{l\in\mathbb{N}}$ be a Newton's Method with $y_1^k = x_k$.

2.4 Numerical Results

We now present numerical results for Gilbert method and Modified Gilbert method. We also present numerical results for Newton's method and Gilbert method. The numerical experiments show the our proposed algorithm enjoy quadratic convergence.

In this section, we assume that q is the point to use for the projection and the matrix $\{Q_i\}$ is a set of ellipsoids centered at the origin. We did 100 times numerical experiment for each example in Python to compare the performance of Gilbert Algorithm, Modified Gilbert Algorithm and Newton's method. Computations we performed on i7-8550U CPU @ 1.80 GHz (4.0 GHz) 4 cores and 8 threads and 8 GB of RAM. Python runs in 3.7.4 and we use multiprocessing module in Python. We use one thread for Gilbert Method and Modified Gilbert Method and two threads for Newton's Method for comparisons.

We consider n-dimensional random symmetric positive matrices P_l sampled from a uniform distribution taking values between -0.5 and 0.5 when $i < j$ while $[P_l]_{ii}$ are sampled from a uniform distribution taking value between 9.5 and 10.5.

Then the matrices Q_1 and Q_2 and the vector q are defined as follows

$$Q_1 = P_1 + \begin{bmatrix} 10^k & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \; Q_2 = P_2, \; q = \begin{bmatrix} 0 \\ 10 \\ \vdots \\ 0 \end{bmatrix}.$$

Here the parameter k stands for the degeneracy of the matrix and the parameter n stands for the dimensions of the matrices. We present the average number of iterations performed for each algorithm for various$k$ when $n = 5$ in table 2.1. We also present the average number of iterations performed for each algorithm for various n when $k = 7$ in table 2.2. Note that modified Gilbert method works well since the inner ellipsoid is a good approximation of Minkowski sum of $\{\mathcal{E}(0, Q_1), \mathcal{E}(0, Q_2)\}$.

Table 2.1: Average number of iterations for Gilbert and modified Gilbert when $n = 5$.

k	**Gilbert Method**	**Modified Gilbert Method**
1	95.3	93.3
2	101.9	94.2
3	107.8	98.9
4	132.0	113.6
5	200.0	193.5
6	375.8	180.4
7	889.9 (not converge)	164.1
8	1897.2 (not converge)	175.5
9	4307.7 (not converge)	242.3

Note that Gilbert method does not converge when $k \geq 7$ because of the limited precision of double floating used in the computations. We also terminate all algorithm if they take more that 10,000 steps to make sure that the code stops after the certain amount of time.

Table 2.2: Average number of iterations for Gilbert and modified Gilbert when $k = 7$.

n	**Gilbert Method**	**Modified Gilbert Method**
2	464.0	201.0
4	744.0	346.5
8	877.7	186.3
16	1678.3	447.6
32	4437.8	919.8

If the inner ellipsoid is a good approximation of Minkowski sum of ellipsoids, Modified Gilbert generally works well. In most cases, however there is no significant difference between Gilbert method and Modified Gilbert method since either the inner ellipsoid is not a good approximation of Minkowski sum of ellipsoids or Gilbert method also works well.

Let $\{K_l\}_{l=1,\cdots,m}$ be a zero matrix and add 10^k to j-th element which is randomly selected for each l. Let $Q_i = P_i + K_i$ and z^* be a random vector. Each element of z^* is a random value from the normal distribution $N(0, 1)$. We have that z^* is a vector that spacial gradient $\nabla f(z^*) = 0$ which means that

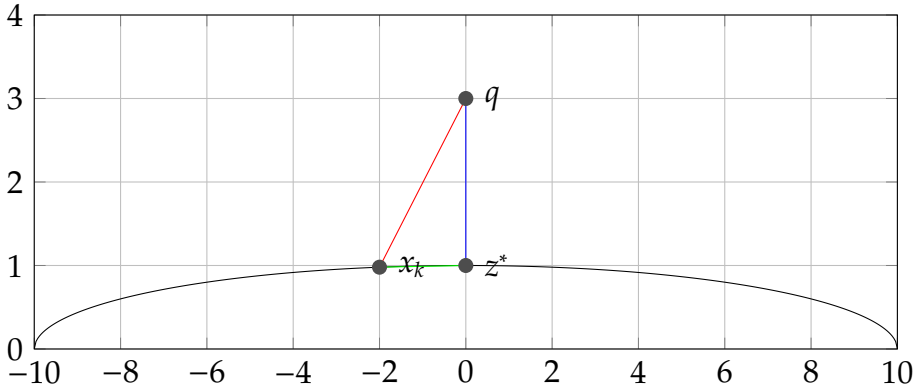
$$\nabla f(z^*) = z^* - q + \sum_{i=1}^{k} \frac{Q_i z^*}{\sqrt{z^{*T} Q_i z^*}} = 0,$$
$$q = z^* + \sum_{i=1}^{k} \frac{Q_i z^*}{\sqrt{z^{*T} Q_i z^*}}.$$

Since we know that the exact z^* in this case, we have used the stopping criterion for $\|z_{k+1} - z^*\| < \epsilon$. There is no significant difference between Gilbert method and modified Gilbert method. This is because the inner ellipsoid is not a good approximation for this example.

For Newton's method, we have used Gilbert Method as an initial guess and the parallel algorithm proposed in the previous section. We have three main aspects that we could change, the number of matrices, the dimension of the matrices and the parameter k for making the matrix have a bad condition number. We present the time result when the number of matrices m is 4, dimension of matrices n is 5 and we change k to compare Gilbert method and Newton's method. We also present the time result varying n with $k = 3.5$ and $m = 4$.

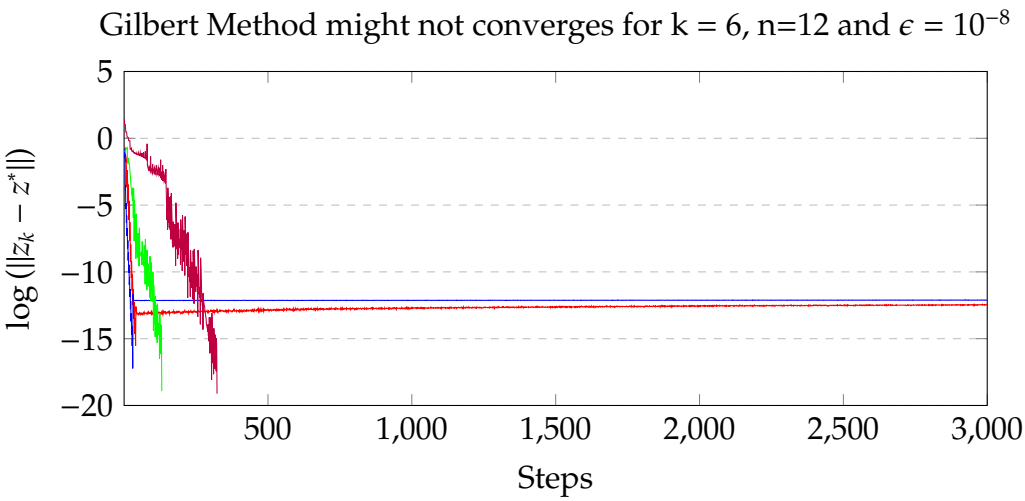
We know that $\|z_k\| - \|z^*\|$ decreases and for an extreme case like this example, $\|z_{k+1} - z^*\|$ could be much bigger than $\|z_k\| - \|z^*\|$. We present a figure that shows such a case. This is an ellipsoid $\mathcal{E}(0, Q)$ where $Q = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}$, $q = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$ and $z^* = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. We plot the ellipsoid $\mathcal{E}(0, Q)$, q, z^* and the point $\begin{bmatrix} -2 \\ \sqrt{0.96} \end{bmatrix}$.

Ellipsoid $\mathcal{E}(0, Q)$ with a bad condition number

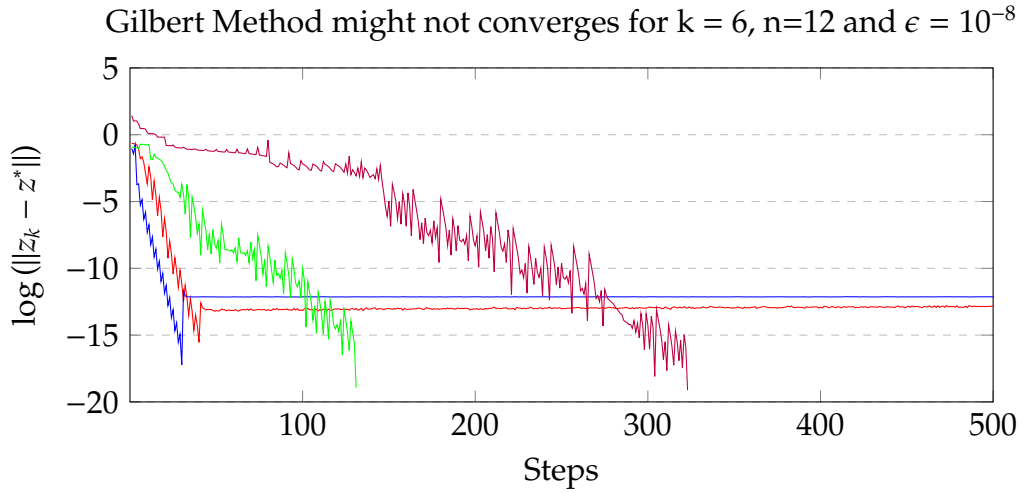


Let the length of red line be c, the length of blue line be b and the length of green line be a. Since it is almost flat for this ellipsoid case, we have that $c^2 \approx b^2 + a^2$. We have that $c - b \approx \frac{a^2}{\sqrt{b^2+a^2}+b}$. We have that when $a = \epsilon$, $c - b$ will be significantly smaller than a because we treat the length b is much bigger than ϵ. This implies

that $\|z_{k+1} - z^*\|$ could be much bigger than $\|z_k\| - \|z^*\|$. Thus, we might not get a convergence of z_k with $\epsilon = 10^{-8}$ for a double precision floating point format. We plot the graph for Gilbert method. We truncated on 3,000 steps and the graph clearly show that sometimes Gilbert does not converge.



To clearly show what happens between 1 and 500 steps, we change the range of the domain (Steps).



We now present numerical experiments using the same setting as before. Ta-

ble 2.3 provides the average number of iterations for our proposed method and Gilbert method for several k when $n = 5$. Table 2.4 provides the average number of iterations for our proposed method and Gilbert method for several n when $k = 3.5$. We truncated on 10,000 iteration for Gilbert Method because for some cases, Gilbert Method never converges. This is because we use floating point number system with double precision and round-off error might be significant for some cases.

Table 2.3: Average number of iterations for Gilbert and our proposed Newton's based approach when $n = 5$.

k	**Gilbert Method**	**Newton's Method**
3	108.4	91.7
4	110.6	92.1
5	1610.1	94.3
6	2341.4	95.5
7	2246.5	148.6
8	3878.0	91.3

Table 2.4: Average number of iterations for Gilbert and our proposed Newton's based approach when $k = 3.5$

n	**Gilbert Method**	**Newton's Method**
16	144.8	96.8
32	261.7	117.7

To illustrate that our approach based on Newton's method has a quadratic convergence rate, we need to use a higher precision. This is because Newton's method converges in a few steps which does not show a quadratic convergence clearly. We use Matlab function which is a symbolic method to implement the code and make a plot for the same example above. We use log of the actual error, $||x_n - x^*||$ for convenience. To illustrate the quadratic convergence rate, we use the

definition of the quadratic convergence rate,

$$\limsup \frac{\|z_{k+1} - z^*\|}{\|z_k - z^*\|^2} = r.$$

Since we are interested in the tail of the sequence $\{z_k\}$, we assume that for any k such that $k \geq N$,

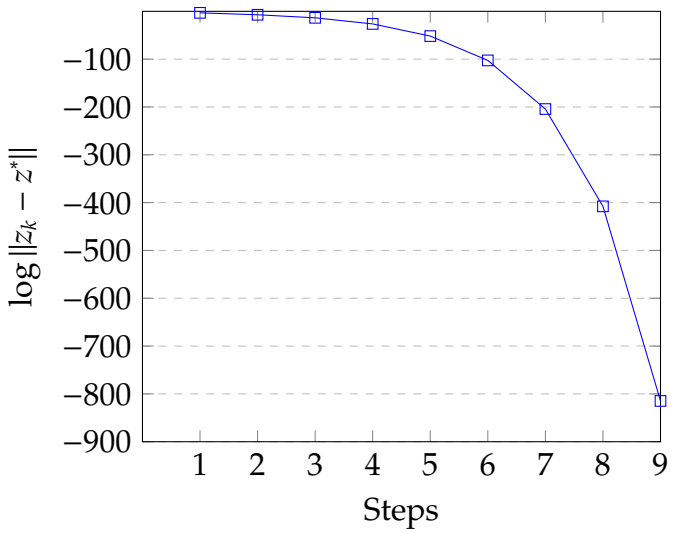
$$\frac{\|z_{k+1} - z^*\|}{\|z_k - z^*\|^2} \approx r.$$

We use the property of the log function,

$$\log \|z_{k+1} - z^*\| - 2 \log \|z_k - z^*\| \approx \log r.$$

Let $v_k \approx \log \|z_k - z^*\| + \log r$. We have that $v_{k+1} \approx 2v_k$. This means that for Newton's method, we expect the log of the error decays exponentially.

Numerical convergence rate for our proposed method.



Similarly, for Gilbert method, we assume that for any k such that $k \geq N$,

$$\frac{\|z_{k+1} - z^*\|}{\|z_k - z^*\|} \approx r.$$

We use the property of the log function,

$$\log \|z_{k+1} - z^*\| - \log \|z_k - z^*\| \approx \log r.$$

Let $v_k \approx \log \|z_k - z^*\|$. We have that $v_{k+1} \approx v_k + \log r$. This means that we expect the log of the error decay linearly. For the illustration, we use $\log(\|z_k\| - \|z^*\|)$ because we only have that $\|z_k\|$ decreases while $\|z_k - z^*\|$ doesn't.

In summary, our proposed method is less sensitive to numerical rounding errors due to floating point compared to Gilbert method. It is clear form the theory and numerical experiments that Newton's method is better than Gilbert Method in terms of the convergence rate.

2.5 Summary

In this chapter, we have explore the algorithm for computing the projection onto the Minkowski sum of full ellipsoids. We have presented a novel algorithm with quadratic convergence. However, our approach requires a good initial guess which needs to be close to the solution to achieve quadratic convergence. This initial guess is obtained from (modified) Gilbert method.

Recall that the assumption of having full ellipsoids is mandatory for our proposed algorithm. In the next chapter we remove this assumption and explore several other possible algorithm to handle these degenerate cases.

Chapter Three

Linear Optimal Control Problem with degenerate Ellipsoidal constraints

3.1 Introduction

In this chapter, we present several algorithm to solve certain optimal control problems described in chapter 1 when the ellipsoid is not full, i.e., degenerated. Recall that our proposed algorithm in chapter 2 cannot be applied in this setting because it requires ellipsoids to be full. In this chapter we present several optimization based algorithms to solve this class of optimal control problem.

The outline for the remainder of this chapter is as follows. In section 3.2, we formulate the optimal control problems we are interested in and connection to the Hopf representation formula. Section 3.3 describes the standard primal-dual optimization algorithm used in [58, 60]. Section 3.4 describes non-linear primal-dual formulations of the optimization problem. This formulation allows for several choices that yields in practice different numerical solvers. We also propose a regularization-based approach to efficiently compute an initial guess using the algorithm presented in chapter 2.

3.2 Optimal control formulation

We consider certain optimal control problems with linear dynamics, which read as follows:

$$S(x,t) = \min\left\{ \int_t^T L(u(s))ds + J_x(x(T)) \colon x(t) = x, x'(s) = Ax(s) + Bu(s)\ \forall s \in [t,T] \right\}, \tag{3.1}$$

where the running cost $L : \mathbb{R}^m \to \mathbb{R}$ is a proper, lower semicontinuous, convex, 1-coercive function, the terminal cost $J_x : \mathbb{R}^n \to \mathbb{R}$ is convex, and A, B are matrices

with real entries. By reversing time (i.e., let $s = T - t$), S solves the following forward-time HJ PDE:

$$\begin{cases} \dfrac{\partial S(x,s)}{\partial s} + H_x(x, \nabla_x S(x,s)) = 0 & x \in \mathbb{R}^n, s > 0, \\ S(x,0) = J_x(x) & x \in \mathbb{R}^n, \end{cases} \tag{3.2}$$

whose Hamiltonian $H_x \colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ equals

$$H_x(x,p) = \sup_{u \in \mathbb{R}^n} \langle -(Ax + Bu), p \rangle - L(u) = -\langle Ax, p \rangle + L^*(-B^T p), \tag{3.3}$$

where A is an $n \times n$ matrix, and B is an $n \times m$ matrix.

After the change of variable $z(s) = e^{-sA}x(s)$ (with $z = e^{-tA}x$), the optimal control problem (3.1) becomes

$$\tilde{S}(z,t) = \min \left\{ \int_t^T L(u(s))ds + J_x(e^{TA}z(T)) \colon z(t) = z, z'(s) = e^{-sA}Bu(s) \; \forall s \in [t,T] \right\}. \tag{3.4}$$

Define J_z by $J_z(z) = J_x(e^{TA}z)$. Then, the corresponding HJ PDE reads

$$\begin{cases} \dfrac{\partial \tilde{S}(z,s)}{\partial s} + H_z(s, \nabla_z \tilde{S}(z,s)) = 0 & z \in \mathbb{R}^n, s > 0, \\ \tilde{S}(z,0) = J_z(z) & z \in \mathbb{R}^n, \end{cases} \tag{3.5}$$

where the Hamiltonian $H_z : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ equals

$$H_z(s,p) = \sup_{u \in \mathbb{R}^n} \left\{ \langle -e^{-sA}Bu, p \rangle - L(u) \right\} = L^*(-B^T e^{-sA^T} p). \tag{3.6}$$

Under some assumptions, the solution of the corresponding HJ PDE is given by

the generalized Hopf formula, which reads

$$\tilde{S}(z,s) = \sup_{p\in\mathbb{R}^n}\left\{\langle z,p\rangle - \int_0^s L^*(-B^T e^{-\tau A^T}p)d\tau - J_z^*(p)\right\}. \tag{3.7}$$

Then, the optimal control $\bar{u}\colon [t,T]\to\mathbb{R}^m$ in problem (3.4) is given by

$$\bar{u}(T-s) = \arg\max_{u\in\mathbb{R}^n}\left\{\langle -e^{-sA}Bu,\bar{p}\rangle - L(u)\right\} = \nabla L^*(-B^T e^{-sA^T}\bar{p}), \tag{3.8}$$

where $\bar{p}$ is the maximizer in (3.7), which also equals $\nabla_z\tilde{S}(z,s)$. Note that by straightforward computation, we have

$$J_z^*(p) = \sup_{z\in\mathbb{R}^n}\langle z,p\rangle - J_z(z) = \sup_{z\in\mathbb{R}^n}\langle z,p\rangle - J_x(e^{TA}z) = \sup_{x\in\mathbb{R}^n}\langle e^{-TA}x,p\rangle - J_x(x) = J_x^*(e^{-TA^T}p). \tag{3.9}$$

After discretization in time, the optimization problem in (3.7) is approximated by the following multi-time Hopf formula:

$$\sup_{p\in\mathbb{R}^n}\left\{\langle z,p\rangle - \sum_{j=1}^k c_j L^*(-B^T e^{-\tau_j A^T}p) - J_z^*(p)\right\} = -\min_{p\in\mathbb{R}^n} F(Kp) + G(p), \tag{3.10}$$

where $\{c_j\}$ and $\{\tau_j\}$ are the weights and quadrature points of any quadrature rule approximating the integration over $[0,s]$ and F, G, and K are defined, such that

$$F(Kp) = \sum_{j=1}^k c_j L^*(-B^T e^{-\tau_j A^T}p), \quad G(p) = J_z^*(p) - \langle z,p\rangle. \tag{3.11}$$

Assume that

$$L^*(w) = \sqrt{\langle w, Mw\rangle}, \quad \forall w\in\mathbb{R}^m, \tag{3.12}$$

where M is a symmetric, positive definite matrix (i.e., $L = I_{\{x:\langle x,M^{-1}x\rangle\le 1\}}$, or in other words, we constrain the control to lie in the matrix associated with the matrix M),

and the terminal cost J_x is a quadratic function ($J_x(x) = \frac{1}{2}\langle x, M_J x\rangle - \langle b_J, x\rangle + a_J$). Then, the function G is also quadratic:

$$\begin{aligned} G(p) = J_z^*(p) - \langle z, p\rangle &= \frac{1}{2}\langle e^{-TA^T}p + b_J, M_J^{-1}(e^{-TA^T}p + b_J)\rangle - \langle z, p\rangle - a_J, \\ &= \frac{1}{2}\langle p + e^{TA^T}b_J, e^{-TA}M_J^{-1}e^{-TA^T}(p + e^{TA^T}b_J)\rangle - \langle z, p\rangle - a_J. \end{aligned} \tag{3.13}$$

3.3 A standard Primal-dual formulation

We want to solve an optimization problem of the following form:

$$\arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ G(\boldsymbol{x}) + \sum_{i=1}^{k} c_i \sqrt{\langle \boldsymbol{x}, \boldsymbol{Q}_i \boldsymbol{x}\rangle}), \right\}, \tag{3.14}$$

where $G \in \Gamma_0(\mathbb{R}^n)$, $c_i > 0$ (which in [58] corresponds to the weights in the quadrature rule used for discretization in time; specifically, [58] uses $c_i = \Delta t, \forall i$), and $\boldsymbol{Q}_i \in \mathbb{R}^{n\times n}$. For our problem of interest, we have

$$G(\boldsymbol{x}) = \frac{1}{2}\left\langle \boldsymbol{x} - \boldsymbol{q}, \boldsymbol{V}^{-1}(\boldsymbol{x} - \boldsymbol{q})\right\rangle - \langle \boldsymbol{b}, \boldsymbol{x}\rangle + a, \tag{3.15}$$

for some vectors $\boldsymbol{b}, \boldsymbol{q} \in \mathbb{R}^n$, scalar $a \in \mathbb{R}$, and symmetric positive definite matrix $\boldsymbol{V} \in \mathbb{R}^{n\times n}$. The matrices $\boldsymbol{Q}_i$ are assumed to be either symmetric positive definite (SPD) or symmetric positive semi-definite (SPSD) and to have the following structure. Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$, $\boldsymbol{M} \in \mathbb{R}^{m\times m}$, $T > 0$, and $t_i \in \{1, \ldots, k\}$. Then,

$$\begin{aligned} \boldsymbol{Q}_i &= (-e^{-(T-t_i)\boldsymbol{A}}\boldsymbol{B})\boldsymbol{M}(-e^{-(T-t_i)\boldsymbol{A}}\boldsymbol{B})^\top, \\ &= e^{-(T-t_i)\boldsymbol{A}}\boldsymbol{B}\boldsymbol{M}\boldsymbol{B}^\top e^{-(T-t_i)\boldsymbol{A}^\top}. \end{aligned}$$

As in (3.1), the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ refer to the matrices of the same name in the dynamics of the optimal control problem

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t),$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector and $\boldsymbol{u}$ is some control input defined in $\mathcal{U} \subset \mathbb{R}^m$. The matrix $\boldsymbol{M}$ is some transformation to give the appropriate problem-specific control bound (i.e., to constrain the control to the ellipsoid associated with the matrix $\boldsymbol{M}$). In Section VI of [58], the matrix $\boldsymbol{V}$ is identified as $\boldsymbol{V}(0)$ and takes the form

$$\boldsymbol{V} = e^{T\boldsymbol{A}^\top} \boldsymbol{W}^{-1} e^{T\boldsymbol{A}} \qquad (\implies \boldsymbol{V}^{-1} = e^{-T\boldsymbol{A}} \boldsymbol{W} e^{-T\boldsymbol{A}^\top}),$$

where $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix that is associated with the terminal cost of the optimal control problem (or equivalently, the initial condition of the HJ PDE). Note in Section 3.2, we identify $\boldsymbol{W} = M_J^{-1}$, $\boldsymbol{q} = -e^{T\boldsymbol{A}^\top} b_J$, $\boldsymbol{b} = z$, and $a = -a_J$ (e.g., see (3.13) for more details).

Problem (3.14) can be written as the following minimization problem:

$$\arg\min_{\boldsymbol{x} \in \mathbb{R}^n} \{G(\boldsymbol{x}) + F(\boldsymbol{K}\boldsymbol{x})\}, \tag{3.16}$$

where $G \in \Gamma_0(\mathbb{R}^n)$ and $F \in \Gamma_0(\mathbb{R}^{n \times k})$. For this problem, we get to pick the operator $\boldsymbol{K}$ and the function F to use for the primal-dual algorithm. For example, we can define

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{I}_n \\ \boldsymbol{I}_n \\ \cdots \\ \boldsymbol{I}_n \end{bmatrix},$$

and

$$F([\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k]) = \sum_{i=1}^{k} c_i \sqrt{\langle \boldsymbol{y}_i, \boldsymbol{Q}_i \boldsymbol{y}_i \rangle}. \tag{3.17}$$

Alternatively, we could instead choose $\boldsymbol{K}$ and F as follows:

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{B}^\top e^{-\tau_1 \boldsymbol{A}^\top} \\ \boldsymbol{B}^\top e^{-\tau_2 \boldsymbol{A}^\top} \\ \vdots \\ \boldsymbol{B}^\top e^{-\tau_k \boldsymbol{A}^\top} \end{bmatrix},$$

and

$$F([\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k]) = \sum_{i=1}^{k} c_i \sqrt{\langle \boldsymbol{y}_i, \boldsymbol{M} \boldsymbol{y}_i \rangle}.$$

We discuss some possible choices of $\boldsymbol{K}$ and F more in depth later on.

3.4 Non-linear Primal-dual methods

We use the non-linear primal dual method described in [21] to solve (3.16), the m-th iteration is

$$\begin{aligned} x_{m+1} &= \arg\min_{x \in \mathbb{R}^n} \{G(x) + \langle \tilde{y}_m, Kx \rangle + \frac{1}{2\tau_m} \|x - x_m\|^2\}, \\ y_{m+1} &= \arg\min_{y \in \mathbb{R}^{n \times k}} \{F^*(y) - \langle y, K\tilde{x}_{m+1} \rangle + \frac{1}{2\sigma_m} \|y - y_m\|^2\}, \end{aligned} \tag{3.18}$$

where $\tilde{y}_m$ ($\tilde{x}_{m+1}$, resp.) is some linear combination of y_m and y_{m-1} (x_{m+1} and x_m, resp.).

In our case, the first equation in (3.18) is equivalent to

$$\begin{aligned} x_{m+1} &= \arg\min_{p\in\mathbb{R}^n}\{J_z^*(p) + \langle K^T\tilde{y}_m - z, p\rangle + \frac{1}{2\tau_m}\|p - x_m\|^2\}, \\ &= \arg\min_{p\in\mathbb{R}^n}\{J_z^*(p) + \frac{1}{2\tau_m}\|p - x_m + \tau_m(K^T\tilde{y}_m - z)\|^2\}. \end{aligned} \tag{3.19}$$

When J_x is quadratic ($J_x(x) = \frac{1}{2}\langle x, M_J x\rangle - \langle b_J, x\rangle + a_J$), it has an explicit formula

$$x_{m+1} = \left(\frac{I}{\tau_m} + e^{-TA}M_J^{-1}e^{-TA^T}\right)^{-1}\left(-e^{-TA}M_J^{-1}b_J + \frac{x_m}{\tau_m} - K^T\tilde{y}_m + z)\right). \tag{3.20}$$

If J_x is not quadratic, then we need to compute the proximal point of $\tau_m J_z(\frac{\cdot}{\tau_m})$ or $\tau_m J_z^*$:

$$x_{m+1} = \text{prox}_{\tau_m J^*}(x_m - \tau_m(K^T\tilde{y}_m - z)) = x_m - \tau_m(K^T\tilde{y}_m - z) - \text{prox}_{\tau_m J(\frac{\cdot}{\tau_m})}(x_m - \tau_m(K^T\tilde{y}_m - z)). \tag{3.21}$$

The second equation in (3.18) is equivalent to

$$\boldsymbol{y}_{m+1} = \arg\min_{\boldsymbol{y}\in\mathbb{R}^{n\times k}}\{F^*(\boldsymbol{y}) + \frac{1}{2\sigma_m}\|\boldsymbol{y} - \boldsymbol{y}_m - \sigma_m K\tilde{\boldsymbol{x}}_{m+1}\|^2\}. \tag{3.22}$$

There are two cases, depending on whether F is separable or not.

(a) If F is separable, i.e., $F(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k) = \sum_{i=1}^k c_i\sqrt{\langle \boldsymbol{y}_i, \boldsymbol{Q}_i\boldsymbol{y}_i\rangle}$. By straightforward computation, F^* is in $\Gamma_0(\mathbb{R}^{n\times k})$ and equals

$$F^*(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k) = \sup_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k\in\mathbb{R}^n}\sum_{i=1}^k(\langle \boldsymbol{y}_i, \boldsymbol{z}_i\rangle - c_i\sqrt{\langle \boldsymbol{y}_i, \boldsymbol{Q}_i\boldsymbol{y}_i\rangle}) = \sum_{i=1}^k I_{\{z\in(\ker Q_i)^\perp:\,\langle z, Q_i^\dagger z\rangle\le 1\}}(\frac{\boldsymbol{z}_i}{c_i}). \tag{3.23}$$

Then (3.22) can computed in parallel, i.e., $\boldsymbol{y}_{m+1} = (\boldsymbol{y}_{m+1,1}, \ldots, \boldsymbol{y}_{m+1,k})$, where

each $\boldsymbol{y}_{m+1,j} \in \mathbb{R}^n$ satisfies

$$\boldsymbol{y}_{m+1,j} = \arg\min_{\boldsymbol{y}\in\mathbb{R}^n}\{I_{\{z\in(ker\,Q_i)^\perp:\,\langle z,Q_i^\dagger z\rangle\le 1\}}(\frac{\boldsymbol{y}}{c_i}) + \frac{1}{2\sigma_m}\|\boldsymbol{y} - \boldsymbol{y}_{m,j} - \sigma_m(K\tilde{\boldsymbol{x}}_{m+1})_j\|^2\}. \quad (3.24)$$

In other words, $\boldsymbol{y}_{m+1,j}$ is the projection of $\boldsymbol{y}_{m,j} + \sigma_m(K\tilde{\boldsymbol{x}}_{m+1})_j$ on the ellipsoid $\{\boldsymbol{z} \in (ker\,Q_i)^\perp : \langle \boldsymbol{z}, Q_i^\dagger \boldsymbol{z}\rangle \le 1\}$ (if Q_i is not invertible, then this ellipsoid is degenerate).

(b) If F is not separable, i.e., $F(\boldsymbol{y}) = c_i\sqrt{\langle \boldsymbol{y}, \hat{\boldsymbol{Q}}_{i,\epsilon}\boldsymbol{y}\rangle}$, where the invertible matrix $\hat{Q}_{i,\epsilon}$ is the regularized matrix of Q_i with the regularization parameter ϵ (see Section 3.4.3). Then, we have

$$F^*(z) = I_{\sum_{i=1}^k\{z\in\mathbb{R}^n:\,\langle z,\hat{Q}_{i,\epsilon}^{-1}z\rangle\le c_i^2\}}(z), \quad (3.25)$$

and hence (3.22) is the projection of $\boldsymbol{y}_{m,j} + \sigma_m(K\tilde{\boldsymbol{x}}_{m+1})_j$ onto the Minkowski sum of ellipsoids $\sum_{i=1}^k\{\boldsymbol{z} \in \mathbb{R}^n : \langle \boldsymbol{z}, \hat{Q}_{i,\epsilon}^{-1}\boldsymbol{z}\rangle \le c_i^2\}$, which can be computed using the algorithm of chapter 2.

3.4.1 Possible Choices of K and F

Note that the function G as defined by (3.13) is determined directly by J_x, but we have the freedom to choose F and K. There are some ways to do that, which are summarized here:

(a) **First choice: simple form of** F. This choice (the non-accelerated version)

corresponds to the method used in [58]. In [58], they define

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{B}^\top e^{-\tau_1 \boldsymbol{A}^\top} \\ \boldsymbol{B}^\top e^{-\tau_2 \boldsymbol{A}^\top} \\ \vdots \\ \boldsymbol{B}^\top e^{-\tau_k \boldsymbol{A}^\top} \end{bmatrix} \text{ and } F([\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k]) = \sum_{i=1}^{k} c_i \sqrt{\langle \boldsymbol{y}_i, \boldsymbol{M}\boldsymbol{y}_i \rangle}. \tag{3.26}$$

Since F is separable, the second step in primal-dual method is equivalent to doing k **projections onto the same ellipsoid** of different points and hence can be computed in parallel. However, K may be very complicated. For instance, its condition number may be bad, which will influence the convergence rate of the primal-dual algorithm.

To obtain an accelerated algorithm, one needs to compute the appropriate matrix norm of $\boldsymbol{K}$. In this setting, we have that $\boldsymbol{K}\colon (\mathbb{R}^n, \|\cdot\|_2) \to (\mathbb{R}^{m\times k}, \|\cdot\|_F)$, where

$$\|\boldsymbol{A}\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{k} A_{ij}^2 \right)^{1/2}.$$

To compute all the parameters needed for the primal-dual algorithm, we need to compute the induced matrix norm of the operator K:

$$\begin{aligned} \|\boldsymbol{K}\|_{\text{op}}^2 &= \sup_{\|\boldsymbol{x}\|_2=1} \|\boldsymbol{K}\boldsymbol{x}\|_F^2 = \sup_{\|\boldsymbol{x}\|_2=1} \sum_{j=1}^{k} \|B^T e^{-\tau_j A^T} \boldsymbol{x}\|_2^2, \\ &\leq \sum_{j=1}^{k} \sup_{\|x_j\|_2=1} \|B^T e^{-\tau_j A^T} \boldsymbol{x}_j\|_2^2 = \sum_{j=1}^{k} \|B^T e^{\tau_j A^T}\|_{op}^2. \end{aligned} \tag{3.27}$$

Here is an alternative: We can view the operator norm $\boldsymbol{K}$ as an operator acting on the spaces $(\mathbb{R}^n, \|\cdot\|_2)$ and $(\mathbb{R}^{nk}, \|\cdot\|_2)$. In effect, we have that $\boldsymbol{K}$ is an $nk\times n$ matrix. Then the induced matrix norm is just the largest singular value of $\boldsymbol{K}$. We can implement this numerically (as a first step, at least).

(b) **Second choice: simple form of K and separable function F.** In order to decrease the number of iterations in the primal-dual algorithm, we make the matrix K simple. For instance, we choose

$$\boldsymbol{K} = \begin{bmatrix} I_n \\ I_n \\ \vdots \\ I_n \end{bmatrix} \text{ and } F([\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k]) = \sum_{i=1}^{k} c_i \sqrt{\langle \boldsymbol{B}^\top e^{-\tau_i \boldsymbol{A}^\top} \boldsymbol{y}_i, \boldsymbol{M}\boldsymbol{B}^\top e^{-\tau_i \boldsymbol{A}^\top} \boldsymbol{y}_i \rangle}. \quad (3.28)$$

Since F is still separable, the second step in the primal-dual method is also equivalent to solving k **projections onto different ellipsoids** (where the matrix associated with the ith ellipsoid is given by $e^{-\tau_i \boldsymbol{A}} \boldsymbol{B}\boldsymbol{M}\boldsymbol{B}^\top e^{-\tau_i \boldsymbol{A}^\top}$, i.e., $\boldsymbol{Q}_i$ on page 1).

(c) **Third choice: semi-complicated form of K and simple F.** This is a variant of the the primal-dual algorithm found in [58]. We define $\boldsymbol{K}$ and F by

$$\boldsymbol{K} = \begin{bmatrix} e^{-\tau_1 \boldsymbol{A}^\top} \\ e^{-\tau_2 \boldsymbol{A}^\top} \\ \vdots \\ e^{-\tau_k \boldsymbol{A}^\top} \end{bmatrix} \text{ and } F([\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k]) = \sum_{i=1}^{k} c_i \sqrt{\langle \boldsymbol{y}_i, \boldsymbol{B}\boldsymbol{M}\boldsymbol{B}^\top \boldsymbol{y}_i \rangle}. \quad (3.29)$$

What we have done here is push the matrix $\boldsymbol{B}$ into the sum of ellipsoids. The main justification of this approach reads as follows: In this setting, $\boldsymbol{K}\colon (\mathbb{R}^n, \|\cdot\|_2) \to (\mathbb{R}^{m \times k}, \|\cdot\|_F)$ where $\|\cdot\|_F$ denotes the Frobenius norm as before. The corresponding operator norm is

$$\|\boldsymbol{K}\|_{op}^2 = \sup_{\|\boldsymbol{x}\|_2 = 1} \|\boldsymbol{K}\boldsymbol{x}\|_F^2 = \sup_{\|\boldsymbol{x}\|_2 = 1} \sum_{j=1}^{k} \left\| e^{-\tau_j \boldsymbol{A}^\top} \boldsymbol{x} \right\|_2^2.$$

The operator norm here can be computed explicitly if one knows the eigenvector of $\boldsymbol{A}$ associated to the smallest eigenvalue of $\boldsymbol{A}$. Note that

$$\sup_{\|\boldsymbol{x}\|_2=1} \sum_{j=1}^{k} \left\| e^{-\tau_j \boldsymbol{A}^\top} \boldsymbol{x} \right\|_2^2 \leqslant \sup_{\|\boldsymbol{x}\|_2=1} \sum_{j=1}^{k} \left\| e^{-\tau_j \boldsymbol{A}^\top} \right\|_2^2 \|\boldsymbol{x}\|_2^2 = \sum_{j=1}^{k} \left\| e^{-\tau_j \boldsymbol{A}^\top} \right\|_2^2 = \sum_{j=1}^{k} e^{-2\tau_j \sigma}.$$

where σ denotes the smallest eigenvalue of $\boldsymbol{A}$. To see why the last equality holds, let $\boldsymbol{v} \in \mathbb{R}^n$ denotes an eigenvector of the matrix $\boldsymbol{A}$ with corresponding eigenvalue λ. Then, $\boldsymbol{v}$ is also an eigenvector of the exponential matrix $e^{\boldsymbol{A}}$, and its corresponding eigenvalue is e^{λ}. If $\boldsymbol{v}_\sigma$ denotes the eigenvector of $\boldsymbol{A}$ for which σ is the smallest eigenvalue in the spectrum of $\boldsymbol{A}$, then $\boldsymbol{v}_\sigma$ becomes the eigenvector of $e^{-\tau_j \boldsymbol{A}}$ with largest eigenvalue $e^{-\tau_j \sigma}$. The operator norm then becomes

$$\|\boldsymbol{K}\|_{op}^2 = \sum_{j=1}^{k} e^{-2\tau_j \sigma}.$$

(d) **Fourth choice: degenerate K and separable function F.** In fact, the "redundancy" of K and F gives us more freedom to choose K and F. For instance, when B or M is degenerate, we modify the second choice using the methods in Section 3.4.3. Then, the second step of the primal-dual method is similar to before, but we compute several projections, each of which corresponds to a **non-degenerate ellipsoid** (the ellipsoid in (3.26) is degenerate if M is singular, and the ellipsoid in (3.28) is degenerate if BMB^T is degenerate).

(e) **Fifth choice: simple form of K and non-separable F.** If we put K to be the identity matrix, then we have

$$\boldsymbol{K} = I_n \text{ and } F(\boldsymbol{y}) = \sum_{i=1}^{k} c_i \sqrt{\langle \boldsymbol{B}^\top e^{-\tau_i \boldsymbol{A}^\top} \boldsymbol{y}, \boldsymbol{M} \boldsymbol{B}^\top e^{-\tau_i \boldsymbol{A}^\top} \boldsymbol{y} \rangle} = \sum_{i=1}^{k} c_i \sqrt{\langle \boldsymbol{y}, \boldsymbol{Q}_i \boldsymbol{y} \rangle}. \tag{3.30}$$

However, when at least one of $Q_i = e^{-\tau_i \boldsymbol{A}} \boldsymbol{B} \boldsymbol{M} \boldsymbol{B}^\top e^{-\tau_i \boldsymbol{A}^\top}$ is degenerate, the

second step of primal-dual method cannot be solved directly because the algorithm 2 of chapter 2 cannot be applied to compute the projection onto the Minkowski sum of non-full ellipsoids. Therefore, we need to use a regularization method in Section 3.4.3, where the matrix Q_i is regularized by $\hat{Q}_{i,\epsilon} = Q_i + P_\epsilon$ (P_ϵ converges to zero if ϵ converges to zero and may depend on i). Then, we will have an outer iteration with respect to ϵ. Let ϵ_k be a positive decreasing sequence converging to zero. The k-th step in the algorithm is

(a) Compute initialization $y_{k+1,0}$ according to the optimizer y_k in the previous step.

(b) Run primal-dual algorithm ([21] and algorithm of chapter 2) to get the minimizer for $\sup_{p\in\mathbb{R}^n}\left\{\langle z,p\rangle - \sum_{i=1}^k c_i\sqrt{\left\langle p,\hat{\boldsymbol{Q}}_{i,\epsilon}p\right\rangle} - J_z^*(p)\right\}$ (see Section 3.4).

After N iterations, when ϵ_n is small, we use the minimizer y_N at the N-th step as an initialization for methods 1-4 described above.

3.4.2 Initialization Method (and A Change of Variables)

We can reformulate problem (3.14) equivalently as follows. Suppose we can write $\boldsymbol{V}^{-1} = e^{-T\boldsymbol{A}}LL^\top e^{-T\boldsymbol{A}^\top}$, e.g., using the Cholesky decomposition of the matrix $\boldsymbol{W}$ associated to the initial condition. Then problem (3.14) becomes equivalent to

$$\arg\min_{\tilde{\boldsymbol{x}}\in\mathbb{R}^n}\left\{\frac{1}{2}\left\|\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{q}}\right\|_2^2 + \sum_{i=1}^k\sqrt{\left\langle\tilde{\boldsymbol{x}},\tilde{\boldsymbol{Q}}_i\tilde{\boldsymbol{x}}\right\rangle}\right\}, \tag{3.31}$$

where

$$\tilde{\boldsymbol{q}} = \boldsymbol{L}^\top e^{-T\boldsymbol{A}^\top}\boldsymbol{q} + \boldsymbol{L}^{-1}e^{T\boldsymbol{A}}\boldsymbol{b} \quad\text{and}\quad \tilde{\boldsymbol{Q}}_i = c_i^2\boldsymbol{L}^{-1}e^{t_i\boldsymbol{A}}\boldsymbol{B}\boldsymbol{M}\boldsymbol{B}^\top e^{t_i\boldsymbol{A}^\top}(\boldsymbol{L}^{-1})^\top.$$

If $\tilde{\boldsymbol{x}}^*$ denotes the unique solution to this problem, then the projection on the ellipsoid is given by $\tilde{\boldsymbol{q}} - \tilde{\boldsymbol{x}}^*$ or, equivalently, $\tilde{\boldsymbol{q}} - L^\top e^{-TA^\top}\boldsymbol{x}^*$ where $\boldsymbol{x}^*$ denotes the solution of (3.14).

If $\tilde{\boldsymbol{Q}}_i$ is well-posed and SPD, then we can use the algorithm of chapter 2 to solve (3.31) directly. Otherwise, we can use the regularization methods (see Section 3.4.3) to approximate ill-posed or SPSD $\tilde{\boldsymbol{Q}}_i$ with well-posed or SPD matrices and then use the algorithm of chapter 2 to solve the resulting approximation of (3.31).

3.4.3 Regularization methods

Here we suppose that the matrices $\boldsymbol{Q}_i$ are either (a) symmetric positive semi-definite (SPSD), or (b) symmetric positive definite (SPD) and ill-conditioned. We want to solve (3.31) to use the algorithm of chapter 2 as an initialization algorithm. Note that we cannot directly use the algorithm of chapter 2 because the matrices $\boldsymbol{Q}_i$ are assumed to be degenerate. To solve this issue, we need to amend the matrices $\boldsymbol{Q}_i$ to make them strictly positive definite (and therefore well-conditioned). For this case, we use the primal-dual algorithm to solve the minimization problem (3.16), where we use the algorithm of chapter 2 to compute the prox of F^*.

(a) Assume $\boldsymbol{Q}_i$ is SPSD and that we order the eigenvalues as follows: $\lambda_1 \geq \lambda_2 \geq$

$\ldots \lambda_j \geq 0 = \lambda_{j+1} = \cdots = \lambda_n$. Then, $\boldsymbol{Q}_i$ has the following eigendecomposition:

$$\boldsymbol{Q}_i = U\Lambda U^\top = U \begin{bmatrix} \lambda_1 & & & & & \\ & \cdots & & & & \\ & & \lambda_j & & & \\ & & & 0 & & \\ & & & & \cdots & \\ & & & & & 0 \end{bmatrix} U^\top.$$

We have several options for choosing a SPD modification of $\boldsymbol{Q}_i$. Below are two options:

(i) Pick $\hat{\boldsymbol{Q}}_i$ to be:

$$\hat{\boldsymbol{Q}}_i = U\left(\Lambda + \epsilon \begin{bmatrix} 0 & & & & \\ & \cdots & & & \\ & & 0 & & \\ & & & 1 & \\ & & & & \cdots \\ & & & & & 1 \end{bmatrix}\right) U^\top = U \begin{bmatrix} \lambda_1 & & & & & \\ & \cdots & & & & \\ & & \lambda_j & & & \\ & & & \epsilon & & \\ & & & & \cdots & \\ & & & & & \epsilon \end{bmatrix} U^\top,$$

where $\epsilon > 0$. Then,

$$\boldsymbol{K}_i = U\Lambda_K U^\top, \quad \text{where } (\Lambda_K)_{xy} = \begin{cases} 1 & x = y \leq j, \\ 0 & \text{otherwise.} \end{cases}$$

(ii) Pick $\hat{\boldsymbol{Q}}_i$ to be:

$$\hat{\boldsymbol{Q}}_i = U(\Lambda + \epsilon \boldsymbol{I}_{n\times n})U^\top,$$

where $\epsilon > 0$. Then,

$$\boldsymbol{K}_i = U\Lambda_K U^\top, \quad \text{where } (\Lambda_K)_{xy} = \begin{cases} \sqrt{\dfrac{\lambda_x}{\lambda_x + \epsilon}} & x = y \leq j, \\ 0 & \text{otherwise.} \end{cases}$$

(iii) Pick $\hat{\boldsymbol{Q}}_i$ to be:

$$\hat{\boldsymbol{Q}}_i = Q_i + \epsilon \boldsymbol{I}_{n\times n}. \tag{3.32}$$

(b) If $\boldsymbol{Q}_i$ is ill-conditioned, then we can do something similar to above to get a better conditioned $\hat{\boldsymbol{Q}}_i$.

Now we are ready to our regularization approach.Replace Q_i by $\hat{Q}_i = Q_i + P$, where P is a symmetric positive definite matrix. For example, a simple case is to consider $P = \varepsilon I$. Then, we can define $K_i = UM_K U^T$, where M_K solves $M_K^T(\Lambda + U^T P U)M_K = \Lambda$, in order to satisfy $K_i^T \hat{Q}_i K_i = Q_i$. If $U^T P U$ is diagonal (as in (i) and (ii) above), then the equation can be solved directly (M_K is diagonal) as follows. If $U^T P U$ is a diagonal matrix Λ_P with diagonal elements $\lambda_{P,i}$, then M_K is a diagonal matrix whose i-th element is $\sqrt{\frac{\lambda_{Q,i}}{\lambda_{Q,i}+\lambda_{P,i}}}$, where $\lambda_{Q,i}$ is the i-th eigenvalue of Q.

3.5 Summary

We have proposed several numerical algorithms to solve optimal control problems with linear dynamics when the running cost is the characteristic function of an ellipsoid (potentially degenerate). Such degenerate cases are important for

application because it includes linear dynamics obtained by Newton mechanics. We leave as a future work the implementation of all these numerical methods and their performances.

Chapter Four

Numerical solver for certain multi-time Hamilton-Jacobi PDEs

4.1 Introduction

In this chapter, we provide an algorithm for computing the proximal point of a convex polyhedral function whose domain is a polyhedral set. This algorithm will be used to compute solutions to certain multi-time Hamilton-Jacobi PDEs and certain optimal control problems with linear dynamics.

Proximal point based optimization algorithms (e.g., ADMM and primal-dual method, among others) are often employed to solve optimal control problems and such algorithms require the computation of proximal points, e.g., numerical solutions to optimization problems of the form

$$\min_{x\in\mathbb{R}^n} f(x) + \frac{\rho}{2}\|x - c\|^2,$$

where $\rho > 0$, $c \in \mathbb{R}^n$, and $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function. However, in general, computing proximal points of arbitrary convex functions f, and especially of non-smooth convex functions f, is hard. These computations are further complicated by the fact that optimal control applications require fast computations that are tractable in high dimensions.

In this chapter, we present a gradient descent based algorithm for computing the proximal point of a certain class of nonsmooth, convex functions that is computationally tractable in high dimensions. In particular, we consider polyhedral functions f with domain that is a polyhedral set, which we define more precisely in Section 4.2.

We consider this class of functions for several reasons. Polyhedral functions could be used to approximate more general convex functions and polyhedral

domain allows us to consider constraints that can be represented by a finite collection of affine functions. More general convex sets could be approximated by polyhedral sets. The above two traits are significant for optimal control problems, which often involve optimizing general convex (and nonconvex) functions with constraints and examples of such functions are ℓ^1/ℓ^∞ balls/norms, n-simplex. To the authors' knowledge, no efficient solver for computing the proximal point of this class of functions exists, although efficient solvers for some specific examples of functions in this class do exist.

We show that our algorithm has comparable or faster computational runtime than the interior-point method, which is a classical method for solving constrained optimization problems. We also demonstrate that our algorithm is computationally tractable in high dimensions (e.g., $n = 16$), whereas the interior-point method suffers from memory constraints that prevent it from even computing a solution in some high-dimensional cases. Thus, our algorithm has the potential to be used as a building block in solvers for certain classes of optimal control problems.

The outline for the remainder of this chapter is as follows. In Section 4.2, we present our algorithm for computing the proximal point of polyhedral functions with polyhedral domain. Specifically, in Section 4.2.1, we describe some properties of our algorithm, and in Section 4.2.2, we detail how the descent direction for our algorithm can be computed. In Section 4.3, we compare the performance of our algorithm with that of the interior-point method. In Section 4.4, we demonstrate how our algorithm can be used for different applications. Specifically, in Section 4.4.1, we apply our algorithm to a certain class of multi-time Hamilton-Jacobi partial differential equations, and in Section 4.4.2, we apply our algorithm to a certain class of optimal control problems with linear dynamics. In Section 4.5, we provide some brief concluding remarks. Finally, in Appendix A.1, we review

some relevant definitions and theorems from convex analysis that are utilized in this chapter, while in Appendices A.2-A.3, we provide some technical lemmas used to prove some properties of our algorithm.

4.2 A gradient-descent-based algorithm

In this section, we provide an algorithm for computing the proximal point of a convex polyhedral function whose domain is a polyhedral set. In our numerical experiments, we observe that this algorithm terminates in finitely many iterations.

Let the set $C \subseteq \mathbb{R}^n$ be a convex (possibly unbounded) closed polyhedral set defined by

$$C = \bigcap_{i=1}^{m} \{x \in \mathbb{R}^n : \langle p_i, x \rangle \leq \alpha_i\} = \left\{ x \in \mathbb{R}^n : \sup_{i \in \{1,\dots,m\}} \{\langle p_i, x \rangle - \alpha_i\} \leq 0 \right\}, \tag{4.1}$$

for some $m \in \mathbb{N}$, $p_1, \dots, p_m \in \mathbb{R}^n$, and $\alpha_1, \dots, \alpha_m \in \mathbb{R}$. Let the function $\varphi : \mathbb{R}^n \to \mathbb{R}$ be a convex polyhedral function defined by

$$\varphi(x) = \sup_{j \in \{1,\dots,\tilde{m}\}} \{\langle q_j, x \rangle - \beta_j\}, \tag{4.2}$$

for some $\tilde{m} \in \mathbb{N}$, $q_1, \dots, q_{\tilde{m}} \in \mathbb{R}^n$, and $\beta_1, \dots, \beta_{\tilde{m}} \in \mathbb{R}$. Our goal is to solve the following minimization problem:

$$\min_{x \in \mathbb{R}^n} F(x) = \min_{x \in C} \left\{ \frac{1}{2} \|x - \gamma\|^2 + \varphi(x) \right\}, \tag{4.3}$$

where the function $F\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is defined by

$$F(x) := \frac{1}{2}\|x-\gamma\|^2 + \varphi(x) + I_C(x) \quad \forall\, x \in \mathbb{R}^n, \tag{4.4}$$

where I_C is the indicator function of the set C, which was defined in (4.1). In other words, our goal is to compute the proximal point of the function $\varphi + I_C$. We present an algorithm for computing the proximal point of $\varphi + I_C$ in Algorithm 1.

Algorithm 1: The proposed algorithm for solving the problem (4.3).

Input : The parameters $p_1, \ldots, p_m, q_1, \ldots, q_{\tilde{m}} \in \mathbb{R}^n$, $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_{\tilde{m}} \in \mathbb{R}$. The initialization $x_0 \in C$.

Output: The minimizer $\bar{x}$ of the problem (4.3).

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

2 Compute the descent direction $d^k \in \mathbb{R}^n$ by

$$d^k := -\Pi_{\{x^k-\gamma\}+\partial\varphi(x^k)+\partial I_C(x^k)}(0) = -\Pi_{\partial\varphi(x^k)+\partial I_C(x^k)}(-x^k+\gamma) - x^k + \gamma. \tag{4.5}$$

3 **if** $d^k = 0$ **then return** $\bar{x} = x^k$;

4 Compute the set $S^+(d^k) \subseteq \{1, \ldots, m\}$ defined by

$$S^+(d^k) := \{i \in \{1, \ldots, m\} \colon \langle p_i, d^k\rangle > 0\}, \tag{4.6}$$

and compute the set $\tilde{S}^+(d^k) \subseteq \{1, \ldots, \tilde{m}\}$ defined by

$$\tilde{S}^+(d^k) := \{j \in \{1, \ldots, \tilde{m}\} \colon \langle q_j, d^k\rangle > \varphi'(x^k; d^k)\}. \tag{4.7}$$

5 Compute $\Delta t^k \in \mathbb{R}$ as follows:

$$\Delta t^k := \min\left\{\Delta t_1^k, \Delta t_2^k, 1\right\}, \tag{4.8}$$

where $\Delta t_1^k \in \mathbb{R}$ and $\Delta t_2^k \in \mathbb{R}$ are defined by

$$\begin{aligned} \Delta t_1^k &:= \min\left\{\frac{\alpha_i - \langle p_i, x^k\rangle}{\langle p_i, d^k\rangle} \colon i \in S^+(d^k)\right\}, \\ \Delta t_2^k &:= \min\left\{\frac{\beta_j - \langle q_j, x^k\rangle + \varphi(x^k)}{\langle q_j, d^k\rangle - \varphi'(x^k; d^k)} \colon j \in \tilde{S}^+(d^k)\right\}. \end{aligned} \tag{4.9}$$

6 Update the iterate $x^{k+1} = x^k + \Delta t^k d^k$;

7 **end**

Now, we will explain the intuition behind Algorithm 1, which is a gradient-descent-based method, step-by-step. In each iteration, we first find a descent direction d^k. The idea of gradient descent is to follow the direction of the negative gradient. However, in our case, the objective function F in problem (4.3) is not differentiable, and we have to use the subgradient instead of the gradient. Thus, we follow the direction of the negative subgradient whose norm is the smallest among all subgradients (recall that by the theoretical guarantee in [5], the derivative of the gradient flow of a non-smooth convex function is the negative subgradient with the smallest norm). In other words, we choose the descent direction $d^k = -\Pi_{\partial F(x^k)}(0)$, where by straightforward computation, we have that

$$\partial F(x) = \{x - \gamma\} + \partial\varphi(x) + \partial I_C(x) \quad \forall x \in C. \tag{4.10}$$

Furthermore, the second equality in (4.5) holds since $\Pi_{K+\{x\}}(0) = \Pi_K(-x) + x$ holds for any point $x \in \mathbb{R}^n$ and for any convex set $K \subseteq \mathbb{R}^n$. Note that the subdifferential sets $\partial\varphi(x)$ and $\partial I_C(x)$ have explicit formulas, and hence, we can rewrite (4.5) as the explicit solution to a minimization problem as follows. Recall that for any $x \in C$, we have that

$$\partial I_C(x) = \text{cone}\,\{p_i \colon i \in S(x)\} = \left\{ \sum_{i \in S(x)} a_i p_i \colon a_i \geq 0 \forall i \in S(x) \right\}, \tag{4.11}$$

where the set $S(x) \subseteq \{1, \ldots, m\}$ is defined by

$$S(x) := \{i \in \{1, \ldots, m\} \colon \langle p_i, x \rangle = \alpha_i\}. \tag{4.12}$$

Moreover, for any $x \in \mathbb{R}^n$, we also have that

$$\partial\varphi(x) = \operatorname{conv}\{q_j \colon j \in \tilde{S}(x)\} = \left\{ \sum_{j\in\tilde{S}(x)} b_j q_j \colon b_j \geq 0 \forall j \in \tilde{S}(x), \sum_{j\in\tilde{S}(x)} b_j = 1 \right\}, \tag{4.13}$$

where the set $\tilde{S}(x) \subseteq \{1, \ldots, \tilde{m}\}$ is defined by

$$\tilde{S}(x) := \{j \in \{1, \ldots, \tilde{m}\} \colon \langle q_j, x\rangle - \beta_j = \varphi(x)\}. \tag{4.14}$$

Therefore, the direction d^k is given by $d^k = -\bar{w} - \bar{v} - x^k + \gamma$, where $(\bar{w}, \bar{v})$ is the minimizer of the following optimization problem:

$$\begin{aligned} \min_{\substack{w,v\in\mathbb{R}^n \\ a_i,b_j\in\mathbb{R}}} \Bigg\{ \left\| -x^k + \gamma - w - v \right\|^2 : w = \sum_{i\in S(x^k)} a_i p_i, v = \sum_{j\in\tilde{S}(x^k)} b_j q_j, \\ a_i \geq 0 \forall i \in S(x^k), b_j \geq 0 \forall j \in \tilde{S}(x^k), \sum_{j\in\tilde{S}(x^k)} b_j = 1 \Bigg\}, \end{aligned} \tag{4.15}$$

which can be solved using the numerical method in Section 4.2.2.

In the next step of Algorithm 1, if the direction d^k is not zero, we compute the maximal time period Δt^k for which the direction d^k $(= -\Pi_{\partial F(x^k)}(0))$ is equivalent to $-\Pi_{\partial F(x^k+td^k)}(0)$ for all $t \in [0, \Delta t^k)$. Therefore, we choose Δt^k to be the minimal value in $\{\Delta t_1^k, \Delta t_2^k, 1\}$, where Δt_1^k is the time until we arrive at a new face of C whose index is given by the minimizer in the first equation in (4.9) and Δt_2^k is the time until we arrive at a new face of epi φ whose index is the minimizer in the second equation in (4.9). Note that the sets S^+ and $\tilde{S}^+$ may be empty. If S^+ or $\tilde{S}^+$ is empty, then the corresponding equation in (4.9) becomes the minimum over an empty set, and we adopt the convention that the minimum over an empty set is $+\infty$. We set a maximal threshold of 1.0 for Δt^k to prevent the trajectory (i.e., the straight line segment connecting x^k and x^{k+1}) from passing the minimizer as it moves to a new

face of C or epi φ. As a simple illustration of the significance of this threshold, consider the example where φ is constant and the vector γ is in the interior of the set C. If we set $\Delta t^k = \min\{\Delta t_1^k, \Delta t_2^k\}$, then the first iterate x^1 will be on the intersection of the boundary of C and the ray $\{x^0 + s(\gamma - x^0) \colon s \geq 0\}$. However, if we set $\Delta t^k = \min\{\Delta t_1^k, \Delta t_2^k, 1\}$, then the next iterate x^1 is γ, which is the minimizer; in other words, the trajectory terminates at the minimizer before reaching a new face of C. Note that, unlike in linear programming problems, the quadratic term in the objective function allows the minimizer not to be in the set of the extreme points of C or the x-coordinate of the extreme points of epi φ. We also note that the step size Δt^k is almost always less than 1.0 (in Lemma A.2.1, we prove that $\Delta t^k \in (0, 1]$), and when the step size Δt^k equals 1.0, the algorithm will terminate in the next step (see Lemma A.2.3).

In Section 4.2.1, we provide some theoretical guarantees for Algorithm 1 and an interpretation of Algorithm 1 in terms of the gradient flow of the function F. Then, in Section 4.2.2, we provide more details on computing the descent direction d^k in (4.5) numerically.

4.2.1 Properties of the algorithm

In this section, we show that Algorithm 1 is related to the gradient descent algorithm for the function F defined by (4.4). Let $\Delta t^k \in \mathbb{R}$ and $x^k, d^k \in \mathbb{R}^n$ be the scalars and vectors defined in Algorithm 1. Define the trajectory $x \colon [0, +\infty) \to \mathbb{R}^n$ by

$$x(s) := x^k + (1 - \exp(-s + s^k))d^k \quad \forall s \in [s^k, s^{k+1}), \tag{4.16}$$

where each s^k is defined by

$$s^k := \sum_{i=1}^{k-1} (-\log(1 - \Delta t^i)), \tag{4.17}$$

where we use the convention $\log 0 = -\infty$. If Algorithm 1 terminates after a finite number of iterations K, then we define

$$x(s) := x^K \quad \forall s \in [s^K, +\infty). \tag{4.18}$$

Note that if Algorithm 1 does not terminate in finite iterations, then by Lemma A.2.3 we have that $\Delta t^i \in (0, 1)$ holds for each $i \in \mathbb{N}$, and hence $\{s^k\}_{k=1}^{\infty}$ is a strictly increasing sequence in $\mathbb{R}$. In other words, the trajectory $x(\cdot)$ defined in (4.16) consists of infinitely many line segments, where the k-th segment corresponds to the k-th iteration in Algorithm 1. Similarly, by Lemma A.2.3, if Algorithm 1 terminates after a finite number of iterations K, then the trajectory $x(\cdot)$ defined in (4.16) consists of $K - 1$ line segments, where the k-th segment corresponds to the k-th iteration in Algorithm 1.

In the following proposition, we show that the trajectory $x(\cdot)$ as defined by (4.16) is the gradient flow of the function F. In this sense, each step in Algorithm 1 corresponds to a point of the gradient flow $x(\cdot)$ where the direction $\frac{x'(s)}{\|x(s)\|}$ changes. Therefore, our proposed algorithm can be seen as a gradient descent method with certain step sizes.

Proposition 4.2.1. *Let x_0 be a point in C. Let $x\colon [0, +\infty) \to \mathbb{R}^n$ be the trajectory defined in* (4.16)*. Then, the function $x(\cdot) \in C(0, +\infty; \mathbb{R}^n)$ is the unique solution to the following differential inclusion:*

$$\begin{cases} x'(s) \in -\partial F(x(s)) \quad s > 0, \\ x(0) = x_0. \end{cases} \tag{4.19}$$

Proof. Since ∂F is a maximal monotone set-valued function, by [5, Chap 3.2, Thm 1] the problem (4.19) has a unique solution, which is also the unique solution for almost all $s \geq 0$ of the following ODE:

$$\begin{cases} x'(s) = -\Pi_{\partial F(x(s))}(0) \quad s > 0, \\ x(0) = x_0. \end{cases} \tag{4.20}$$

Thus, it suffices to prove that the function $x(\cdot)$ defined in (4.16) satisfies the ODE (4.20) almost everywhere. The initial condition is satisfied by the definition of $x(\cdot)$. The function $x(\cdot)$ is continuous and piecewise smooth, and its derivative in each time period reads

$$x'(s) = \exp(-s + s^k)d^k \text{ if } s \in (s^k, s^{k+1}). \tag{4.21}$$

If Algorithm 1 terminates in the $(K+1)$-th iteration, then we have that $x'(s) = 0$ holds for $s > s^K$. For any $s \in (s^k, s^{k+1})$, by straightforward computation using the definition of s^k, we have that

$$1 - \exp(-s + s^k) \in (0, 1 - \exp(-s^{k+1} + s^k)) = (0, \Delta t^k). \tag{4.22}$$

Hence, Lemma A.2.4 holds for $x(s)$, which, together with (4.10), implies

$$-\Pi_{\partial F(x(s))}(0) = d^k + x^k - x(s) = d^k + x^k - (x^k + (1 - \exp(-s + s^k))d^k) = \exp(-s + s^k)d^k = x'(s).$$

Moreover, if Algorithm 1 terminates in the $(K+1)$-th iteration, then we have $d^K = 0$ and

$$0 = -\Pi_{\{x^K - \gamma'\} + \partial\varphi(x^K) + \partial I_C(x^K)}(0) = -\Pi_{\partial F(x^K)}(0).$$

Thus, for any $s \geq s^K$, we have

$$x'(s) = 0 = -\Pi_{\partial F(x^K)}(0) = -\Pi_{\partial F(x(s))}(0).$$

Therefore, the trajectory $x(\cdot)$ defined in (4.16) satisfies (4.20) almost everywhere, and hence, it is the unique solution to (4.19). ■

Now, we present our main result in the following proposition, which shows that Algorithm 1 converges to the unique minimizer of the problem (4.3) under certain assumptions.

Proposition 4.2.2. *Let the initial point x_0 be a point in C. Then, Algorithm 1 is well-defined. Assume there exists $c \in (0, 1)$ such that either $\Delta t^k \geq c$ for any k. Then, Algorithm 1 converges to the unique minimizer of the problem* (4.3).

Proof. By Lemmas A.2.1 (c) and A.2.2 and induction on k, we have $\Delta t^k \in (0, 1]$ and $x^k \in C$. Therefore, Algorithm 1 is well-defined. By straightforward checking, the objective function F is convex, lower semi-continuous, 1-coercive, and strictly convex in its domain. Therefore the optimization problem (4.3) has a unique minimizer, denoted by $\bar{x}$. By [5, Sec. 3.4, Theorem 2], the solution to the differential inclusion (4.19) converges to the unique minimizer $\bar{x}$.

If there exists K such that $\Delta t^K = 1$, then by Lemma A.2.3, Algorithm 1 terminates at the $(K + 1)$-th iteration. By definition of x and s^{K+1} in (4.16) and (4.17), we have $s^{K+1} = +\infty$ and $\lim_{s \to +\infty} x(s) = x^K + d^K = x^{K+1}$. Hence, the solution to the differential inclusion (4.19) converges to x^{K+1}. Therefore, x^{K+1} is the unique minimizer to the problem (4.3).

If Algorithm 1 terminates in K iterations but there does not exist K such that

$\Delta t^K = 1$, then by (4.18), we have $\lim_{s\to+\infty} x(s) = x^K$, and hence the output x^K of Algorithm 1 is the unique minimizer to the problem (4.3).

Now, it remains to prove the results when Algorithm 1 does not terminate in finite iterations. By assumption we have $\Delta t^k \geq c$ for any $k \in \mathbb{N}$. By (4.16) and (4.17), we get $x(s^k) = x^k$ for any $k \in \mathbb{N}$ and $s^k \geq \sum_{i=1}^{k-1}(-\log(1-c)) = -(k-1)\log(1-c)$, which converges to $+\infty$. Therefore, we obtain $\lim_{k\to\infty} x^k = \lim_{k\to\infty} x(s^k) = \lim_{s\to+\infty} x(s) = \bar{x}$. In other words, Algorithm 1 converges to the unique minimizer $\bar{x}$ of the optimization problem (4.3). ■

Remark 4.2.3. In our numerical experiments, we observe that Algorithm 1 terminates in finite iterations. If φ is an affine function, we provide a proof for this finite termination property (see Proposition A.2.8).

4.2.2 Computing the descent direction

To compute the descent direction d^k as defined in (4.5), we need to solve the optimization problem in (4.15).

In this section, we only consider a numerical algorithm for solving (4.15) for a fixed k. Thus, to simplify notation, we denote $Y = -x^k + \gamma$, which is a fixed vector in $\mathbb{R}^n$. We also consider the vectors $\{p_i : i \in S(x^k)\}$ and $\{q_j : j \in \tilde{S}(x^k)\}$. Using an abuse of notation and rearranging the indices, we denote these two sets by $\{p_i : i = 1, \ldots, I\}$ and $\{q_j : j = 1, \ldots, J\}$, respectively, to avoid the complicated index set notation. Note that the vectors p_i, q_j in this section are not the same as the vectors p_i, q_j in (4.1) and (4.2); rather, the vectors p_i, q_j in this section are a subset of the vectors p_i, q_j in (4.1) and (4.2).

With this simplified notation, the optimization problem (4.15) becomes

$$\min_{\substack{w,v\in\mathbb{R}^n \\ a_i,b_j\in\mathbb{R}}} \left\{ \|Y - w - v\|^2 : w = \sum_{i=1}^{I} a_i p_i, v = \sum_{j=1}^{J} b_j q_j, a_i \geq 0 \forall i,\ b_j \geq 0 \forall j, \sum_{j=1}^{J} b_j = 1 \right\}. \tag{4.23}$$

To solve (4.23), we consider two cases, depending on whether the set $\{q_j : j = 1, \ldots, J\}$ is linearly independent or not. We discuss the two cases and corresponding numerical algorithms in the following two sections.

4.2.2.1 The first case

In the first case, we assume $q_1, \ldots, q_J$ are linearly independent, and we apply Algorithm 2, which solves (4.23) using a splitting method by iteratively solving for the terms w and v in (4.23). Specifically, the first iterate $w^{\ell+1}$ (4.30) is equivalent to the projection of $Y - v^\ell$ onto the polyhedral cone generated by $\{p_1, \ldots, p_I\}$. We compute (4.30) using any algorithm for computing the projection onto a polyhedral cone. For the iterate corresponding to v in (4.23), we apply a change of variable using an invertible matrix Q whose first J column vectors are the vectors $q_1, \ldots, q_J$ (such a matrix Q exists since the vectors $q_1, \ldots, q_J$ are linearly independent). With some linear algebraic computation, the constraints $v = \sum_{j=1}^{J} b_j q_j$, $b_j \geq 0 \forall j$, and $\sum_{j=1}^{J} b_j = 1$ are equivalent to

$$v \in \text{cone}\,\{q_1, \ldots, q_J\}, \quad Q^{-1}v = (b_1, \ldots, b_J, \mathbf{0}), \quad 1 = \sum_{j=1}^{J} b_j = \sum_{j=1}^{n} (Q^{-1}v)_j = \langle (Q^{-1})^T \mathbf{1}, v \rangle. \tag{4.24}$$

Thus, we can remove the variables $b_1, \ldots, b_J$ and reformulate the update step for v as (4.31). Note that (4.31) is an optimization problem of the following form:

$$\min_{v \in \mathbb{R}^n} \{\|z - v\|^2 \colon v \in V, \langle c, v \rangle = 1\}, \tag{4.25}$$

where V is a polyhedral convex cone and $c \in \mathbb{R}^n$. In (4.31), we have $V = \text{cone}\,\{q_1, \ldots, q_J\}$, $z = Y - w^{\ell+1}$, and $c = (Q^{-1})^T \mathbf{1}$. To solve (4.31), we propose an efficient algorithm (that terminates in finitely many steps) for solving the general problem (4.25), which is summarized in Algorithm 3.

Next, we briefly describe the derivation of Algorithm 3. By Lemma A.3.1, solving (4.25) is equivalent to finding the root of the function $g \colon \mathbb{R} \to \mathbb{R}$ defined by

$$g(\lambda) := \langle c, \Pi_V(z - \lambda c) \rangle - 1, \tag{4.26}$$

where $\lambda \in \mathbb{R}$ is the Lagrange multiplier. Then, the minimizer $\bar{v}$ in (4.25) is given by $\Pi_V(z - \bar{\lambda} c)$, where $\bar{\lambda}$ is the root of g. Since g is a one-dimensional, piecewise affine, decreasing function, by Lemma A.3.2, the root of g can be efficiently computed using Newton's method. The value of g in (4.26) for any λ can be computed using any algorithm which can compute the projection on a polyhedral convex cone. By Lemma A.3.2, the directional derivatives of g are given by

$$g'(\lambda; 1) = -\|c\|^2 - \langle c, \Pi_{W(\lambda)}(-c) \rangle, \quad g'(\lambda; -1) = \|c\|^2 - \langle c, \Pi_{W(\lambda)}(c) \rangle, \tag{4.27}$$

the set $W(\lambda)$ is a convex polyhedral cone defined by (see (A.44))

$$W(\lambda) = \{y \in \mathbb{R}^n \colon \langle \Pi_V(z - \lambda c), y \rangle = 0, \langle v_j, y \rangle \leq 0 \; \forall j \in \mathcal{J}\} \tag{4.28}$$

and the index set $\mathcal{J}$ is defined by

$$\mathcal{J} = \{j \in \{1, \ldots, N\} \colon \langle v_j, \Pi_{V^\circ}(z - \lambda c)\rangle = 0\}. \tag{4.29}$$

Note that the polyhedral convex cone $W(\lambda)$ can be numerically represented since the vector $\Pi_V(z - \lambda c)$ can be numerically computed using any algorithm which can compute the projection on a polyhedral convex cone. Therefore, the directional derivatives of g can also be computed using any algorithm which can compute the projection on a polyhedral convex cone.

We summarize our proposed Newton's method for solving (4.25) in Algorithm 3. First, we compute the value of g by projecting $z - \lambda^k c$ onto the polyhedral cone V. Since g is a decreasing function, if the value of g is negative, we decrease the value of λ^k to get $\tilde{\lambda}$ using (4.34), where we use the directional derivative $g'(\lambda^k; -1)$ to replace the gradient term in Newton's method; if the value of g is positive, we increase the value of λ^k to get $\tilde{\lambda}$ using (4.35), where we use the directional derivative $g'(\lambda^k; 1)$ to replace the gradient term in Newton's method. To improve the efficiency and robustness of Newton's method, we combine it with a bisection search. Specifically, if the updated value of $\tilde{\lambda}$ falls outside the possible range $[\lambda_l, \lambda_u]$, we throw away the Newton update and use the midpoint $\frac{\lambda_l + \lambda_u}{2}$ instead.

Note that one special case happens when the directional derivative is zero, which in turn will cause the next Newton step to be ill-defined. By Lemma A.3.3, this special case only happens when the value of $g(\lambda^k)$ is negative. To avoid this case, if $g'(\lambda^k; -1) = 0$ and $g(\lambda^k) < 0$, we increase the value of $\tilde{\lambda}$ until the projection $\Pi_{V^\circ}(x - \lambda^k c)$ reaches a new face of V° instead of using a Newton step. When the projection $\Pi_{V^\circ}(x - \lambda^k c)$ reaches a new face of V°, there exists v_j such that $\langle v_j, \Pi_{V^\circ}(x - \tilde{\lambda} c)\rangle = 0$ but $\langle v_j, \Pi_{V^\circ}(x - \lambda^k c)\rangle \neq 0$. After some computation, we get the

update scheme (4.33) (the computational details for the reason behind this update scheme is similar as in Lemma A.2.2 (a)). Since g is a piecewise affine decreasing function, Algorithm 3 terminates in finitely many iterations.

Algorithm 2: The proposed algorithm for solving the problem (4.23), where $q_1, \ldots, q_J$ are linearly independent.

Input : The parameters $Y, p_1, \ldots, p_I, q_1, \ldots, q_J \in \mathbb{R}^n$, the initializations $w^0, v^0 \in \mathbb{R}^n$, and the stopping criteria tolerance $\epsilon > 0$.

Output: The minimizer (w^N, v^N) of (4.23).

Compute the invertible matrix Q with n rows and n columns, such that its first J column vectors are the vectors $q_1, \ldots, q_J$.

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

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

$$\begin{aligned} w^{\ell+1} &= \underset{w \in \mathbb{R}^n}{\arg\min} \left\{ \left\| Y - v^\ell - w \right\|^2 : w = \sum_{i=1}^{I} a_i p_i, a_i \geq 0 \, \forall i \right\}, \\ &= \Pi_{\text{cone } \{p_1, \ldots, p_I\}}(Y - v^\ell). \end{aligned} \tag{4.30}$$

By convention, if $I = 0$, then we define $w^{\ell+1} = 0$.

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

$$\begin{aligned} v^{\ell+1} &= \underset{v \in \mathbb{R}^n}{\arg\min} \left\{ \left\| Y - w^{\ell+1} - v \right\|^2 : v = \sum_{j=1}^{J} b_j q_j, b_j \geq 0 \, \forall j, \sum_{j=1}^{J} b_j = 1 \right\}, \\ &= \underset{v \in \mathbb{R}^n}{\arg\min} \left\{ \left\| Y - w^{\ell+1} - v \right\|^2 : v \in \text{cone } \{q_1, \ldots, q_J\}, \langle (Q^{-1})^T \mathbf{1}, v \rangle = 1 \right\}. \end{aligned} \tag{4.31}$$

By convention, if $J = 0$, then we define $v^{\ell+1} = 0$. Otherwise, compute $v^{\ell+1}$ by solving (4.31) using Algorithm 3, whose inputs are $N = J$, $z = Y - w^{\ell+1}$, $c = (Q^{-1})^T \mathbf{1}$, and $v_j = q_j$ for $j = 1, \ldots, J$.

if $\|w^\ell - w^{\ell+1}\|^2 \leq \epsilon$ *and* $\|v^\ell - v^{\ell+1}\|^2 \leq \epsilon$ **then return** $(w^N, v^N) = (w^{\ell+1}, v^{\ell+1})$;

end

4.2.2.2 The second case

In the second case, we assume the vectors $q_1, \ldots, q_J$ are linearly dependent, and we solve (4.23) using Algorithm 4. Since the vectors $q_1, \ldots, q_J$ are dependent, we

Algorithm 3: The proposed algorithm for solving the problem (4.25).

Input : The parameters $v_1, \ldots, v_N, c, z \in \mathbb{R}^n$. The initial Lagrange multiplier $\lambda^0 \in \mathbb{R}$.

Output: The minimizer $\bar{v}$ of the problem (4.25).

Set the lower bound $\lambda_l = -\infty$ and the upper bound $\lambda_u = +\infty$;

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

Compute the projection $\bar{z}^k$ by

$$\bar{z}^k = \Pi_V(z - \lambda^k c) = \Pi_{\text{cone }\{v_1,\ldots,v_N\}}(z - \lambda^k c), \tag{4.32}$$

and compute the function value $g(\lambda^k) = \langle c, \bar{z}^k \rangle - 1$;

if $g(\lambda^k) = 0$ **then return** $\bar{v} = \bar{z}^k$;

else if $g(\lambda^k) < 0$ **then**

Update the upper bound by $\lambda_u = \min\{\lambda_u, \lambda^k\}$. Compute $g'(\lambda^k; -1)$ using (4.27);

if $g'(\lambda^k; -1) = 0$ **then**

Compute the updated Lagrange multiplier by

$$\tilde{\lambda} = \lambda^k - \min\left\{ -\frac{\langle v_j, z - \lambda^k c - \bar{z}^k \rangle}{\langle v_j, \Pi_{W(\lambda^k)}(c) \rangle} : j \in \{1, \ldots, J\} \right. \\ \left. \text{s.t. } \langle v_j, \Pi_{W(\lambda^k)}(c) \rangle > 0 \right\}, \tag{4.33}$$

where $W(\lambda^k)$ is the set defined in (4.28).

else

Compute the updated Lagrange multiplier by

$$\tilde{\lambda} = \lambda^k - \left| \frac{g(\lambda^k)}{g'(\lambda^k; -1)} \right|. \tag{4.34}$$

end

else

Update the lower bound by $\lambda_l = \max\{\lambda_l, \lambda^k\}$, and compute $g'(\lambda^k; 1)$ using (4.27). Compute the updated Lagrange multiplier by

$$\tilde{\lambda} = \lambda^k + \left| \frac{g(\lambda^k)}{g'(\lambda^k; 1)} \right|. \tag{4.35}$$

end

If $\tilde{\lambda} \in [\lambda_l, \lambda_u]$, set $\lambda^{k+1} = \tilde{\lambda}$; Otherwise, set $\lambda^{k+1} = \frac{1}{2}(\lambda_l + \lambda_u)$;

end

cannot apply the change of variable Q in the first case to eliminate the variables $b_1, \ldots, b_J$ in the subproblem for v. Instead of splitting the original problem (4.23) into two subproblems, in this case, we split it into several subproblems, including one subproblem involving w and multiple subproblems involving v. More specifically, we divide the set $\{q_1, \ldots, q_J\}$ into several subsets $\{q_j \colon j \in \mathcal{J}_m\}$, $m = 1, \ldots, M$, where no two subsets intersect, the union of all subsets recovers the original set $\{q_1, \ldots, q_J\}$, and each subset $\{q_j \colon j \in \mathcal{J}_m\}$ is linearly independent. Then, we apply a change of variable (analogous to the change of variable Q in the first case) to each subset as follows.

Let Q_m be the change of variable matrix corresponding to the m-th subset, i.e., Q_m is an invertible matrix with n rows and n columns, such that its first $\mathcal{J}_m$ column vectors are $\{q_j \colon j \in \mathcal{J}_m\}$ (where such a matrix Q_m exists because the set $\{q_j \colon j \in \mathcal{J}_m\}$ is linearly independent). Let $v_m = \sum_{j \in \mathcal{J}_m} b_j q_j$. Then, using similar computations as in the first case, the constraints $v = \sum_{j=1}^{J} b_j q_j$, $b_j \geq 0 \forall j$, and $\sum_{j=1}^{J} b_j = 1$ are equivalent to

$$v = \sum_{m=1}^{M} v_m, \quad v_m \in \text{cone}\,\{q_j \colon j \in \mathcal{J}_m\} \forall m, \quad 1 = \sum_{m=1}^{M} \langle (Q_m^{-1})^T \mathbf{1}, v_m \rangle. \tag{4.36}$$

Thus, the optimization problem (4.23) is equivalent to

$$\begin{aligned} \min_{w, v_0, \ldots, v_M \in \mathbb{R}^n} \Bigg\{ \|Y - v_0\|^2 \colon v_0 = w + \sum_{m=1}^{M} v_m, w \in W, v_m \in C_m \\ \forall m \in \{1, \ldots, M\}, \sum_{m=1}^{M} \langle (Q_m^{-1})^T \mathbf{1}, v_m \rangle = 1 \Bigg\}, \end{aligned} \tag{4.37}$$

where we introduce a new variable $v_0 = w + \sum_{m=1}^{M} v_m$ and we set W to be the polyhedral cone generated by $\{p_1, \ldots, p_I\}$ and C_m to be the polyhedral cone generated

by $\{q_j : j \in \mathcal{J}_m\}$, i.e., we define W and C_m by

$$W = \text{cone}\,\{p_1, \ldots, p_I\}, \quad C_m = \text{cone}\,\{q_j : j \in \mathcal{J}_m\} \quad \forall m = 1, \ldots, M. \tag{4.38}$$

Now, we apply an ADMM scheme to solve (4.37). ADMM is an iterative algorithm, where each iteration consists of two steps: first, we update the primal variables, and second, we update the dual variables. In our case, these two steps are detailed as follows.

In the first step, we fix the dual variables $y^\ell \in \mathbb{R}^n$ and $\lambda^\ell \in \mathbb{R}$ and update the primal variables $w, v_0, \ldots, v_M$ by solving the following minimization problem:

$$\min_{w, v_0, \ldots, v_M \in \mathbb{R}^n} \mathcal{L}(w, v_0, \ldots, v_M, y^\ell, \lambda^\ell), \tag{4.39}$$

where the function $\mathcal{L}$ is defined by

$$\begin{aligned}\mathcal{L}(w, v_0, \ldots, v_M, y, \lambda) =& \frac{1}{2}\|Y - v_0\|^2 + \sum_{m=1}^{M} I_{C_m}(v_m) + I_W(w) \\ &+ \frac{\rho}{2}\left\| y + v_0 - w - \sum_{m=1}^{M} v_m \right\|^2 + \frac{\rho}{2}\left(\lambda + \sum_{m=1}^{M} \langle (Q_m^{-1})^T \mathbf{1}, v_m \rangle - 1 \right)^2,\end{aligned} \tag{4.40}$$

where $\rho > 0$ is a positive parameter and $y \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$ are the dual variables corresponding to the constraints $v_0 = w + \sum_{m=1}^{M} v_m$ and $\sum_{m=1}^{M} \langle (Q_m^{-1})^T \mathbf{1}, v_m \rangle = 1$, respectively. To solve (4.39), we use a splitting method to update the variables $v_0, w, v_1, \ldots, v_M$ one-by-one. Since the objective function $\mathcal{L}$ is quadratic in v_0, the minimizer of $v_0 \mapsto \mathcal{L}(w^\ell, v_0, v_1^\ell \ldots, v_M^\ell, y^\ell, \lambda^\ell)$ has an analytical form given explicitly by (4.46). For fixed $v_0 = v_0^{\ell+1}, v_1 = v_1^\ell, \ldots, v_M = v_M^\ell$, the minimizer of $\mathcal{L}$ with respect to w is the projection of $y^\ell + v_0^{\ell+1} - \sum_{m=1}^{M} v_m^\ell$ onto the set W, which can be computed

using any algorithm for computing the projection onto a polyhedral cone. Finally, for each $m = 1, \ldots, M$ and fixed $v_0 = v_0^{\ell+1}$, $w = w^{\ell+1}$, $v_j = v_j^{\ell+1}$ for $j < m$, and $v_j = v_j^{\ell}$ for $j > m$, computing the minimizer of $\mathcal{L}$ with respect to v_m requires us to solve the following optimization problem:

$$\min_{v_m \in C_m} \left\{ \left\| y^\ell + v_0^{\ell+1} - w^{\ell+1} - \sum_{i=1}^{m-1} v_i^{\ell+1} - \sum_{i=m+1}^{M} v_i^\ell - v_m \right\|^2 + \left(\lambda^\ell + \sum_{i=1}^{m-1} \langle (Q_i^{-1})^T \mathbf{1}, v_i^{\ell+1} \rangle + \sum_{i=m+1}^{M} \langle (Q_i^{-1})^T \mathbf{1}, v_i^\ell \rangle + \langle (Q_m^{-1})^T \mathbf{1}, v_m \rangle - 1 \right)^2 \right\}. \tag{4.41}$$

We solve this optimization problem as follows. Define $c, u \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$ by

$$\begin{aligned} c &= (Q_m^{-1})^T \mathbf{1}, \\ u &= y^\ell + v_0^{\ell+1} - w^{\ell+1} - \sum_{i<m} v_i^{\ell+1} - \sum_{i>m} v_i^\ell, \\ \alpha &= \lambda^\ell + \sum_{i<m} \langle (Q_i^{-1})^T \mathbf{1}, v_i^{\ell+1} \rangle + \sum_{i>m} \langle (Q_i^{-1})^T \mathbf{1}, v_i^\ell \rangle - 1. \end{aligned} \tag{4.42}$$

Then, the problem (4.41) is equivalent to

$$\begin{aligned} &\min_{v_m \in C_m} \left\{ \|u - v_m\|^2 + (\alpha + \langle c, v_m \rangle)^2 \right\} \\ = &\min_{v_m \in C_m} \left\{ (v_m + (I + cc^T)^{-1}(\alpha c - u))^T (I + cc^T)(v_m + (I + cc^T)^{-1}(\alpha c - u)) \right\}. \end{aligned} \tag{4.43}$$

We apply the change of variable $\tilde{v} = Ev_m$, where the matrix E is defined by

$$E = \sqrt{I + cc^T}, \tag{4.44}$$

to get that the minimization problem in (4.43) is equivalent to solving the following problem:

$$\min_{\tilde{v} \in \mathbb{R}^n} \left\{ \left\| \tilde{v} - E^{-1}(u - \alpha c) \right\|^2 : E^{-1}\tilde{v} \in C_m \right\};$$

that is, it is equivalent to computing the projection of $E^{-1}(u-\alpha c)$ onto the polyhedral cone $\tilde{C}_m$ defined by

$$\tilde{C}_m = EC_m = \text{cone}\left\{Eq_j \colon j \in \mathcal{J}_m\right\}. \tag{4.45}$$

Thus, the minimizer $\tilde{v}_m$ can be computed using any algorithm for computing the projection onto a polyhedral cone, and we recover $v_m^{\ell+1}$ using (4.48).

Finally, after updating all of the primal variables, the second step of ADMM updates the dual variables y and λ by (4.49).

4.3 Comparison to existing optimization methods

In this section, we compare the performance of Algorithm 1 with that of the interior-point method [74], which is a classical approach for solving constrained optimization problems. Specifically, we compare a simple MATLAB implementation of Algorithm 1 with the interior-point method, as implemented by MATLAB's built-in function quadprog. We note that the original optimization problem (4.3) cannot be solved directly using the interior-point method. Thus, to use the interior-point method, we propose the following reformulation of (4.3):

$$\min_{x\in\mathbb{R}^n} F(x) = \min_{\begin{pmatrix} x \\ s \end{pmatrix} \in C'} \left\{\frac{1}{2}\|x-\gamma\|^2 + s\right\} = \min_{\begin{pmatrix} x \\ s \end{pmatrix} \in C'} \left\{\frac{1}{2}\begin{pmatrix} x \\ s \end{pmatrix}^T \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} x \\ s \end{pmatrix} + \left\langle \begin{pmatrix} x \\ s \end{pmatrix}, \begin{pmatrix} -\gamma \\ 1 \end{pmatrix} \right\rangle\right\} + \frac{1}{2}\|\gamma\|^2, \tag{4.50}$$

Algorithm 4: The proposed algorithm for solving the problem (4.23), where $q_1, \dots, q_J$ are linearly dependent.

Input : The parameters $Y, p_1, \dots, p_I, q_1, \dots, q_J \in \mathbb{R}^n$ and $\rho \in \mathbb{R}$, the initializations $w^0, y^0, v_0^0, v_1^0, \dots, v_M^0 \in \mathbb{R}^n$, and $\lambda^0 \in \mathbb{R}$, and the stopping criteria tolerance $\epsilon > 0$.

Output: The minimizer (w^N, v^N) of (4.23).

Compute $\mathcal{J}_1, \dots, \mathcal{J}_M$ (where $M > 1$), which is a partition of the index set $\{1, \dots, J\}$ (i.e., $\mathcal{J}_i \cap \mathcal{J}_j \neq \emptyset$ and $\cup_{m=1}^M \mathcal{J}_i = \{1, \dots, J\}$), such that $\{q_j : j \in \mathcal{J}_m\}$ are linearly independent for each $m = 1, \dots, M$.

Compute the invertible matrices $Q_1, \dots, Q_M$ with n rows and n columns, such that the first $\mathcal{J}_m$ column vectors of Q_m is $\{q_j : j \in \mathcal{J}_m\}$.

for $\ell = 1, 2, \dots$ **do**

Update $v_0^{\ell+1} \in \mathbb{R}^n$ by

$$v_0^{\ell+1} = \frac{Y}{1+\rho} - \frac{\rho}{1+\rho}\left(y^\ell - w^\ell - \sum_{i=1}^M v_i^\ell\right). \tag{4.46}$$

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

$$w^{\ell+1} = \Pi_{\text{cone }\{p_1,\dots,p_I\}}\left(y^\ell + v_0^{\ell+1} - \sum_{i=1}^M v_i^\ell\right). \tag{4.47}$$

By convention, if $I = 0$, then we define $w^{\ell+1} = 0$;

Update $v_m^{\ell+1} \in \mathbb{R}^n$ for $m = 1, \dots, M$ iteratively by

$$v_m^{\ell+1} = E^{-1}\tilde{v}_m^{\ell+1}, \quad \tilde{v}_m^{\ell+1} = \Pi_{\tilde{C}_m}\left(E^{-1}(u - \alpha c)\right), \tag{4.48}$$

where $u, c \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$ are defined in (4.42), the matrix E is defined by (4.44), and the polyhedral convex cone $\tilde{C}_m$ is defined in (4.45);

Update $y^{\ell+1}, \lambda^{\ell+1} \in \mathbb{R}^n$ by

$$y^{\ell+1} = y^\ell + v_0^{\ell+1} - w^{\ell+1} - \sum_{m=1}^M v_m^{\ell+1}, \quad \lambda^{\ell+1} = \lambda^\ell + \sum_{m=1}^M \langle (Q_m^{-1})^T \mathbf{1}, v_m^{\ell+1}\rangle - 1. \tag{4.49}$$

if $\|w^\ell - w^{\ell+1}\|^2 \le \epsilon$, $\max_{m=1,\dots,M} \|v_m^\ell - v_m^{\ell+1}\|^2 \le \epsilon$, $\|y^\ell - y^{\ell+1}\|^2 \le \epsilon$, *and* $\|\lambda^\ell - \lambda^{\ell+1}\|^2 \le \epsilon$ **then**

return $(w^N, v^N) = \left(w^{\ell+1}, \sum_{m=1}^M v_m^{\ell+1}\right)$.

end

end

where the constraint set $C' \subseteq \mathbb{R}^{n+1}$ is defined by

$$C' := \left\{ \begin{pmatrix} x \\ s \end{pmatrix} \in \mathbb{R}^{n+1} : A \begin{pmatrix} x \\ s \end{pmatrix} \leq b \right\},$$
$$\text{where } A = \begin{pmatrix} p_1 & 0 \\ \vdots & \vdots \\ p_m & 0 \\ q_1 & -1 \\ \vdots & \vdots \\ q_{\tilde{m}} & -1 \end{pmatrix} \in \mathcal{M}_{m+\tilde{m},n+1}(\mathbb{R}),\ b = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_m \\ \beta_1 \\ \vdots \\ \beta_{\tilde{m}} \end{pmatrix} \in \mathbb{R}^{m+\tilde{m}}. \tag{4.51}$$

We present some numerical results to compare the performance of the two methods. In each example, we set all the tolerances (corresponding to the inequalities) in Algorithm 1 to 10^{-8}. We then pick the tolerances for the interior-point method such that the average ℓ^2 error of the two methods are of the same order of magnitude. Additionally, for each example, we compute the running time as the average time per call in seconds over 100 runs, the number of iterations as the average iteration count per call over 100 runs, and the ℓ^2 error as the average ℓ^2 error of the computed minimizer per call over 100 runs. To compute the ℓ^2 errors, we compute the true minimizer as the numerical minimizer resulting from running Algorithm 1 with all tolerances set to 10^{-15} and quadprog with all tolerances set to 10^{-16}. We run all of these numerical comparisons on an 8th Gen Intel Laptop Core i5-8250U with a 1.60GHz processor.

We first test the performance of the two methods on computing the minimizer in (4.3), where the parameters of the problem are chosen randomly. Specifically, in (4.3), we let γ be a random vector in $[0, 25] \times [-25, 25]^{n-1}$, C be a random

cone (where the support vectors p_i are random vectors in $[0,1] \times [-1,1]^{n-1}$ and $\alpha_i = 0$, $i = 1, \ldots, 2n$), and φ be a random function (where q_j are random vectors in $[-2.5, 2.5]^n$ and β_j are random scalars in $[-1,1]$, $j = 1, \ldots, 2n$). We note that we pick the first elements of γ and each p_i to be positive in order to lower the probability that γ is in the set C. We also initialize Algorithm 1 with a random vector x_0 in $[-\frac{1}{2n}, \frac{1}{2n}]^n \cap C$. For this example, we use the following tolerances for quadprog: for $n = 8$, we use a function tolerance of 10^{-15} and a step tolerance of 10^{-8}, and for $n = 16$, we use a function tolerance of 10^{-14} and a step tolerance of 10^{-8}. In Table 4.1, we see that for $n = 8, 16$, both methods achieve average ℓ^2 errors on the order of 10^{-12} in similar running times (between 2×10^{-3} and 2×10^{-2} seconds per call on average), although Algorithm 1 has slightly slower average running time than the interior-point method in 16 dimensions (approximately 2×10^{-2} seconds versus 4×10^{-3} seconds, respectively). Thus, we see that both methods have comparable performance in this example even in high dimensions.

n	**Algorithm**	**Running Time (s)**	**Number of Iterations**	ℓ^2 **Error**
8	Algorithm 1	7.0770e-03	7.72	2.1926e-12
	Interior-Point Method	2.4851e-03	9.51	2.0896e-12
16	Algorithm 1	1.5590e-02	15.49	4.0208e-12
	Interior-Point Method	4.1559e-03	12.06	1.0172e-12

Table 4.1: Computational results comparing the performance of Algorithm 1 and the interior-point method in computing the minimizer of (4.3) with random γ, C, and φ and for different dimensions n. The running times, iteration counts, and ℓ^2 errors of the computed minimizer correspond to their average values per call over 100 runs.

Next, we test the performance of the two methods on a more structured problem. Specifically, in (4.3), we let γ be a random vector in $[0, 25] \times [-25, 25]^{n-1}$, C be the ℓ^1 unit ball, and φ be the ℓ^∞ norm. Note that in this example, the number of support vectors for the set C scales exponentially with the dimension n ($m = 2^n$). We initialize Algorithm 1 randomly as in the previous example. Additionally, we use the following tolerances for quadprog: we use a function tolerance of 10^{-16}

and a step tolerance of 10^{-8}. In Table 4.2, we see that when $n = 8$, on average, Algorithm 1 achieves ℓ^2 errors on the order of 10^{-15} in less than 2.5×10^{-3} seconds, whereas the interior-point method requires approximately 3.9×10^{-1} seconds on average to achieve similar ℓ^2 errors. When $n = 16$, the interior-point method runs out of memory and thus is unable to compute a minimizer for this case. However, Algorithm 1 does not suffer from the same memory issues in this case and is able to compute the minimizer in 7.6×10^{-2} seconds on average. Thus, in this more structured example, we see that Algorithm 1 has faster run times than the interior-point method and does not have memory issues, whereas the interior-point method does, when the dimension n is high.

n	**Algorithm**	**Running Time (s)**	**Number of Iterations**	ℓ^2 **Error**
8	Algorithm 1	2.4512e-03	9.73	7.3964e-15
	Interior-Point Method	3.9079e-01	13.54	1.9915e-15
16	Algorithm 1	7.6073e-02	18.24	2.3438e-14
	Interior-Point Method	n/a	n/a	n/a

Table 4.2: Computational results comparing the performance of Algorithm 1 and the interior-point method in computing the minimizer of (4.3) with $C = \{x \in \mathbb{R}^n : \|x\|_1 \leq 1\}$, $\varphi = \|\cdot\|_\infty$, and random γ for different dimensions n. The running times, iteration counts, and ℓ^2 errors of the computed minimizers correspond to their average values per call over 100 runs. When $n = 16$, the interior-point method runs out of memory and thus is unable to compute a minimizer.

4.4 Numerical applications to multi-time HJ PDEs and certain linear optimal control problems

In this section, we demonstrate how our algorithms can be used in different applications. We focus on optimal control problems and Hamilton–Jacobi PDEs. Optimal control problems are an important class of optimization problems with many applications, including robot manipulator control [68, 53, 57, 35, 17], humanoid

robot control [31, 34, 61, 37, 33, 27], and trajectory planning [18, 85, 46, 26, 79, 67]. One widely used setup is when the dynamical system is a linear ODE. Under this setup, with time discretization, the original problem may be converted to a multi-time HJ PDE (see [25, 59, 69]). Therefore, we show the possible applications of our proposed algorithm for solving certain multi-time HJ PDEs (see Section 4.4.1) and certain optimal control problems with linear dynamics (see Section 4.4.2).

For both applications, we apply an ADMM scheme [43] to solve the corresponding problem, and we use Algorithm 1 as a building block in ADMM. ADMM is an iterative method for solving optimization problems of the following form:

$$\min_{x\in\mathbb{R}^n} f(x) + g(x), \tag{4.52}$$

where f and g are convex functions with numerically computable proximal points. The k-th iteration of the ADMM scheme reads:

$$\begin{cases} p^{k+1} = \arg\min\limits_{p\in\mathbb{R}^n} \left\{ f(p) + \dfrac{\rho}{2}\|p - q^k + y^k\|^2 \right\}, \\ q^{k+1} = \arg\min\limits_{q\in\mathbb{R}^n} \left\{ g(q) + \dfrac{\rho}{2}\|p^{k+1} - q + y^k\|^2 \right\}, \\ y^{k+1} = y^k + p^{k+1} - q^{k+1}, \end{cases} \tag{4.53}$$

where ρ is a positive parameter. If either f or g is a convex polyhedral function whose domain is a polyhedral set, then computing the corresponding iterate (p^{k+1} or q^{k+1}, respectively) is equivalent to solving an optimization problem of the form (4.3) and hence, can be computed using Algorithm 1.

4.4.1 Application to multi-time Hamilton-Jacobi PDEs

In this section, we consider certain multi-time Hamilton-Jacobi PDEs of the form

$$\begin{cases} \dfrac{\partial V(x,t_1,\dots,t_N)}{\partial t_1} + H_1(\nabla_x V(x,t_1,\dots,t_N)) = 0 \quad x \in \mathbb{R}^n, t_1,\cdots,t_N > 0, \\ \qquad \vdots \\ \dfrac{\partial V(x,t_1,\dots,t_N)}{\partial t_j} + H_j(\nabla_x V(x,t_1,\dots,t_N)) = 0 \quad x \in \mathbb{R}^n, t_1,\cdots,t_N > 0, \\ \qquad \vdots \\ \dfrac{\partial V(x,t_1,\dots,t_N)}{\partial t_N} + H_N(\nabla_x V(x,t_1,\dots,t_N)) = 0 \quad x \in \mathbb{R}^n, t_1,\cdots,t_N > 0, \\ V(x,0,\cdots,0) = J(x) \qquad\qquad\qquad\qquad\quad x \in \mathbb{R}^n, \end{cases} \tag{4.54}$$

where the Hamiltonians $H_1,\dots,H_N : \mathbb{R}^n \to \mathbb{R}$ and the initial condition $J : \mathbb{R}^n \to \mathbb{R}$ are convex functions. The corresponding optimal control problem reads

$$\min\left\{\sum_{i=1}^{N}\int_{\sum_{j=1}^{i-1}t_j}^{\sum_{j=1}^{i}t_j} H_i^*(y'(t))dt + J(y(0)) \colon y\left(\sum_{j=1}^{N}t_j\right) = x\right\}. \tag{4.55}$$

The optimal control in the optimal control problem (4.55) can be recovered from the spatial gradient $\nabla_x V(x,t)$ of the viscosity solution V to the HJ PDE (4.54) (see [7], for instance). Under certain assumptions, the solution of (4.54) is given by the generalized Hopf formula [69]:

$$\begin{aligned} V(x,t_1,\dots,t_N) &= \max_{p\in\mathbb{R}^n}\left\{\langle p,x\rangle - J^*(p) - \sum_{i=1}^{N} t_i H_i(p)\right\}, \\ &= -\min_{p\in\mathbb{R}^n}\left\{-\langle p,x\rangle + J^*(p) + \sum_{i=1}^{N} t_i H_i(p)\right\}, \end{aligned} \tag{4.56}$$

where J^* is the Fenchel-Legendre transform of the initial condition J. We assume the objective function in the minimization problem in (4.56) can be written as $f+g$,

where $f, g : \mathbb{R}^n \to \mathbb{R}$ are two functions whose proximal points are numerically computable. Then, we apply an ADMM scheme (4.53) to solve the optimization problem in (4.56).

There are many cases for which (4.56) can be solved using the ADMM algorithm in (4.53). We will describe one such case and illustrate how ADMM and Algorithm 1 can be applied to solve (4.56). Assume the initial condition J is defined by

$$J(x) = \inf_{u \in \mathbb{R}^n} \left\{ \frac{1}{2}\|u\|^2 + \|x - u\|_1 \right\} \quad \forall x \in \mathbb{R}^n. \tag{4.57}$$

Then, its Fenchel-Legendre transform is $J^*(x) = \frac{1}{2}\|x\|^2 + I_{B_\infty}(x)$ (see [45, Chapter X.2] for the computation of Fenchel-Legendre transform), where I_{B_∞} denotes the indicator function of the unit ℓ_∞-ball $B_\infty = \{x \in \mathbb{R}^n : \|x\|_\infty \leq 1\}$. Assume the Hamiltonian H_1 is the ℓ_∞-norm, which can be expressed in the form of (4.2). Also assume that the remaining Hamiltonians are in the form of

$$H_j(p) = \|p\|_{E_j} = \sqrt{\langle p, E_j p \rangle} \quad \forall p \in \mathbb{R}^n, \tag{4.58}$$

for any $j = 2, \ldots, N$, where each E_j is a positive definite symmetric matrix with n rows and n columns. In this case, we set

$$\begin{aligned} f(p) &= t_1 H_1(p) + J^*(p) = t_1\|p\|_\infty + \frac{1}{2}\|p\|^2 + I_{B_\infty}(p), \\ g(p) &= \sum_{i=2}^{N} t_i H_i(p) - \langle p, x \rangle = \sum_{i=2}^{N} t_i \sqrt{\langle p, E_i p \rangle} - \langle p, x \rangle. \end{aligned} \tag{4.59}$$

The minimizer p^{k+1} in (4.53) is given by

$$\begin{aligned} p^{k+1} &= \arg\min_{p\in B_\infty} \left\{ t_1\|p\|_\infty + \frac{1}{2}\|p\|^2 + \frac{\rho}{2}\|p - q^k + y^k\|^2 \right\} \\ &= \arg\min_{p\in B_\infty} \left\{ t_1\|p\|_\infty + \frac{1+\rho}{2}\left\|p - \frac{\rho}{1+\rho}(q^k - y^k)\right\|^2 \right\} \\ &= \arg\min_{p\in B_\infty} \left\{ \frac{t_1}{1+\rho}\|p\|_\infty + \frac{1}{2}\left\|p - \frac{\rho}{1+\rho}(q^k - y^k)\right\|^2 \right\}. \end{aligned} \tag{4.60}$$

Hence, the minimizer p^{k+1} can be computed by solving (4.3), where we set $C = B_\infty$, $\varphi(p) = \frac{t_1}{1+\rho}\|p\|_\infty$, and $\gamma = \frac{\rho}{1+\rho}(q^k - y^k)$. In other words, in (4.1) and (4.2), we set

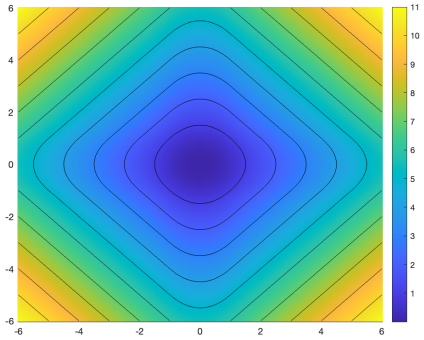
$$\begin{cases} m = \tilde{m} = 2n, \\ \alpha_i = 1,\ \beta_i = 0, \quad \forall i = 1,\dots,2n, \\ p_j = -p_{n+j} = e_j,\ q_j = -q_{n+j} = \dfrac{t_1}{1+\rho}e_j, \quad \forall j = 1,\dots,n, \end{cases}$$

where e_j is the j-th standard basis vector in $\mathbb{R}^n$, and we can compute p^{k+1} using Algorithm 1. The minimizer q^{k+1} in (4.53) is given by

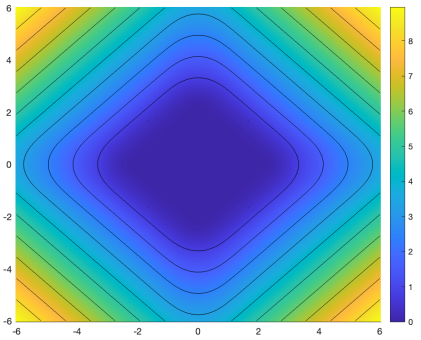
$$\begin{aligned} q^{k+1} &= \arg\min_{q\in\mathbb{R}^n} \left\{ \sum_{i=2}^{N} t_i\sqrt{\langle q, E_i q\rangle} - \langle q, x\rangle + \frac{\rho}{2}\|p^{k+1} - q + y^k\|^2 \right\} \\ &= \arg\min_{q\in\mathbb{R}^n} \left\{ \sum_{i=2}^{N} t_i\sqrt{\langle q, E_i q\rangle} + \frac{\rho}{2}\left\|p^{k+1} - q + y^k + \frac{x}{\rho}\right\|^2 \right\} \\ &= \arg\min_{q\in\mathbb{R}^n} \left\{ \sum_{i=2}^{N} \frac{t_i}{\rho}\sqrt{\langle q, E_i q\rangle} + \frac{1}{2}\left\|q - p^{k+1} - y^k - \frac{x}{\rho}\right\|^2 \right\}. \end{aligned} \tag{4.61}$$

We apply the algorithm in [20] to compute q^{k+1}.

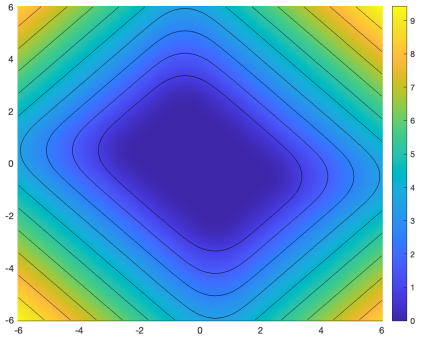
Next, we provide a 16-dimensional numerical example using the above set-up with three Hamiltonians ($N = 3$), where E_2 and E_3 are matrices with 16 rows and



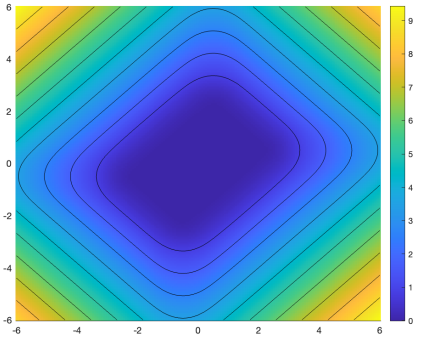
(a) $(t_1, t_2, t_3) = (0, 0, 0)$



(b) $(t_1, t_2, t_3) = (1, 0.5, 0.5)$



(c) $(t_1, t_2, t_3) = (0.5, 1, 0.5)$



(d) $(t_1, t_2, t_3) = (0.5, 0.5, 1)$

Figure 4.1: Evaluation of the value function $V(x, t_1, t_2, t_3)$ in (4.64) at $x \in [-6, 6]^2 \times \{0\}^{14}$ and different times t_1, t_2, and t_3. Plots for $(t_1, t_2, t_3) = (0, 0, 0)$ (initial condition), $(1, 0.5, 0.5)$, $(0.5, 1, 0.5)$, and $(0.5, 0.5, 1)$ 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 $x \in \mathbb{R}^{16}$.

16 columns satisfying

$$E_2 = \begin{pmatrix} \tilde{E}_2 & O & \dots & O \\ O & \tilde{E}_2 & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & \tilde{E}_2 \end{pmatrix}, \quad E_3 = \begin{pmatrix} \tilde{E}_3 & O & \dots & O \\ O & \tilde{E}_3 & \dots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \dots & \tilde{E}_3 \end{pmatrix}, \tag{4.62}$$

with $\tilde{E}_2$ and $\tilde{E}_3$ defined by

$$\tilde{E}_2 = \begin{pmatrix} 1.001 & -0.999 \\ -0.999 & 1.001 \end{pmatrix}, \quad \tilde{E}_3 = \begin{pmatrix} 1.001 & 0.999 \\ 0.999 & 1.001 \end{pmatrix}. \tag{4.63}$$

Thus, the corresponding multi-time HJ PDE is given by

$$\begin{cases} \dfrac{\partial V(x,t_1,t_2,t_3)}{\partial t_1} + \|\nabla_x V(x,t_1,t_2,t_3)\|_\infty = 0 & x \in \mathbb{R}^n, t_1,t_2,t_3 > 0, \\ \dfrac{\partial V(x,t_1,t_2,t_3)}{\partial t_2} + \|\nabla_x V(x,t_1,t_2,t_3)\|_{E_2} = 0 & x \in \mathbb{R}^n, t_1,t_2,t_3 > 0, \\ \dfrac{\partial V(x,t_1,t_2,t_3)}{\partial t_3} + \|\nabla_x V(x,t_1,t_2,t_3)\|_{E_3} = 0 & x \in \mathbb{R}^n, t_1,t_2,t_3 > 0, \\ V(x,0,0,0) = \inf\limits_{u\in\mathbb{R}^n} \left\{ \dfrac{1}{2}\|u\|^2 + \|x-u\|_1 \right\} & x \in \mathbb{R}^n. \end{cases} \tag{4.64}$$

Two-dimensional slices of the numerical solution V at times $(t_1,t_2,t_3) = (0,0,0)$, $(1,0.5,0.5)$, $(0.5,1,0.5)$, and $(0.5,0.5,1)$ are shown in Figures 4.1 (A)-(D), respectively. In each figure, we show the contour plot of the two-dimensional function $(x_1,x_2) \mapsto V(x_1,x_2,\mathbf{0},t_1,t_2,t_3)$ at the corresponding time (t_1,t_2,t_3). From this numerical experiment, we demonstrate our proposed Algorithm 1's ability to solve high-dimensional multi-time HJ PDEs.

4.4.2 Application to optimal control problems with linear dynamics

In this section, we consider certain optimal control problems with linear dynamics (see [25, 59]), which read as follows:

$$V(z,t) = \min\left\{\int_0^t L(u(s))ds + J(z(0))\colon z(t) = z, z'(s) = Az(s) + B(s)u(s)\ \forall s \in [0,t]\right\}, \tag{4.65}$$

where the running cost $L : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a proper, lower semicontinuous, convex, 1-coercive function, the initial cost $J : \mathbb{R}^n \to \mathbb{R}$ is convex, and $A, B(s)$ for all $s \in [0,t]$ are $n \times n$ matrices with real entries. The corresponding HJ PDE reads

$$\begin{cases} \dfrac{\partial V(z,t)}{\partial t} + \langle Az, \nabla_z V(z,t)\rangle + L^*(B(t)^T \nabla_z V(z,t)) = 0 & z \in \mathbb{R}^n, t > 0, \\ V(z,0) = J(z) & z \in \mathbb{R}^n. \end{cases} \tag{4.66}$$

After the change of variable $x(s) = e^{-As}z(s)$, the optimal control problem becomes

$$\tilde{V}(x,t) = \min\left\{\int_0^t L(u(s))ds + J(x(0))\colon x(t) = x, x'(s) = e^{-As}B(s)u(s)\ \forall s \in [0,t]\right\}, \tag{4.67}$$

where the terminal condition is $x = e^{-At}z$. Then, the corresponding HJ PDE reads

$$\begin{cases} \dfrac{\partial \tilde{V}(x,t)}{\partial t} + H(t, \nabla_x \tilde{V}(x,t)) = 0 & x \in \mathbb{R}^n, t > 0, \\ \tilde{V}(x,0) = J(x) & x \in \mathbb{R}^n, \end{cases} \tag{4.68}$$

with Hamiltonian $H : [0,\infty) \times \mathbb{R}^n \to \mathbb{R}$ defined by

$$H(s,p) = \sup_{u \in \mathbb{R}^n}\left\{\langle e^{-As}B(s)u, p\rangle - L(u)\right\} = L^*\left(B^T(s)e^{-A^T s}p\right) \quad \forall p \in \mathbb{R}^n, s \geq 0. \tag{4.69}$$

Under certain assumptions, the solution of the corresponding HJ PDE is given by the generalized Hopf formula (see [64, Section 5.3.2]):

$$V(z,t) = \tilde{V}(e^{-At}z,t) = \sup_{p\in\mathbb{R}^n}\left\{\langle e^{-At}z,p\rangle - \int_0^t L^*\left(B^T(s)e^{-A^Ts}p\right)ds - J^*(p)\right\}. \tag{4.70}$$

After discretization in time, the solution is approximated by the following multi-time Hopf formula:

$$\sup_{p\in\mathbb{R}^n}\left\{\langle e^{-At}z,p\rangle - \sum_{j=1}^{N} w_j L^*\left(B^T(t_j)e^{-A^Tt_j}p\right) - J^*(p)\right\} = -\min_{p\in\mathbb{R}^n}\{f(p)+g(p)\}, \tag{4.71}$$

where $\{w_j\}$ and $\{t_j\}$ are the weights and quadrature points of any quadrature rule approximating the integration over $[0,t]$. If we write the objective function in (4.71) in the form of $f+g$, where $f,g\colon \mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$ are convex lower semi-continuous functions whose proximal points are numerically computable, then we can solve (4.71) using the ADMM scheme (4.53). The optimal control $\bar{u}(s)$ in the two optimal control problems (4.65) and (4.67) can be computed by

$$\bar{u}(s) = \nabla L^*\left(B^T(s)e^{-A^Ts}\bar{p}\right), \tag{4.72}$$

where $\bar{p}$ is the numerical optimizer in (4.71) resulting from ADMM. Finally, the optimal trajectory $\bar{z}(s)$ in the original problem (4.65) can be computed by solving the following ODE:

$$\bar{z}'(s) = A\bar{z}(s) + B(s)\nabla L^*\left(B^T(s)e^{-A^Ts}\bar{p}\right). \tag{4.73}$$

Now, we describe an example of one such optimal control problem and demonstrate how the ADMM scheme (4.53) and Algorithm 1 can be applied to solve this

problem. Consider the case where the initial condition J is defined by

$$J(x) = \inf_{u \in B_1} \|x - u\|_1 \quad x \in \mathbb{R}^n, \tag{4.74}$$

where B_1 denotes the unit ℓ_1-ball $B_1 = \{x \in \mathbb{R}^n : \|x\|_1 \leq 1\}$ and the running cost L is the indicator of the ellipsoid $\mathcal{E} = \{p \in \mathbb{R}^n : \langle p, E^{-1}p \rangle \leq 1\}$, where E is a symmetric positive definite matrix with n rows and n columns. Then, we have (see [45, Chapter X.2] for the computation of the Fenchel-Legendre transforms)

$$J^*(p) = I_{B_\infty}(p) + \|p\|_\infty, \quad L^*(p) = \sqrt{\langle p, Ep \rangle} \quad \forall p \in \mathbb{R}^n, \tag{4.75}$$

where B_∞ denotes the unit ℓ_∞-ball in $\mathbb{R}^n$. In this case, we solve (4.71) by applying ADMM (4.53) with $f(p) = J^*(p)$ and $g(p) = \sum_{j=1}^N w_j L^*(B^T(t_j)e^{-A^T t_j}p) - \langle e^{-At}z, p \rangle$. Then, the first line in (4.53) becomes

$$p^{k+1} = \operatorname*{arg\,min}_{p \in \mathbb{R}^n} \left\{ J^*(p) + \frac{\rho}{2}\|p - q^k + y^k\|^2 \right\} = \operatorname*{arg\,min}_{p \in \mathbb{R}^n : \|p\|_\infty \leq 1} \left\{ \frac{1}{\rho}\|p\|_\infty + \frac{1}{2}\|p - q^k + y^k\|^2 \right\}, \tag{4.76}$$

which is in the form of (4.3), where we set $\gamma = q^k - y^k$, $C = B_\infty$, and $\varphi(p) = \frac{1}{\rho}\|p\|_\infty$. In other words, the parameters in (4.1) and (4.2) are defined by

$$\begin{cases} m = \tilde{m} = 2n, \\ \alpha_i = 1,\ \beta_i = 0, \quad \forall i = 1, \ldots, 2n, \\ p_j = -p_{n+j} = e_j,\ q_j = -q_{n+j} = \dfrac{e_j}{\rho}, \quad \forall j = 1, \ldots, n, \end{cases}$$

where e_j is the j-th standard basis vector in $\mathbb{R}^n$. With these parameters, we compute

p^{k+1} using Algorithm 1. The second line in (4.53) becomes

$$\begin{aligned} q^{k+1} &= \arg\min_{q\in\mathbb{R}^n}\left\{\sum_{j=1}^{N} w_j L^*\left(B^T(t_j)e^{-A^T t_j}q\right) - \langle e^{-At}z, q\rangle + \frac{\rho}{2}\|p^{k+1} - q + y^k\|^2\right\}, \\ &= \arg\min_{q\in\mathbb{R}^n}\left\{\sum_{j=1}^{N} \frac{w_j}{\rho}\sqrt{\langle q, e^{-At_j}B(t_j)EB^T(t_j)e^{-A^T t_j}q\rangle} + \frac{1}{2}\left\|q - p^{k+1} - y^k - \frac{e^{-At}z}{\rho}\right\|^2\right\}. \end{aligned} \tag{4.77}$$

We assume the matrix $B(s)$ is invertible for any $s \in [0, T]$. Then the optimization problem in (4.77) can be solved using the algorithm proposed in [20]. In our implementation, we apply the trapezoidal rule with a uniform grid as our quadrature rule, i.e., we set $\Delta t = \frac{t}{N-1}$, $w_1 = w_N = \frac{\Delta t}{2}$, $w_i = \Delta t$ for any $i = 2, \ldots, N-1$, and $t_j = (j-1)\Delta t$ for any $j = 1, \ldots, N$.

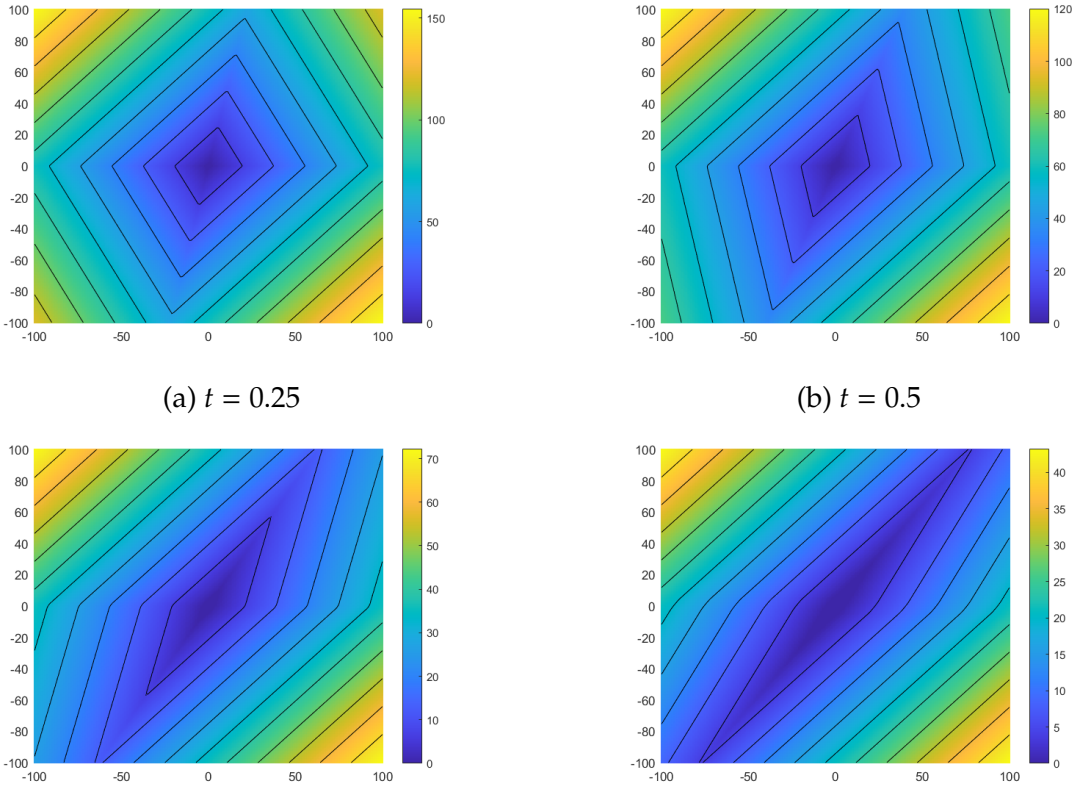
Next, we present a 16-dimensional example of the problem described above, where the matrices $A, B(s), E \in \mathcal{M}_{16,16}(\mathbb{R})$ are defined by

$$A = \begin{pmatrix} \tilde{A} & O & \ldots & O \\ O & \tilde{A} & \ldots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \ldots & \tilde{A} \end{pmatrix}, \quad B(s) = I_{16}, \quad E = \begin{pmatrix} \tilde{E} & O & \ldots & O \\ O & \tilde{E} & \ldots & O \\ \vdots & \vdots & \ddots & \vdots \\ O & O & \ldots & \tilde{E} \end{pmatrix}, \tag{4.78}$$

where I_{16} denotes the identity matrix in $\mathcal{M}_{16,16}(\mathbb{R})$ and $\tilde{A}$ and $\tilde{E}$ are matrices defined by

$$\tilde{A} = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}, \quad \tilde{E} = \begin{pmatrix} 0.0125 & 0.024 \\ 0.024 & 0.2501 \end{pmatrix}. \tag{4.79}$$

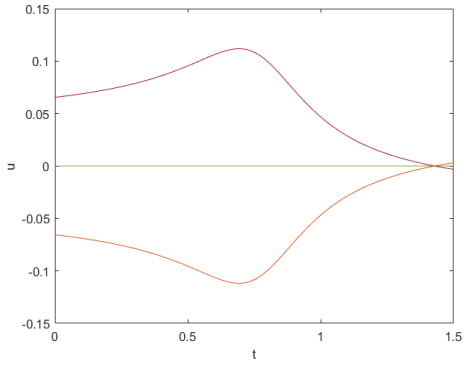
We compute the function value $V(z, t)$ in (4.65) at different times t and spatial points $z = (z_1, z_2, \mathbf{0})$ for $z_1, z_2 \in [-100, 100]$. The solution at $t = 0.25, 0.5, 1.0, 1.5$ is shown in Figures 4.2 (A)-(D), respectively. Moreover, given a terminal point x and



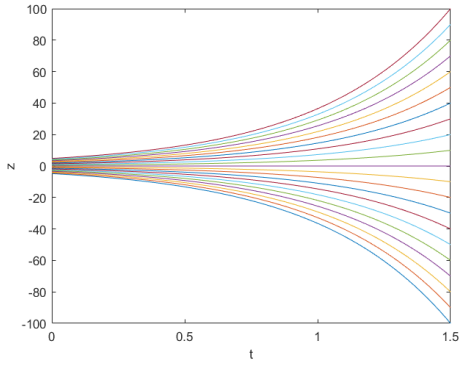
(a) $t = 0.25$ (b) $t = 0.5$

(c) $t = 1.0$ (d) $t = 1.5$

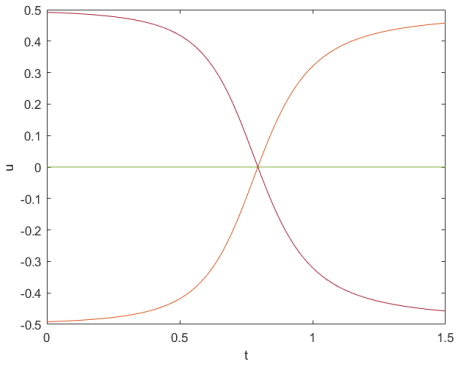
Figure 4.2: Evaluation of the value function $V(z, t)$ in (4.65) with matrices $A, B(s), E$ defined in (4.78) and initial condition J defined in (4.74) and for $z \in [-100, 100]^2 \times \{0\}^{14}$ and different times t. Plots for $t = 0.25$, 0.5, 1.0, and 1.5 are depicted in (A)-(D), respectively. Level lines are superimposed on the plots. The two axes in each figure correspond to the first and second components of the spatial variable $z \in \mathbb{R}^{16}$.



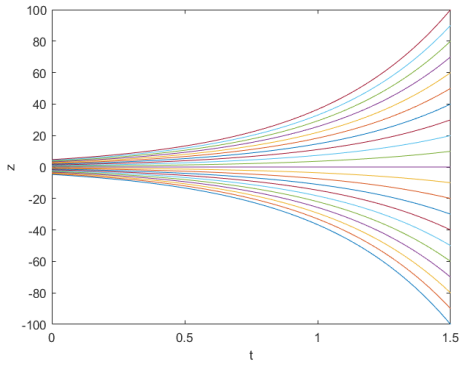
(a) First component of the optimal control



(b) First component of the optimal trajectory



(c) Second component of the optimal control



(d) Second component of the optimal trajectory

Figure 4.3: The graphs for the optimal controls (left) and optimal trajectories (right) for the 16-dimensional optimal control problem (4.65) with matrices $A, B(s), E$ defined in (4.78) and initial condition J defined in (4.74) and for $t = 1.5$ and $z = (z_1, z_2, \mathbf{0}) \in \mathbb{R}^{16}$, where $z_1 = z_2 \in [-100, 100]$. The first and second components of the optimal controls are plotted in (A) and (C), respectively, and the first and second components of the optimal trajectories are plotted in (B) and (D), respectively.

time horizon t, we compute the optimal control and optimal trajectory using (4.72) and (4.73). For this example, equation (4.72) becomes

$$\bar{u}(s) = \frac{EB^T(s)e^{-A^Ts}\bar{p}}{\sqrt{\langle B^T(s)e^{-A^Ts}\bar{p}, EB^T(s)e^{-A^Ts}\bar{p}\rangle}} = \frac{Ee^{-A^Ts}\bar{p}}{\sqrt{\langle e^{-A^Ts}\bar{p}, Ee^{-A^Ts}\bar{p}\rangle}}, \tag{4.80}$$

and equation (4.73) becomes

$$z'(s) = Az(s) + \frac{Ee^{-A^Ts}\bar{p}}{\sqrt{\langle e^{-A^Ts}\bar{p}, Ee^{-A^Ts}\bar{p}\rangle}}. \tag{4.81}$$

In Figure 4.3, we plot the optimal controls and optimal trajectories for a fixed time horizon $t = 1.5$ and several terminal conditions $(z_1, z_2, \mathbf{0}) \in \mathbb{R}^{16}$, where $z_1 = z_2 \in [-100, 100]$. The graphs of the first and second components of the optimal controls are plotted in Figures 4.3 (A) and (C), respectively, and the graphs of the first and second components of the optimal trajectories are plotted in Figures 4.3 (B) and (D), respectively. In each figure, the x-axis corresponds to the time variable s, and the y-axis corresponds to the value of $u(s)$ in (4.80) or $z(s)$ in (4.81). Additionally, the different colors correspond to the different terminal conditions $(z_1, z_2, \mathbf{0})$. From this example, we observe that our proposed Algorithm 1 can be used to solve certain high-dimensional optimal control problems with linear dynamics.

4.5 Summary

In this chapter, we proposed an algorithm for computing the proximal point of convex polyhedral function. We show that our algorithm has comparable or faster computational runtime than the interior-point method.

We present multi-time Hamilton Jacobi PDEs with the Hamiltonian H_1 is the ℓ_∞-

norm, which can be expressed in the form of (4.2) and the remaining Hamiltonians are in the form of matrix norm. The later part of the Hamiltonians can be computed by using the algorithm in Chapter 1 and the ℓ_∞-norm part can be computed by the algorithm we proposed in this Chapter. The purpose of this combination is that we can handle more general applications by combining the methods have been developed. For the next step, we will try to handle more general Hamiltonians by combining other numerical methods.

Chapter Five

Conclusion

In this thesis we have considered certain classes of (multi-time) Hamilton-Jacobi partial differential equations that arise from optimal control problems. We have proposed several efficient numerical optimization-based algorithm to efficiently compute solutions to the HJ PDEs in high dimensions. Specifically, in chapter 2, we have proposed a novel algorithm for computing the projection on the Minkowski sum of full ellipsoids. We have leveraged this algorithm to efficiently compute solution to optimal control problems with linear dynamics when the running cost is the characteristic function of a full ellipsoid and the terminal cost is quadratic. In chapter 3, we have proposed several algorithms to compute solutions to certain optimal control that involve degenerate ellipsoids. In chapter 4, we have proposed a novel algorithm for computing the proximal point of a polyhedral function. This algorithm has been used to solve optimal control considered in chapter 2 but with more general terminal costs.

There are several possible future works. First, algorithms described in chapter 3 should be implemented as numerical experiments should be performed to decide which approach performs best. Second, modern applications require computing solutions to these optimal control problems in real-time, e.g., for collision avoidance problems. In addition, computations must be performed on low-energy embedded system such as Field-Programmable Gate Arrays (FPGAs). It would be of interest to study the feasibility of implementing our proposed methods on FPGAs. Finally, the numerical algorithms presented in this thesis could be extended to cope with more general running and terminal costs that arise from application in unmanned autonomous vehicles.

Appendix A

Mathematical background and proofs of certain lemmas and propositions

A.1 Mathematical background

In this section, we review several basic definitions and theorems from convex analysis. All of the results and notation can be found in [44, 45]. We also refer the readers to [9, 15, 83].

First, a set C in $\mathbb{R}^n$ is convex if $\alpha x + (1-\alpha)y \in C$ whenever $x, y \in C$ and $\alpha \in [0,1]$. The relative interior of C, denoted as ri C, is the interior of C with respect to the minimal hyperplane containing C in $\mathbb{R}^n$. A point $x \in C$ is an extreme point of C if there are no two points $x_1, x_2 \in C$, such that $x_1 \neq x_2$ and $x = \frac{x_1+x_2}{2}$. A face F of C is a nonempty, convex set $F \subset C$, such that if $x, y \in C$ and $\alpha x + (1-\alpha)y \in F$ for some $\alpha \in (0,1)$, then the segment $\{\beta x + (1-\beta)y \colon \beta \in [0,1]\} \subset F$. The convex hull of a set C, denoted by conv C, consists of all the convex combinations of the elements of C, i.e., conv C is defined by

$$\operatorname{conv} C = \left\{ \sum_{i=1}^{m} \alpha_i x_i \colon m \in \mathbb{N}, \alpha_i \geq 0, x_i \in C \forall i = 1, \ldots, m, \sum_{i=1}^{m} \alpha_i = 1 \right\}.$$

If $C \subset \mathbb{R}^n$ is a nonempty closed convex set, then we define the projection operator $\Pi_C \colon \mathbb{R}^n \to C$ by

$$\Pi_C(x) = \arg\min \left\{ \frac{1}{2}\|y - x\|^2 : y \in C \right\}, \tag{A.1}$$

where $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^n$. We note that under the above assumptions on C, the minimizer in the definition of $\Pi_C(x)$ exists and is unique. The following proposition provides some characterizations of the projection operator.

Proposition A.1.1. *[44, Theorem III.3.1.1, Proposition III.3.2.3] Let C be a closed convex set in $\mathbb{R}^n$. Let x be a vector in $\mathbb{R}^n$. Then, $\bar{x} = \Pi_C(x)$ holds if and only if $\bar{x}$ is a point in C*

satisfying $\langle x - \bar{x}, u - \bar{x}\rangle \leq 0$ *for any* $u \in C$*, where we use the angle bracket* $\langle \cdot, \cdot \rangle$ *to denote the inner product operator in any Euclidean space* $\mathbb{R}^n$*. Moreover, if* C *is a closed convex cone, then* $\bar{x} = \Pi_C(x)$ *holds if and only if* $\bar{x}$ *is a point in* C *satisfying* $\langle x - \bar{x}, \bar{x}\rangle = 0$ *and* $\langle x - \bar{x}, u\rangle \leq 0$ *for any* $u \in C$*.*

A convex set K is a convex cone if $x \in K$ implies that $\alpha x \in K$ for all scalars $\alpha \geq 0$. In particular, the polyhedral cone generated by the vectors $\{v_1, v_2, \ldots, v_k\} \subset \mathbb{R}^n$, denoted by cone $\{v_1, v_2, \ldots, v_k\}$, is the set of all conical combinations of $\{v_1, v_2, \ldots, v_k\}$; that is,

$$\text{cone } \{v_1, v_2, \ldots, v_k\} = \left\{ \sum_{i=1}^{k} \alpha_i v_i : \alpha_i \geq 0, \forall i = 1, \ldots, k \right\}. \tag{A.2}$$

The polar cone K° of a convex cone K is defined by

$$K^\circ = \{y \in \mathbb{R}^n : \langle y, x\rangle \leq 0, \forall x \in K\}. \tag{A.3}$$

In particular, if $K = \text{cone } \{v_1, v_2, \ldots, v_k\}$, then $K^\circ = \{y \in \mathbb{R}^n : \langle y, v_i\rangle \leq 0, \forall i = 1, \ldots, k\}$. The projections onto a closed convex cone K and its polar cone K° can be related by the following proposition, also known as Moreau's identity.

Proposition A.1.2. *[44, Theorem III.3.2.5] (Moreau's Identity) Let* K *be a closed convex cone and* $x \in \mathbb{R}^n$*. Then, we have*

$$x = \Pi_K(x) + \Pi_{K^\circ}(x).$$

Moreover, there holds $\langle \Pi_K(x), \Pi_{K^\circ}(x)\rangle = 0$*.*

For any convex set C, the tangent cone of C at $x \in C$, denoted by $T_C(x)$, is

characterized by

$$d \in T_C(x) \text{ if and only if } \exists\{x_k\} \subset C, \{t_k\} \subset \mathbb{R}, \text{ such that } x_k \to x, t_k \downarrow 0, \frac{x_k - x}{t_k} \to d. \tag{A.4}$$

For any convex set C, the normal cone of C at $x \in C$, denoted by $N_C(x)$, can be characterized by

$$q \in N_C(x) \text{ if and only if } \langle q, y - x \rangle \leq 0 \text{ for any } y \in C. \tag{A.5}$$

A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is said to be convex if for any $\alpha \in (0,1)$ and any $x, y \in \mathbb{R}^n$,

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y).$$

A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is strictly convex if the above inequality is strict for all $x \neq y \in \mathbb{R}^n$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is 1-coercive if it is characterized by the following:

$$\lim_{k\to\infty} \frac{f(x_k)}{\|x_k\|} = +\infty \text{ for any } \{x_k\} \subset \mathbb{R}^n \text{ such that } \lim_{k\to\infty} \|x_k\| = +\infty.$$

The function f is called proper if it is not identically equal to $+\infty$. The domain of f, denoted by dom f, is defined to be the set where f does not take the value $+\infty$. A proper function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is called lower semi-continuous if for every sequence $\{x_k\}_{k=1}^{\infty} \subset \mathbb{R}^n$ with $\lim_{k\to\infty} x_k = x \in \mathbb{R}^n$, we have $\liminf_{k\to\infty} f(x_k) \geq f(x)$.

We denote $\Gamma_0(\mathbb{R}^n)$ to be the set of proper, convex, and lower semi-continuous functions from $\mathbb{R}^n$ to $\mathbb{R} \cup \{+\infty\}$. A vector $p \in \mathbb{R}^n$ is called a subgradient of $f \in \Gamma_0(\mathbb{R}^n)$

at $x \in \mathbb{R}^n$ if it satisfies

$$f(y) \geq f(x) + \langle p, y - x \rangle, \quad \forall y \in \mathbb{R}^n. \tag{A.6}$$

The collection of all such subgradients is called the subdifferential of f at x, denoted as $\partial f(x)$.

As a simple example, for any convex set C, define the indicator function I_C by

$$I_C(x) := \begin{cases} 0, & x \in C, \\ +\infty, & x \notin C. \end{cases} \tag{A.7}$$

Then, the subdifferential of the indicator function is given by

$$\partial I_C(x) = N_C(x). \tag{A.8}$$

It is easy to check that $0 \in \partial f(x)$ if and only if x is a minimizer of f. As a result, one can check whether x is a minimizer of a function by computing its subdifferential. The subdifferential operator is a (maximal) monotone operator. To be specific,

$$\langle p - q, x - y \rangle \geq 0 \text{ for any } p \in \partial f(x) \text{ and } q \in \partial f(y). \tag{A.9}$$

Moreover, in most cases, the subdifferential operator commutes with summation.

Proposition A.1.3. *[45, Corollary XI.3.1.2] Let $f, g \in \Gamma_0(\mathbb{R}^n)$. Assume* ri dom $f \cap$ ri dom $g \neq \emptyset$. *Then $\partial(f + g)(x) = \partial f(x) + \partial g(x)$ for any $x \in$* dom $f \cap$ dom g.

Subdifferential operators also have some continuity properties.

Proposition A.1.4. *[45, Proposition XI.4.1.1] Let f be a function in $\Gamma_0(\mathbb{R}^n)$. Let $\{x_n\}$ be a sequence in* dom f *converging to a point $x \in \mathbb{R}^n$, and let $\{p_n\}$ be a sequence in $\mathbb{R}^n$ converging to $p \in \mathbb{R}^n$ and satisfying $p_n \in \partial f(x_n)$ for any $n \in \mathbb{N}$. Then, we have $p \in \partial f(x)$.*

For any $f \in \Gamma_0(\mathbb{R}^n)$ and $x \in \text{dom } f$, the directional derivative at x along any direction d, denoted as $f'(x;d)$, is well-defined in $\mathbb{R} \cup \{\pm\infty\}$ and is given by

$$f'(x;d) = \inf_{t>0} \frac{f(x+td) - f(x)}{t}. \tag{A.10}$$

When f is differentiable at x, $f'(x;\cdot) = \langle \nabla f(x), \cdot \rangle$ is a linear function. In general, when f is not differentiable, $f'(x;\cdot)$ is only sublinear, in which case we can consider the linear functions dominated by it. Each normal vector of such linear functions gives a subgradient of f at x.

Proposition A.1.5. *[45, Example X.2.4.3] Let $f \in \Gamma_0(\mathbb{R}^n)$ and $x \in$ dom f such that $\partial f(x)$ is nonempty, then we have $(f'(x;\cdot))^* = I_{\partial f(x)}$. Moreover, if f is finite-valued or an indicator function of a convex closed polyhedral set, then $f'(x,\cdot)$ is a function in $\Gamma_0(\mathbb{R}^n)$, and hence we have $f'(x;d) = I^*_{\partial f(x)}(d)$ for any $x \in$ dom f and $d \in \mathbb{R}^n$.*

If $f \in \Gamma_0(\mathbb{R}^n)$, then its Fenchel–Legendre transform $f^* : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is defined by

$$f^*(s) = \sup_{x \in \mathbb{R}^n} \{\langle s, x \rangle - f(x)\}. \tag{A.11}$$

We note that for any $f \in \Gamma_0(\mathbb{R}^n)$, we have $f^* \in \Gamma_0(\mathbb{R}^n)$ and $(f^*)^* = f$.

A.2 Some technical lemmas for Section 4.2

In this section, we prove some lemmas that are used to prove the main results in Section 4.2. In each lemma, we consider the set C defined in (4.1) and the function φ defined in (4.2). We assume that x^k, d^k,γ, and Δt^k are the vectors and scalars defined in Algorithm 1 before the algorithm has terminated (i.e., when d^k is not zero).

Lemma A.2.1. *Let $x^k \in C$ and $d^k \neq 0$. Then, we have*

(a) $S(x^k) \cap S^+(d^k) = \emptyset$.

(b) $\tilde{S}(x^k) \cap \tilde{S}^+(d^k) = \emptyset$.

(c) $\Delta t^k > 0$.

Proof. To prove the conclusion, it suffices to show

$$\langle x^k, p_i \rangle < \alpha_i \quad \forall\, i \in S^+(d^k) \tag{A.12}$$

and

$$\langle x^k, q_j \rangle - \varphi(x^k) < \beta_j \quad \forall\, j \in \tilde{S}^+(d^k). \tag{A.13}$$

First, we show (A.12) by contrapositive. Assume there exists $I \in S^+(d^k)$ satisfying $\langle x^k, p_I \rangle \geq \alpha_I$. Then, by definition of C, we have

$$\langle x^k, p_I \rangle = \alpha_I. \tag{A.14}$$

Define the function F by (4.4). By definition of d^k, there holds $-d^k = \Pi_{\partial F(x^k)}(0)$,

which, by Proposition A.1.1, implies $-d^k \in \partial F(x^k)$ and

$$\langle d^k, y + d^k \rangle \leq 0 \quad \forall\, y \in \partial F(x^k). \tag{A.15}$$

By (A.14) and the definition of $S(x)$ in (4.12), we obtain $I \in S(x^k)$, and hence $p_I \in \partial I_C(x^k)$ by (4.11). For any $z \in \mathbb{R}^n$, we have

$$\begin{aligned}\langle -d^k + p_I, z - x^k \rangle + F(x^k) &= \langle -d^k, z - x^k \rangle + F(x^k) + \langle p_I, z - x^k \rangle + I_C(x^k) \\ &\leq F(z) + I_C(z) \\ &= F(z),\end{aligned}$$

where the inequality holds since we have $-d^k \in \partial F(x_k)$ and $p_I \in \partial I_C(x^k)$. Thus, by definition of subdifferential, $-d^k + p_I$ is a point in $\partial F(x^k)$. Then, choosing y in (A.15) to be $-d^k + p_I$, we get $\langle d^k, p_I \rangle \leq 0$, which contradicts with the assumption that $I \in S^+(d^k)$.

Now, we show (A.13) by contrapositive. Assume there exists $J \in \tilde{S}^+(d^k)$ satisfying $\langle x^k, q_J \rangle - \varphi(x^k) \geq \beta_J$. By definition of φ, we have $\langle x^k, q_J \rangle - \varphi(x^k) = \beta_J$, which implies $J \in \tilde{S}(x^k)$ by (4.14). According to (4.13), we obtain $q_J \in \partial\varphi(x^k)$. As a result, by Proposition A.1.5, there holds

$$\varphi'(x^k; d^k) = \sup_{q \in \partial\varphi(x^k)} \langle q, d^k \rangle \geq \langle q_J, d^k \rangle,$$

which contradicts with the assumption $J \in \tilde{S}^+(d^k)$. ■

Lemma A.2.2. *Let $x^k \in C$ and $d^k \neq 0$. Then, the following statements hold.*

(a) If $x = x^k + td^k$ for some $t \in (0, \Delta t_1^k)$, we have $x \in C$ and $\partial I_C(x) \subseteq \partial I_C(x^k)$. More specifically, there holds $\partial I_C(x) = \text{cone}\left\{p_i \colon i \in S(x^k), \langle p_i, d^k \rangle = 0\right\}$.

(b) *If* $x = x^k + td^k$ *for some* $t \in (0, \Delta t_2^k)$, *we have* $\partial\varphi(x) \subseteq \partial\varphi(x^k)$ *and* $\varphi'(x; d^k) = \varphi'(x^k; d^k)$. *More specifically, there holds* $\partial\varphi(x) = \operatorname{conv}\left\{q_j \colon j \in \tilde{S}(x^k), \varphi'(x^k; d^k) = \langle q_j, d^k\rangle\right\}$.

(c) *If* $x = x^k + td^k$ *for some* $t \in (0, \Delta t^k]$, *we have* $-(d^k + x^k - \gamma) \in \partial\varphi(x) + \partial I_C(x)$.

Proof. (a) Let $x = x^k + td^k$ hold for some $t \in (0, \Delta t_1^k)$. Then, we have

$$\langle p_i, x\rangle = \langle p_i, x^k\rangle + t\langle p_i, d^k\rangle \quad \forall i \in \{1, \ldots, m\}.$$

If $i \in S^+(d^k)$, there holds $\langle p_i, d^k\rangle > 0$, which implies

$$\langle p_i, x\rangle = \langle p_i, x^k\rangle + t\langle p_i, d^k\rangle < \langle p_i, x^k\rangle + \Delta t_1^k\langle p_i, d^k\rangle \le \langle p_i, x^k\rangle + \alpha_i - \langle p_i, x^k\rangle = \alpha_i, \quad \text{(A.16)}$$

where the second inequality holds by (4.9). If $i \notin S^+(d^k)$, there holds $\langle p_i, d^k\rangle \le 0$, which implies

$$\langle p_i, x\rangle = \langle p_i, x^k\rangle + t\langle p_i, d^k\rangle \le \langle p_i, x^k\rangle \le \alpha_i, \quad \text{(A.17)}$$

where the last inequality holds since x^k is in C. Therefore, $\langle p_i, x\rangle \le \alpha_i$ holds for each $i \in \{1, \ldots, m\}$, and hence x is in C.

Now, we compute the subdifferential set $\partial I_C(x)$. By (4.11), it suffices to compute the index set $S(x)$. We consider the two cases $i \in S^+(d^k)$ and $i \notin S^+(d^k)$ discussed above. For any $i \in S^+(d^k)$, notice that the first inequality in (A.16) is a strict inequality, which implies that $i \notin S(x)$. Then, we consider the case when $i \notin S^+(d^k)$. By definition, i is in $S(x)$ if and only if both inequalities in (A.17) become equalities. The first inequality in (A.17) becomes equality if and only if $\langle p_i, d^k\rangle = 0$ holds, and the second inequality in (A.17) becomes equality if and only if i is in $S(x^k)$.

Therefore, we get

$$\begin{aligned}\partial I_C(x) &= \text{cone } \{p_i \colon i \in S(x)\} = \text{cone } \left\{p_i \colon i \notin S^+(d^k), i \in S(x^k), \langle p_i, d^k\rangle = 0\right\} \\ &= \text{cone } \left\{p_i \colon i \in S(x^k), \langle p_i, d^k\rangle = 0\right\} \subseteq \text{cone } \left\{p_i \colon i \in S(x^k)\right\} = \partial I_C(x^k),\end{aligned}$$

where the third equality holds by Lemma A.2.1 (a), and the first and the last equality holds by (4.11).

(b) Let $x = x^k + td^k$ hold for some $t \in (0, \Delta t_2^k)$. We first compute the subdifferential set $\partial\varphi(x)$. By (4.13), it suffices to compute the index set $\tilde{S}(x)$. We consider the two cases of $j \in \tilde{S}^+(d^k)$ and $j \notin \tilde{S}^+(d^k)$. On the one hand, if $j \in \tilde{S}^+(d^k)$ holds, then we have

$$\begin{aligned}\langle q_j, x\rangle - \beta_j &= \langle q_j, x^k\rangle - \beta_j + t\langle q_j, d^k\rangle \\ &= \langle q_j, x^k\rangle - \beta_j + t\varphi'(x^k; d^k) + t(\langle q_j, d^k\rangle - \varphi'(x^k; d^k)) \\ &< \langle q_j, x^k\rangle - \beta_j + t\varphi'(x^k; d^k) + \Delta t_2^k(\langle q_j, d^k\rangle - \varphi'(x^k; d^k)) \\ &\leq \langle q_j, x^k\rangle - \beta_j + t\varphi'(x^k; d^k) + (\beta_j - \langle q_j, x^k\rangle + \varphi(x^k)) \\ &= \varphi(x^k) + t\varphi'(x^k; d^k) \leq \varphi(x),\end{aligned} \tag{A.18}$$

where the first inequality holds since $\langle q_j, d^k\rangle - \varphi'(x^k; d^k)$ is positive by definition of $\tilde{S}^+(d^k)$ in (4.7), the second inequality holds by (4.9), and the last inequality holds by convexity of $s \mapsto \varphi(x^k + sd^k)$. Since the first inequality is a strict inequality in (A.18), we conclude that $j \notin \tilde{S}(x)$. On the other hand, if $j \notin \tilde{S}^+(d^k)$ holds, then we have $\langle q_j, d^k\rangle \leq \varphi'(x^k; d^k)$, which implies

$$\langle q_j, x\rangle - \beta_j = \langle q_j, x^k\rangle - \beta_j + t\langle q_j, d^k\rangle \leq \langle q_j, x^k\rangle - \beta_j + t\varphi'(x^k; d^k) \leq \varphi(x^k) + t\varphi'(x^k; d^k) \leq \varphi(x), \tag{A.19}$$

where the second inequality holds by definition of φ in (4.2) and the last inequality

holds by convexity of $s \mapsto \varphi(x^k + sd^k)$. In this case, $j \in \tilde{S}(x)$ holds if and only if all three inequalities in (A.19) become equalites. In other words, we have

$$\begin{aligned}
\tilde{S}(x) &= \Big\{ j \in \{1, \ldots, \tilde{m}\} \colon\ j \notin \tilde{S}^+(d^k), \varphi'(x^k; d^k) = \langle q_j, d^k \rangle, \\
&\qquad j \in \tilde{S}(x^k), \varphi(x^k) + t\varphi'(x^k; d^k) = \varphi(x) \Big\}, \\
&= \Big\{ j \in \tilde{S}(x^k) \colon \varphi'(x^k; d^k) = \langle q_j, d^k \rangle, \varphi(x^k) + t\varphi'(x^k; d^k) = \varphi(x) \Big\}, \\
&\subseteq \tilde{S}(x^k),
\end{aligned} \tag{A.20}$$

where the second equality holds by Lemma A.2.1 (b). Therefore, we have $\tilde{S}(x) \subseteq \tilde{S}(x^k)$ and $\partial\varphi(x) \subseteq \partial\varphi(x^k)$. Using Proposition A.1.5, we have

$$\varphi'(x; d^k) = \sup_{q \in \partial\varphi(x)} \langle q, d^k \rangle \leq \sup_{p \in \partial\varphi(x^k)} \langle p, d^k \rangle = \varphi'(x^k; d^k), \tag{A.21}$$

where the inequality holds by $\partial\varphi(x) \subseteq \partial\varphi(x^k)$. Let q be an arbitrary vector in $\partial\varphi(x)$ and p be an arbitrary vector in $\partial\varphi(x^k)$. Since the subdifferential operator $\partial\varphi$ is monotone, by (A.9), we have

$$0 \leq \langle q - p, x - x^k \rangle = t\langle q - p, d^k \rangle,$$

which implies $\langle p, d^k \rangle \leq \langle q, d^k \rangle$. As a result, we apply Proposition A.1.5 again to conclude that

$$\varphi'(x; d^k) = \sup_{q \in \partial\varphi(x)} \langle q, d^k \rangle \geq \sup_{p \in \partial\varphi(x^k)} \langle p, d^k \rangle = \varphi'(x^k; d^k). \tag{A.22}$$

Combining (A.21) with (A.22), we obtain $\varphi'(x; d^k) = \varphi'(x^k; d^k)$. Note that this holds for any point $x^k + td^k$ with $t \in (0, \Delta t_2^k)$, and hence the function φ is affine on the line segment $\{x^k + sd^k \colon s \in [0, \Delta t_2^k)\}$. As a result, we have $\varphi(x^k) + t\varphi'(x^k; d^k) = \varphi(x)$,

and (A.20) becomes $\tilde{S}(x) = \left\{ j \in \tilde{S}(x^k) \colon \varphi'(x^k; d^k) = \langle q_j, d^k \rangle \right\}$. Therefore, we have

$$\partial\varphi(x) = \operatorname{conv} \left\{ q_j \colon j \in \tilde{S}(x) \right\} = \operatorname{conv} \left\{ q_j \colon j \in \tilde{S}(x^k) \colon \varphi'(x^k; d^k) = \langle q_j, d^k \rangle \right\}.$$

(c) By (4.5), we have $-d^k - x^k + \gamma = \Pi_{\partial I_C(x^k) + \partial\varphi(x^k)}(-x^k + \gamma)$. Hence, $-d^k - x^k + \gamma$ is a point in $\partial I_C(x^k) + \partial\varphi(x^k)$. In other words, there exist $p \in \partial I_C(x^k)$ and $q \in \partial\varphi(x^k)$ such that $p + q = -d^k - x^k + \gamma$. By Proposition A.1.5, we have

$$(\varphi + I_C)'(x^k; d^k) = \varphi'(x^k; d^k) + I_C'(x^k; d^k) \geq \langle q, d^k \rangle + \langle p, d^k \rangle = \langle p + q, d^k \rangle. \tag{A.23}$$

Moreover, since $p + q$ is the projection of $-x^k + \gamma$ onto $\partial I_C(x^k) + \partial\varphi(x^k)$, by Proposition A.1.1, for any $w \in \partial I_C(x^k) + \partial\varphi(x^k)$, we have

$$0 \geq \langle w - (-d^k - x^k + \gamma), -x^k + \gamma - (-d^k - x^k + \gamma) \rangle = \langle w - (-d^k - x^k + \gamma), d^k \rangle = \langle w - (p+q), d^k \rangle. \tag{A.24}$$

Therefore, by Proposition A.1.5 again, we have

$$\begin{aligned} (\varphi + I_C)'(x^k; d^k) = \varphi'(x^k; d^k) + I_C'(x^k; d^k) &= \sup_{u \in \partial I_C(x^k)} \langle u, d^k \rangle + \sup_{v \in \partial\varphi(x^k)} \langle v, d^k \rangle, \\ &= \sup_{w \in \partial I_C(x^k) + \partial\varphi(x^k)} \langle w, d^k \rangle \leq \langle p + q, d^k \rangle, \end{aligned} \tag{A.25}$$

where the last inequality follows from (A.24). Comparing (A.23) with (A.25), we conclude that all inequalities in (A.23) and (A.25) become equalities, and hence we have

$$\langle q, d^k \rangle = \varphi'(x^k; d^k), \quad \langle p, d^k \rangle = I_C'(x^k; d^k) = 0, \quad \langle -d^k - x^k + \gamma, d^k \rangle = \langle q + p, d^k \rangle = \varphi'(x^k; d^k), \tag{A.26}$$

where the second equality in the second formula holds because I_C is zero at $x^k + td^k$

for any $t \in [0, \Delta t^k]$. By (a) and (b), we have $q \in \partial\varphi(x)$ and $p \in \partial I_C(x)$, which implies that $-d^k - x^k + \gamma = p + q \in \partial\varphi(x) + \partial I_C(x)$ holds. ■

Lemma A.2.3. *If $\Delta t^k = 1$ and $x^k \in C$ for some $k \in \mathbb{N}$, then we have $d^{k+1} = 0$.*

Proof. By Lemma A.2.2 (c), we have $-(d^k + x^k - \gamma) \in \partial\varphi(x^{k+1}) + \partial I_C(x^{k+1})$. By straightforward computation, we get

$$\{x^{k+1} - \gamma\} + \partial\varphi(x^{k+1}) + \partial I_C(x^{k+1}) \ni x^{k+1} - \gamma - (d^k + x^k - \gamma) = (x^k + \Delta t^k d^k) - (d^k + x^k) = 0,$$

which implies $d^{k+1} = -\Pi_{\{x^{k+1}-\gamma\}+\partial\varphi(x^{k+1})+\partial I_C(x^{k+1})}(0) = 0$. ■

Lemma A.2.4. *Let $x^k \in C$ and $d^k \neq 0$. Let $x = x^k + td^k$ hold for some $t \in (0, \Delta t^k)$. Then, there holds $-\Pi_{\{x-\gamma\}+\partial\varphi(x)+\partial I_C(x)}(0) = d^k + x^k - x$.*

Proof. We have $-\Pi_{\{x-\gamma\}+\partial\varphi(x)+\partial I_C(x)}(0) = -\Pi_{\partial\varphi(x)+\partial I_C(x)}(-x+\gamma)-x+\gamma$. Then, to prove the statement, it suffices to prove $\Pi_{\partial\varphi(x)+\partial I_C(x)}(-x+\gamma) = -(d^k+x^k-\gamma)$. By Lemma A.2.2(c), we have $-(d^k + x^k - \gamma) \in \partial\varphi(x) + \partial I_C(x)$, and hence, by Proposition A.1.1, it remains to check

$$\langle -x + \gamma + d^k + x^k - \gamma, y + d^k + x^k - \gamma \rangle \leq 0 \quad \forall\, y \in \partial\varphi(x) + \partial I_C(x). \tag{A.27}$$

Let y be an arbitrary point in $\partial\varphi(x) + \partial I_C(x)$. By Lemma A.2.2 (a)-(b), we have $y \in \partial\varphi(x^k) + \partial I_C(x^k)$, which, by Proposition A.1.1 and (4.5), implies

$$\langle -x^k + \gamma + d^k + x^k - \gamma, y + d^k + x^k - \gamma \rangle \leq 0. \tag{A.28}$$

After some computation, we have

$$\begin{aligned}\langle -x + \gamma + d^k + x^k - \gamma, y + d^k + x^k - \gamma\rangle &= \langle (1-t)d^k, y + d^k + x^k - \gamma\rangle \\ &= (1-t)\langle -x^k + \gamma + d^k + x^k - \gamma, y + d^k + x^k - \gamma\rangle \\ &\leq 0,\end{aligned}$$

where the inequality holds since we have (A.28) and $1 - t > 1 - \Delta t^k \geq 0$ by (4.8). Therefore, the inequality (A.27) holds and the conclusion follows. ■

Lemma A.2.5. *Let k be an index such that the k-th and $(k+1)$-th iteration in Algorithm 1 is well-defined, and $d^{k+1} \neq 0$ holds. Then, we have $d^{k+1} \neq (1 - \Delta t^k)d^k$.*

Proof. Since $d^{k+1} \neq 0$ holds, Algorithm 1 does not terminate in the k-th iteration, and hence d^{k+2} and x^{k+2} are well-defined. By Lemmas A.2.1 and A.2.3, we have $\Delta t^k < 1$ and $\Delta t^{k+1} > 0$. We prove the statement by contrapositive. Assume the conclusion is not true, i.e., $d^{k+1} = (1 - \Delta t^k)d^k$. Since we have $\Delta t^k < 1$, by definition of Δt^k in (4.8), either $\Delta t^k = \Delta t_1^k$ or $\Delta t^k = \Delta t_2^k$ holds. Now, we consider these two cases one by one.

Firstly, let $\Delta t^k = \Delta t_1^k$. By definition of Δt_1^k in (4.9), there exists $i \in S^+(d^k)$ such that $\Delta t^k = \frac{\alpha_i - \langle p_i, x^k\rangle}{\langle p_i, d^k\rangle}$ is true. As a result, for any $t \in (0, \Delta t_1^{k+1})$, we have

$$\begin{aligned}\alpha_i - \langle p_i, x^{k+1} + td^{k+1}\rangle &= \alpha_i - \langle p_i, x^k\rangle - \langle p_i, \Delta t^k d^k + t(1 - \Delta t^k)d^k\rangle \\ &= \Delta t^k \langle p_i, d^k\rangle - (\Delta t^k + t(1 - \Delta t^k))\langle p_i, d^k\rangle \\ &= -t(1 - \Delta t^k)\langle p_i, d^k\rangle < 0,\end{aligned}$$

where the last inequality holds by definition of $S^+(d^k)$ in (4.6). Therefore, we conclude that $x^{k+1} + td^{k+1}$ is not in the set C, which contradicts to Lemma A.2.2 (a).

Then, we consider the case when $\Delta t^k = \Delta t_2^k$ holds. By definition of Δt_2^k in (4.9), there exists $j \in \tilde{S}^+(d^k)$ such that $\Delta t^k = \frac{\beta_j - \langle q_j, x^k \rangle + \varphi(x^k)}{\langle q_j, d^k \rangle - \varphi'(x^k; d^k)}$ is true. As a result, for any $t \in (0, \Delta t_2^{k+1})$, we have

$$\begin{aligned}
&\beta_j - \langle q_j, x^{k+1} + td^{k+1} \rangle + \varphi(x^{k+1} + td^{k+1}) \\
=&\beta_j - \langle q_j, x^k + \Delta t^k d^k + td^{k+1} \rangle + \varphi(x^k) + \Delta t^k \varphi'(x^k; d^k) + t\varphi'(x^{k+1}; d^{k+1}) \\
=&\beta_j - \langle q_j, x^k \rangle + \varphi(x^k) - \Delta t^k (\langle q_j, d^k \rangle - \varphi'(x^k; d^k)) - t(\langle q_j, d^{k+1} \rangle - \varphi'(x^{k+1}; d^{k+1})) \\
=&- t(\langle q_j, d^{k+1} \rangle - \varphi'(x^{k+1}; d^{k+1})),
\end{aligned} \tag{A.29}$$

where the first equality holds since φ is affine on $\{x^k + sd^k : s \in [0, \Delta t^k]\}$ and $\{x^{k+1} + sd^{k+1} : s \in [0, \Delta t^{k+1}]\}$ by Lemma A.2.2 (b). Applying (A.24) to the index $k+1$, we obtain

$$\begin{aligned}
0 &\geq \langle w - (-x^{k+1} + \gamma - d^{k+1}), d^{k+1} \rangle \\
&= \langle w - (-x^{k+1} + \gamma - (1 - \Delta t^k) d^k), d^{k+1} \rangle \\
&= \langle w - (-x^k + \gamma - d^k), d^{k+1} \rangle,
\end{aligned} \tag{A.30}$$

for any $w \in \partial I_C(x^{k+1}) + \partial \varphi(x^{k+1})$. Therefore, we have

$$\begin{aligned}
\varphi'(x^{k+1}; d^{k+1}) &= \sup_{w \in \partial I_C(x^{k+1}) + \partial \varphi(x^{k+1})} \langle w, d^{k+1} \rangle \leq \langle -x^k + \gamma - d^k, (1 - \Delta t^k) d^k \rangle \\
&= (1 - \Delta t^k) \langle -x^k + \gamma - d^k, d^k \rangle = (1 - \Delta t^k) \varphi'(x^k; d^k),
\end{aligned}$$

where the last equality follows from (A.26). Therefore, (A.29) becomes

$$\begin{aligned}
&\beta_j - \langle q_j, x^{k+1} + td^{k+1} \rangle + \varphi(x^{k+1} + td^{k+1}) \\
=&- t(\langle q_j, d^{k+1} \rangle - \varphi'(x^{k+1}; d^{k+1})) \\
=&- t(1 - \Delta t^k)(\langle q_j, d^k - \varphi'(x^k; d^k)) < 0,
\end{aligned}$$

where the last inequality holds since j is an index in $\tilde{S}^+(d^k)$. It contradicts to the

definition of φ in (4.2). ■

Lemma A.2.6. *Let k be an index such that the k-th and* $(k+1)$*-th iteration in Algorithm 1 is well-defined, and* $d^{k+1} \neq 0$ *holds. Then, we have*

$$(1 - \Delta t^k)^2 \|d^k\|^2 \geq (1 - \Delta t^k)\langle d^k, d^{k+1}\rangle \geq \|d^{k+1}\|^2. \tag{A.31}$$

Moreover, we have $(1 - \Delta t^k)\|d^k\| > \|d^{k+1}\|$.

Proof. By Lemma A.2.2 (c), we have $-(d^k + x^k - \gamma) \in \partial(\varphi + I_C)(x)$ for any $x \in \{x^k + sd^k : s \in [0, \Delta t^k]\}$ and $-(d^{k+1} + x^{k+1} - \gamma) \in \partial(\varphi + I_C)(y)$ for any $x \in \{x^{k+1} + td^{k+1} : t \in [0, \Delta t^{k+1}]\}$. Since the subdifferential operator is monotone, for any $s \in [0, \Delta t^k]$ and $t \in [0, \Delta t^{k+1}]$, we have

$$\begin{aligned}
0 &\leq \langle -(d^k + x^k - \gamma) + (d^{k+1} + x^{k+1} - \gamma), x^k + sd^k - x^{k+1}\rangle, \\
&= \langle d^{k+1} - (1 - \Delta t^k)d^k, (s - \Delta t^k)d^k\rangle, \\
0 &\leq \langle -(d^k + x^k - \gamma) + (d^{k+1} + x^{k+1} - \gamma), x^{k+1} - (x^{k+1} + td^{k+1})\rangle, \\
&= \langle d^{k+1} - (1 - \Delta t^k)d^k, -td^{k+1}\rangle.
\end{aligned}$$

Therefore, by straightforward computation, we have

$$\langle d^{k+1}, d^k\rangle \leq (1 - \Delta t^k)\|d^k\|^2, \quad \|d^{k+1}\|^2 \leq (1 - \Delta t^k)\langle d^{k+1}, d^k\rangle,$$

which implies (A.31). As a result, we have $(1 - \Delta t^k)\|d^k\| \geq \|d^{k+1}\|$. Now we prove that this is a strict inequality by contrapositive. Assume there holds $(1 - \Delta t^k)\|d^k\| = \|d^{k+1}\|$. Then, both inequalities in (A.31) becomes equalities, and hence we get

$$\|(1 - \Delta t^k)d^k - d^{k+1}\|^2 = (1 - \Delta t^k)^2\|d^k\|^2 - 2(1 - \Delta t^k)\langle d^k, d^{k+1}\rangle + \|d^{k+1}\|^2 = 0,$$

which contradicts with Lemma A.2.5. Therefore, we have $(1-\Delta t^k)\|d^k\| > \|d^{k+1}\|$. ■

Lemma A.2.7. *Let f be an arbitrary function in $\Gamma_0(\mathbb{R}^n)$. Let $x_1 \in \mathbb{R}^n$ and $x_2 \in \mathbb{R}^n$ satisfy $\partial f(x_1) = \partial f(x_2)$. Then, we have $\Pi_{\partial f(x_1)}(c - x_1) = \Pi_{\partial f(x_2)}(c - x_2)$ for any $c \in \mathbb{R}^n$.*

Proof. Let v_1 and v_2 be arbitrary vectors in $\partial f(x_1) = \partial f(x_2)$. Then, since the subdifferential operator ∂f is a monotone operator, we have

$$\langle v_1 - v_2, x_1 - x_2\rangle \geq 0 \text{ and } \langle v_2 - v_1, x_1 - x_2\rangle \geq 0,$$

which implies

$$\langle v_1 - v_2, x_1\rangle = \langle v_1 - v_2, x_2\rangle. \tag{A.32}$$

Now, let v_0 equal $\Pi_{\partial f(x_1)}(c - x_1)$, and hence v_0 is a vector in $\partial f(x_1)$. By definition of the projection operator, we have $\langle c - x_1 - v_0, v - v_0\rangle \leq 0$ for any $v \in \partial f(x_1)$. By (A.32), we conclude that

$$0 \geq \langle c - x_1 - v_0, v - v_0\rangle = \langle c - x_2 - v_0, v - v_0\rangle \tag{A.33}$$

holds for any $v \in \partial f(x_2)$. Therefore, $v_0 = \Pi_{\partial f(x_2)}(c - x_2)$ holds by definition of the projection operator, and the conclusion follows. ■

Proposition A.2.8. *Let the initial point x_0 be a point in C. Assume the function φ is an affine function. Then, Algorithm 1 terminates in finite iteration. In other words, there exists $K \in \mathbb{N}$ such that d^K is the zero vector, and x^K is the unique minimizer of the optimization problem* (4.3)*.*

Proof. Since φ is affine, it is differentiable. Moreover, its gradient is a constant, denoted by v. Hence, its subdifferential equals $\partial\varphi(x) = \{v\}$ for any $x \in \mathbb{R}^n$. Let k be

an index such that d^k and d^{k+1} are both non-zero. Denote $x(t) = x^k + td^k$ as $x(t)$ for any $t \in [0, \Delta t^k)$. By Lemma A.2.4, for any $t \in (0, \Delta t^k)$, we obtain

$$\begin{aligned}(1-t)d^k &= d^k + x^k - x(t) = -\Pi_{\{x(t)-\gamma\}+\partial\varphi(x(t))+\partial I_C(x(t))}(0) = -\Pi_{\{x(t)-\gamma+v\}+\partial I_C(x(t))}(0) \\ &= -\Pi_{\partial I_C(x(t))}(\gamma - v - x(t)) + \gamma - v - x(t).\end{aligned} \tag{A.34}$$

Note that this also holds for $t = 0$ by (4.5).

Denote $\Pi_{\partial I_C(x)}(\gamma - v - x)$ as $u(x)$ for any $x \in \mathbb{R}^n$. By Lemma A.1.2, since $\partial I_C(x(t))$ is a closed convex cone, we have $\langle u(x(t)), (1-t)d^k \rangle = 0$, and thus there holds

$$\|\gamma - v - (x^k + td^k)\|^2 = (1-t)^2\|d^k\|^2 + \|u(x^k + td^k)\|^2 \quad \forall t \in [0, \Delta t^k). \tag{A.35}$$

Let t be an arbitrary number in $(0, \Delta t^k)$. According to Lemma A.2.2 (a), we have

$$\partial I_C(x(t)) = \text{cone}\left\{p_i \colon i \in S(x^k), \langle p_i, d^k \rangle = 0\right\},$$

which does not depend on t. Then, by Lemma A.2.7, $u(x(t))$ remains the same for any $t \in (0, \Delta t^k)$. Moreover, by (A.34) and (4.5), we have

$$u(x(t)) = \gamma + v - x(t) - (1-t)d^k = \gamma + v - x^k - d^k = u(x^k). \tag{A.36}$$

Therefore, we get $u(x^k + sd^k) = u(x^k)$ for any $s \in [0, \Delta t^k)$. Similarly, we have $u(x^{k+1} + sd^{k+1}) = u(x^{k+1})$ for any $s \in [0, \Delta t^{k+1})$. As a result, we have

$$\begin{aligned}\|u(x^k)\|^2 &= \lim_{t \to (\Delta t^k)^-} \|u(x^k + td^k)\|^2 = \lim_{t \to (\Delta t^k)^-} \|\gamma - v - (x^k + td^k)\|^2 - (1-t)^2\|d^k\|^2 \\ &= \|\gamma - v - x^{k+1}\|^2 - (1 - \Delta t^k)^2\|d^k\|^2 < \|\gamma - v - x^{k+1}\|^2 - \|d^{k+1}\|^2 = \|u(x^{k+1})\|^2,\end{aligned} \tag{A.37}$$

where the first equality holds because $u(x^k + td^k) = u(x^k)$ holds, the second equality holds by (A.35), the inequality holds by Lemma A.2.6, and the last equality holds by applying (A.35) with index $k+1$ and time $t = 0$. Therefore, we obtain $\|u(x^k)\| < \|u(x^{k+1})\|$ for any index k satisfying $d^k \neq 0$ and $d^{k+1} \neq 0$.

Now, we prove that the algorithm will terminate at a finite step $K \in \mathbb{N}$. Note that there are only finite different sets $\partial I_C(x)$ for any $x \in C$ by (4.11), since the number of different $S(x)$'s is at most 2^m. Then, by Lemma A.2.7, the set $V = \{u(x) \colon x \in C\}$ is a finite set. However, $\{\|u(x^k)\|\}_k$ is a increasing sequence. Therefore, Algorithm 1 terminates in finite iterations. Moreover, the number of iterations is less than 2^m. ■

A.3 Some technical lemmas for Section 4.2.2.1

Lemma A.3.1. *Let $c, z \in \mathbb{R}^n$ be vectors and V be a convex subset in $\mathbb{R}^n$. Let $g \colon \mathbb{R} \to \mathbb{R}$ be the function defined by* (4.26)*. Assume there exists at least one vector $v \in \operatorname{ri} V$ such that $\langle c, v\rangle = 1$. Then, the minimization problem* (4.25) *has a unique solution. Moreover, the vector $\bar{v}$ is the minimizer of* (4.25) *if and only if $\bar{v} = \Pi_V(z - \lambda c)$, where $\lambda \in \mathbb{R}$ solves $g(\lambda) = 0$.*

Proof. The problem (4.25) is equivalent to the following optimization problem:

$$\min_{v \in \mathbb{R}^n} \left\{ \frac{1}{2}\|z - v\|^2 \colon v \in V, \langle c, v\rangle = 1 \right\}, \tag{A.38}$$

where we denote the objective function by F. To be specific, we have

$$F(v) = \frac{1}{2}\|z - v\|^2 + I_V(v) + I_{\{w \in \mathbb{R}^n \colon \langle c, w\rangle = 1\}}(v),$$

where I denotes the indicator function defined in (A.7). By assumption, the domain of the minimization problem is non-empty. Note that the objective function F is 1-coercive, lower semi-continuous, and strictly convex. Therefore, the minimizer exists and is unique. Moreover, the vector $\bar{v}$ is a minimizer if and only if $0 \in \partial F(\bar{v})$. By assumption, the interior of the set V intersects with the interior of the set $\{w \in \mathbb{R}^n \colon \langle c, w\rangle = 1\}$ (which equals itself). Therefore, by Proposition A.1.3, the subdifferential of F equals

$$\partial F(v) = v - z + \partial I_V(v) + \partial I_{\{w \in \mathbb{R}^n \colon \langle c, w\rangle = 1\}}(v) = v - z + N_V(v) + \{\lambda c \colon \lambda \in \mathbb{R}\} \tag{A.39}$$

for any $v \in \operatorname{dom} F$, where the second equality holds by (A.8). Therefore, $\bar{v}$ is a minimizer if and only if $\bar{v} \in V$, $\langle c, \bar{v}\rangle = 1$, and there exists $\lambda \in \mathbb{R}$ such that there holds

$$z - \lambda c \in \bar{v} + N_V(\bar{v}). \tag{A.40}$$

Note that (A.40) is equivalent to $\bar{v} = \Pi_V(z - \lambda c)$. As a result, $\bar{v}$ is a minimizer if and only if there exists $\lambda \in \mathbb{R}$ such that $\bar{v} = \Pi_V(z - \lambda c)$ and $1 = \langle c, \bar{v}\rangle = \langle c, \Pi_V(z - \lambda c)\rangle$. ■

Lemma A.3.2. *Let $c, z \in \mathbb{R}^n$ be vectors, V be a convex cone generated by $v_1, \ldots, v_N \in \mathbb{R}^n$, g be the function defined by* (4.26)*, and $W(\lambda)$ be the set defined by* (4.28) *for any $\lambda \in \mathbb{R}$. Then, the function g is a piecewise affine decreasing function, and the directional derivative of g at any point $\lambda \in \mathbb{R}$ is well-defined and given by*

$$\begin{aligned} g'(\lambda; 1) &= \lim_{h \to 0^+} \frac{g(\lambda + h) - g(\lambda)}{h} = -\|c\|^2 - \langle c, \Pi_{W(\lambda)}(-c)\rangle \leq 0, \\ g'(\lambda; -1) &= \lim_{h \to 0^+} \frac{g(\lambda - h) - g(\lambda)}{h} = \|c\|^2 - \langle c, \Pi_{W(\lambda)}(c)\rangle \geq 0. \end{aligned} \tag{A.41}$$

Proof. Since V is a convex polyhedral set, its polar cone V° is also a convex polyhedral set which equals

$$V^\circ = \{x \in \mathbb{R}^n : \langle v_i, x\rangle \leq 0, \forall i = 1, \ldots, N\}. \tag{A.42}$$

By [42], the directional derivative of the projection map Π_{V° at the point $x \in \mathbb{R}^n$ along the direction $d \in \mathbb{R}^n$ equals

$$(\Pi_{V^\circ})'(x; d) = \Pi_{T_{V^\circ}(\bar{x}) \cap Span(x-\bar{x})^\perp}(d), \tag{A.43}$$

where we use $\bar{x}$ to denote $\Pi_{V^\circ}(x)$ and $T_{V^\circ}(\bar{x})$ is the tangent cone of V° at $\bar{x}$. By straightforward computation using definition of tangent cone and Moreau's identity (see Proposition A.1.2), we get

$$T_{V^\circ}(\bar{x}) = \{y \in \mathbb{R}^n : \langle v_j, y\rangle \leq 0 \; \forall j \in \{i \in \{1, \ldots, N\} : \langle v_i, \bar{x}\rangle = 0\}\},$$
$$Span(x - \bar{x})^\perp = \{y \in \mathbb{R}^n : \langle x - \bar{x}, y\rangle = 0\} = \{y \in \mathbb{R}^n : \langle \Pi_V(x), y\rangle = 0\}.$$

Hence, we have

$$\begin{aligned} T_{V^\circ}(\bar{x}) \cap Span(x - \bar{x})^\perp = \{y \in \mathbb{R}^n : \langle \Pi_V(x), y\rangle = 0, \langle v_j, y\rangle \leq 0 \\ \forall j \in \{i \in \{1, \ldots, N\} : \langle v_i, \Pi_{V^\circ}(x)\rangle = 0\}\}. \end{aligned} \tag{A.44}$$

Note that $T_{V^\circ}(\bar{x}) \cap Span(x - \bar{x})^\perp$ equals $W(\lambda)$ if $x = z - \lambda c$ holds.

By Moreau's identity (see Proposition A.1.2), we get

$$g(\lambda) = \langle c, z - \lambda c - \Pi_{V^\circ}(z - \lambda c)\rangle - 1. \tag{A.45}$$

Let $x = z - \lambda c$ and $\bar{x} = \Pi_{V^\circ}(z - \lambda c)$. Then, the directional derivative of g at λ along

the direction 1 is given by

$$\begin{aligned} g'(\lambda;1) &= \langle c, -c - (\Pi_{V^\circ})'(x;-c)\rangle = -\|c\|^2 - \langle c, \Pi_{T_{V^\circ}(\bar{x})\cap Span(x-\bar{x})^\perp}(-c)\rangle \\ &= -\|c\|^2 - \langle c, \Pi_{W(\lambda)}(-c)\rangle = -\langle -c, -c - \Pi_{W(\lambda)}(-c)\rangle \\ &= -\langle \Pi_{W(\lambda)}(-c) + \Pi_{W(\lambda)^\circ}(-c), \Pi_{W(\lambda)^\circ}(-c)\rangle = -\|\Pi_{W(\lambda)^\circ}(-c)\|^2 \leq 0, \end{aligned} \tag{A.46}$$

where the last two equalities are given by Moreau's identity (see Proposition A.1.2). Similarly, the directional derivative of g at λ along the direction -1 equals

$$\begin{aligned} g'(\lambda;-1) &= \langle c, c - (\Pi_{V^\circ})'(x;c)\rangle = \|c\|^2 - \langle c, \Pi_{T_{V^\circ}(\bar{x})\cap Span(x-\bar{x})^\perp}(c)\rangle = \|c\|^2 - \langle c, \Pi_{W(\lambda)}(c)\rangle \\ &= \langle c, c - \Pi_{W(\lambda)}(c)\rangle = \langle \Pi_{W(\lambda)}(c) + \Pi_{W(\lambda)^\circ}(c), \Pi_{W(\lambda)^\circ}(c)\rangle = \|\Pi_{W(\lambda)^\circ}(c)\|^2 \geq 0. \end{aligned} \tag{A.47}$$

Therefore, the function g is a decreasing function.

It remains to prove that g is piecewise affine. Let F be a face of the polyhedral cone V and $N_V(F)$ be the normal cone of V on face F, whose definition is given by

$$N_V(F) = N_V(v),$$

where v is any point in the relative interior of F. Note that the set $N_V(F)$ does not depend on the choice of the point v. For more details of the definitions and properties, we refer readers to [87, Section 2.2]. First, let x, y be two vectors in $\mathbb{R}^n$, such that $\Pi_V(x)$ and $\Pi_V(y)$ are in the relative interior of a face F of V. Then, we will prove that the function $\Pi_V(\alpha x + (1-\alpha)y)$ is affine with respect to $\alpha \in [0,1]$, i.e., we will show that

$$\Pi_V(\alpha x + (1-\alpha)y) = \alpha\Pi_V(x) + (1-\alpha)\Pi_V(y) \quad \forall \alpha \in [0,1]. \tag{A.48}$$

Since V is a closed convex cone, Proposition A.1.1 gives us that to prove (A.48), it suffices to check that

$$\begin{aligned}\langle \alpha x + (1-\alpha)y - (\alpha \Pi_V(x) + (1-\alpha)\Pi_V(y)), \alpha \Pi_V(x) + (1-\alpha)\Pi_V(y)\rangle = 0,\\ \langle \alpha x + (1-\alpha)y - (\alpha \Pi_V(x) + (1-\alpha)\Pi_V(y)), v\rangle \leq 0, \quad \forall v \in V.\end{aligned} \tag{A.49}$$

By the characterization of $\Pi_V(x)$ and $\Pi_V(y)$ from Proposition A.1.1, we obtain

$$\langle x - \Pi_V(x), v\rangle \leq 0, \quad \langle y - \Pi_V(y), v\rangle \leq 0, \quad \forall v \in V. \tag{A.50}$$

By straightforward computation, we have

$$\langle \alpha x + (1-\alpha)y - (\alpha \Pi_V(x) + (1-\alpha)\Pi_V(y)), v\rangle = \alpha\langle x - \Pi_V(x), v\rangle + (1-\alpha)\langle y - \Pi_V(y), v\rangle \leq 0, \tag{A.51}$$

and hence the second line in (A.49) holds. Next, we check the first line in (A.49). Denote z to be $\alpha x + (1-\alpha)y - (\alpha \Pi_V(x) + (1-\alpha)\Pi_V(y))$. By Proposition A.1.1 and the definition of normal cones, $x - \Pi_V(x)$ is in $N_V(\Pi_V(x)) = N_V(F)$. Similarly, we have $y - \Pi_V(y) \in N_V(\Pi_V(x)) = N_V(F)$. As a result, their convex combination z is also in $N_V(F) = N_V(\Pi_V(x)) = N_V(\Pi_V(y))$. Therefore, we have $\Pi_V(\Pi_V(x) + z) = \Pi_V(x)$, which, by Proposition A.1.1, implies that $\langle z, \Pi_V(x)\rangle = 0$. Using the same argument, we also have $\langle z, \Pi_V(y)\rangle = 0$. Therefore, the first line in (A.49) follows, and the projection operator is affine on the line segment connecting x and y. Now, we prove that g is piecewise affine. For any face F of V, let I_F be the subset of $\mathbb{R}$ defined by

$$I_F = \{\lambda \in \mathbb{R} \colon \Pi_V(z - \lambda c) \in \operatorname{ri} F\}.$$

By [87, Theorem 2.1.2], the relative interior of all faces of V form a disjoint decomposition of V, and hence $\{I_F \colon F \text{ is a face of } V\}$ is a disjoint finite partition of $\mathbb{R}$ (the partition is finite since the convex. polyhedral cone V has finitely many

faces) Moreover, for any face F and any two scalars $\lambda_1, \lambda_2 \in I_F$, we have that $\Pi_V(z - (\alpha\lambda_1 + (1-\alpha)\lambda_2)c) = \alpha\Pi_V(z - \lambda_1 c) + (1-\alpha)\Pi_V(z - \lambda_2 c) \in \operatorname{ri} F$ for any $\alpha \in [0,1]$ by (A.48) and the fact that $\operatorname{ri} F$ is convex. Thus, for any $\alpha \in [0,1]$, we have $\alpha\lambda_1 + (1-\alpha)\lambda_2 \in I_F$ and

$$\begin{aligned} g(\alpha\lambda_1 + (1-\alpha)\lambda_2) &= \langle c, \Pi_V(z - (\alpha\lambda_1 + (1-\alpha)\lambda_2)c)\rangle - 1 \\ &= \langle c, \alpha\Pi_V(z - \lambda_1 c) + (1-\alpha)\Pi_V(z - \lambda_2 c)\rangle - 1 \\ &= \alpha g(\lambda_1) + (1-\alpha) g(\lambda_2). \end{aligned}$$

Therefore, the function g is piecewise affine. ■

Lemma A.3.3. *Let $c, z \in \mathbb{R}^n$ be vectors, V be a convex cone generated by $v_1, \ldots, v_N \in \mathbb{R}^n$, g be the function defined by* (4.26)*. If either $g'(\lambda; 1) = 0$ or $g'(\lambda; -1) = 0$ holds, then we have $g(\lambda) = -1$.*

Proof. Let $W(\lambda)$ be the set defined by (4.28) for any $\lambda \in \mathbb{R}$. First, if $g'(\lambda; 1) = 0$ holds, then by (A.46) we have

$$0 = \Pi_{W(\lambda)^\circ}(-c) = -c - \Pi_{W(\lambda)}(-c).$$

Hence, we have $-c = \Pi_{W(\lambda)}(-c) \in W(\lambda)$, which, by definition of $W(\lambda)$ in (4.28), implies $\langle \Pi_V(z - \lambda c), -c\rangle = 0$. As a result, we have $g(\lambda) = -1$. Similarly, if $g'(\lambda; -1) = 0$ holds, then (A.47) implies

$$0 = \Pi_{W(\lambda)^\circ}(c) = c - \Pi_{W(\lambda)}(c).$$

Hence, we have $c = \Pi_{W(\lambda)}(c) \in W(\lambda)$, which implies $\langle \Pi_V(z - \lambda c), c\rangle = 0$. Therefore, we still have $g(\lambda) = -1$. ■

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] J.-P. Aubin and A. Cellina, *Differential inclusions*, vol. 264 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Springer-Verlag, Berlin, 1984. Set-valued maps and viability theory.

[6] A. Bachouch, C. Huré, N. Langrené, and H. Pham, *Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications*, arXiv preprint arXiv:1812.05916, (2018).

[7] 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.

[8] R. O. Barr, *An efficient computational procedure for a generalized quadratic programming problem*, SIAM Journal on Control, 7 (1969), pp. 415–429.

[9] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer Publishing Company, Incorporated, 1st ed., 2011.

[10] R. E. Bellman, *Adaptive control processes: a guided tour*, Princeton university press, 1961.

[11] D. P. Bertsekas, *Reinforcement learning and optimal control*, Athena Scientific, Belmont, Massachusetts, (2019).

[12] 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.

[13] J. Bonnans and A. Shapiro, *Perturbation Analysis of Optimization Problems*, Springer Series in Operations Research and Financial Engineering, Springer New York, 2013.

[14] J.-F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal, *Numerical optimization: theoretical and practical aspects*, Springer Science & Business Media, 2006.

[15] J. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization: Theory and Examples*, Springer-Verlag New York, 2006.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] J. Darbon and T. Kim, *Efficient optimization algorithms for the projection on the minkowski sum of ellipsoid and its applications to certain optimal control problems*, In preparation., (2022).

[21] J. Darbon and G. P. Langlois, *Efficient and robust high-dimensional sparse logistic regression via nonlinear primal-dual hybrid gradient algorithms*, arxiv:2111.15426, (2021).

[22] 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.

[23] J. Darbon and T. Meng, *On decomposition models in imaging sciences and multi-time Hamilton-Jacobi partial differential equations*, SIAM Imaging Sciences, 13 (2020), pp. 971–1014.

[24] 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.

[25] J. Darbon and S. Osher, *Algorithms for overcoming the curse of dimensionality for certain hamilton–jacobi equations arising in control theory and elsewhere*, Research in the Mathematical Sciences, 3 (2016), p. 19.

[26] 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.

[27] 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.

[28] 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.

[29] S. Dolgov, D. Kalise, and K. Kunisch, *A tensor decomposition approach for high-dimensional Hamilton-Jacobi-Bellman equations*, arXiv preprint arXiv:1908.01533, (2019).

[30] 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.

[31] 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.

[32] L. C. Evans, *Partial differential equations*, American Mathematical Society, Providence, R.I., 2010.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] E. G. Gilbert, *An iterative procedure for computing the minimum of a quadratic form on a convex set*, SIAM Journal on Control, 4 (1966), pp. 61–80.

[41] 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.

[42] A. HARAUX, *How to differentiate the projection on a convex set in Hilbert space*, Journal of the Mathematical Society of Japan, 29 (1977), pp. 615–631.

[43] M. R. Hestenes, *Multiplier and gradient methods*, Journal of Optimization Theory and Applications, 4 (1969), pp. 303–320.

[44] J.-B. Hiriart-Urruty and C. Lemarechal, *Convex Analysis and Minimization Algorithms I: Fundamentals*, vol. 305, Springer-Verlag Berlin Heidelberg, 1993.

[45] ——, *Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods*, vol. 306, Springer-Verlag Berlin Heidelberg, 1993.

[46] 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.

[47] 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.

[48] C. Hu and C. Shu, *A discontinuous Galerkin finite element method for Hamilton–Jacobi equations*, SIAM Journal on Scientific Computing, 21 (1999), pp. 666–690.

[49] C. Huré, H. Pham, A. Bachouch, and N. Langrené, *Deep neural networks algorithms for stochastic control problems on finite horizon, part I: convergence analysis*, arXiv preprint arXiv:1812.04300, (2018).

[50] C. Huré, H. Pham, and X. Warin, *Some machine learning schemes for high-dimensional nonlinear PDEs*, arXiv preprint arXiv:1902.01599, (2019).

[51] 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).

[52] G. Jiang and D. Peng, *Weighted ENO schemes for Hamilton–Jacobi equations*, SIAM Journal on Scientific Computing, 21 (2000), pp. 2126–2143.

[53] L. Jin, S. Li, J. Yu, and J. He, *Robot manipulator control using neural networks: A survey*, Neurocomputing, 285 (2018), pp. 23 – 34.

[54] D. Kalise, S. Kundu, and K. Kunisch, *Robust feedback control of nonlinear PDEs by numerical approximation of high-dimensional Hamilton-Jacobi-Isaacs equations*, arXiv preprint arXiv:1905.06276, (2019).

[55] 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.

[56] 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.

[57] 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.

[58] M. R. Kirchner, G. Hewer, J. Darbon, and S. Osher, *A primal-dual method for optimal control and trajectory generation in high-dimensional systems*, in 2018 IEEE Conference on Control Technology and Applications (CCTA), 2018, pp. 1583–1590.

[59] ——, *A primal-dual method for optimal control and trajectory generation in high-dimensional systems*, in 2018 IEEE Conference on Control Technology and Applications (CCTA), 2018, pp. 1583–1590.

[60] M. R. Kirchner, R. Mar, G. Hewer, J. Darbon, S. Osher, and Y. T. Chow, *Time-optimal collaborative guidance using the generalized hopf formula*, IEEE Control Systems Letters, 2 (2018), pp. 201–206.

[61] 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.

[62] 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.

[63] A. Kurzhanski and P. Varaiya, *Dynamics and control of trajectory tubes. Theory and computation. Systems & Control: Foundations & Applications*, Springer, Cham–Heidelberg–New York–Dordrecht–London, 2014.

[64] A. B. Kurzhanski and P. Varaiya, *Dynamics and control of trajectory tubes*, Systems & Control: Foundations & Applications, Birkhäuser/Springer, Cham, 2014. Theory and computation.

[65] A. A. Kurzhanskiy and P. Varaiya, *Ellipsoidal toolbox (et)*, in Proceedings of the 45th IEEE Conference on Decision and Control, 2006, pp. 1498–1503.

[66] P. Lambrianides, Q. Gong, and D. Venturi, *A new scalable algorithm for computational optimal control under uncertainty*, arXiv preprint arXiv:1909.07960, (2019).

[67] 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.

[68] F. Lewis, D. Dawson, and C. Abdallah, *Robot Manipulator Control: Theory and Practice*, Control engineering, Marcel Dekker, 2004.

[69] 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.

[70] W. McEneaney, *Max-plus methods for nonlinear control and estimation*, Springer Science & Business Media, 2006.

[71] W. M. 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.

[72] 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.

[73] 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.

[74] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM Journal on Optimization, 2 (1992), pp. 575–601.

[75] Y. Nesterov, *Introductory lectures on convex optimization*, Applied Optimization, (2004).

[76] Y. Nesterov, *Smooth minimization of non-smooth functions*, Mathematical Programming, 103 (2005), pp. 127–152.

[77] 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.

[78] S. Osher and C. Shu, *High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations*, SIAM Journal on Numerical Analysis, 28 (1991), pp. 907–922.

[79] 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.

[80] X. Qin and N. T. An, *Smoothing algorithms for computing the projection onto a minkowski sum of convex sets*, Computational Optimization and Applications, 74 (2019), pp. 821–850.

[81] 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*, arXiv preprint arXiv:1903.06652, (2019).

[82] J. Rochet, *The taxation principle and multi-time Hamilton-Jacobi equations*, Journal of Mathematical Economics, 14 (1985), pp. 113 – 128.

[83] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

[84] V. R. Royo and C. Tomlin, *Recursive regression with neural networks: Approximating the HJI PDE solution*, arXiv preprint arXiv:1611.02739, (2016).

[85] 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.

[86] L. Ruthotto, S. Osher, W. Li, L. Nurbekyan, and S. W. Fung, *A machine learning framework for solving high-dimensional mean field game and mean field control problems*, arXiv preprint arXiv:1912.01825, (2019).

[87] R. Schneider, *Convex bodies: the Brunn-Minkowski theory*, vol. 151 of Encyclopedia of Mathematics and its Applications, Cambridge University Press, Cambridge, expanded ed., 2014.

[88] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, Journal of Computational Physics, 375 (2018), pp. 1339 – 1364.

[89] E. Todorov, *Efficient computation of optimal actions*, Proceedings of the national academy of sciences, 106 (2009), pp. 11478–11483.

[90] 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.