Algorithm for Overcoming the Curse of Dimensionality for Time-dependent Non-convex Hamilton-Jacobi Equations Arising from Optimal Control and Differential Games Problems

Yat Tin Chow* Jérôme Darbon† Stanley Osher* Wotao Yin*

Abstract

In this paper we develop a parallel method for solving possibly non-convex time-dependent Hamilton-Jacobi equations arising from optimal control and differential game problems. The subproblems are independent so they can be implemented in an embarrassingly parallel fashion, which usually has an ideal parallel speedup. The algorithm is proposed to overcome the curse of dimensionality [1, 2] when solving HJ PDE. We extend previous work in [5, 6] and apply a generalized Hopf formula to solve HJ PDE involving time-dependent and perhaps non-convex Hamiltonians. We suggest a coordinate descent method for the minimization procedure in the Hopf formula. This method is preferable when even the evaluation of the function value itself requires some computational effort, and also when we handle higher dimensional optimization problem. For an integral with respect to time inside the generalized Hopf formula, we suggest using a numerical quadrature rule. Together with our suggestion to perform numerical differentiation to minimize the number of calculation procedures in each iteration step, we are bound to have numerical errors in our computations. These errors can be effectively controlled by choosing an appropriate mesh-size in time and the method does not use a mesh in space. The use of multiple initial guesses is suggested to overcome possibly multiple local extrema in the case when non-convex Hamiltonians are considered. Our method is expected to have wide application in control theory and differential game problems, and elsewhere.

Mathematics Subject Classification (MSC2000):

Keywords: Hamilton-Jacobi equations, Hopf-Lax formula, optimal control, differential game

1 Introduction

Hamilton-Jacobi-Isaacs partial differential equations (HJ PDE) are of paramount importance in analyzing continuous/differential dynamic games, control theory problems and dynamical systems coming from the physical world, e.g. [12, 24]. Numerical solutions to HJ PDE have enjoyed great attention and have many practical applications. However, most of the methods, e.g. ENO/WENO-type methods [31], Dijkstra-type [8] methods such as fast marching [40] and fast sweeping [39], involve the introduction of a grid and a finite difference discretization of the Hamiltonian. These numerical solvers of HJ PDE are hence incapable of reasonable scaling with respect to dimension, and are therefore impossible to apply to problems in high dimensions.

A number of groups have therefore started to search for the possibility of developing an algorithm which can scale reasonably with dimension, e.g. in [22, 23]. In [6, 5] a causality-free method is proposed to solve for time-independent HJ PDE where a Hopf-Lax formula is used. The PDE is decoupled using the Hopf-Lax formula, and the value of the solution at a point can be effectively calculated as a d-dimensional minimization, which is extremely efficient to solve numerically.

*Department of Mathematics, UCLA, Los Angeles, CA 90095-1555. (ytchow/sjo/wotaoyin@math.ucla.edu) Research supported by ONR grant N000141410683, N000141210838, DOE grant DE-SC00183838, and NSF grant ECCS-1462398.

†Division of Applied Mathematics, Brown University, Providence RI, 02912, USA. (jerome_darbon@brown.edu).

In this paper we propose to extend the method proposed in [6, 5] and apply a generalized Hopf formula to solve time-dependent, possibly non-convex, Hamilton-Jacobi equations. The solution to the Hamilton-Jacobi equation is proposed to be evaluated point-wise by the generalized Hopf formula using a minimization process to solve for the problem, whose dimension is the same as that of the space in which the PDE sits. This way, the values of the solution at each point are decoupled and therefore they can be implemented in parallel without the necessity of any communication. In this work we propose a coordinate descent algorithm to minimize the functional. As an alternative choice of algorithm, we also suggest the ADMM/split Bregman iteration similar to [6, 5]. We observe that in our numerical tests for this time dependent problem type, the coordinate descent algorithm performs better in terms of computational time, but we remark that performance can depend on both the functional to be minimized and the software and hardware implementations. Each algorithm has an edge in different cases. The coordinate descent algorithm is of low complexity, thanks to the fact that only 1 coordinate is updated and only 2 function evaluations are necessary in each step of its iteration. This is especially preferable when we consider high-dimensional cases since the computational complexity in each of the iterations does not grow with dimension. Moreover, from theoretical analysis, we can see that the step-size can be usually chosen as 2 times the coordinate-wise Lipschitz constant of the functional gradient, which is considerably larger than that constant in the full gradient descent method. Furthermore, the value given by the generalized Hopf formula is exact, and therefore there is no numerical error introduced by a finite difference approximation. With a time integral inside the functional to be evaluated, we are bound to have a numerical quadrature rule which introduces numerical errors. The error of our numerical solution can be controlled to be arbitrarily small by controlling the error threshold given by our optimization method and the mesh-size of the 1-dimensional quadrature rule. We also propose the use of numerical differentiation when we perform coordinate descent to minimize computational complexity. The use of multiple initial guesses is suggested to overcome possible local extrema in the case when a non-convex Hamiltonian is encountered. The generalized Hopf formula, together with numerical differentiation and integration, are used for the evaluation of rapid numerical solutions for a time-dependent Hamiltonian in high dimensions, and this method is expected to have wide application in optimal control and differential games, where a non-convex Hamiltonian arises naturally.

The outline of our paper is as follows: in Section 2, we briefly introduce the optimal control system and its related time-dependent Hamilton-Jacobi equations that we focus on in this work, the generalized Hopf-Lax formula used to represent the solution, as well as the reachability function and the control parameters that we wish to obtain from the generalized Hopf-Lax formula. In Section 3, we suggest two algorithms: the coordinate descent algorithm and the ADMM/split Bregman algorithm, to perform the optimization problem given in the Hopf-Lax formula. Details of the respective optimization strategies are also introduced. Numerical experiments in Section 4 illustrate the effectiveness of our new algorithm in solving the HJ PDE.

2 Hamilton-Jacobi Equations, optimal control, differential games and generalized Hopf Formulae

In this section, we first give a brief introduction to the specific optimal control and differential games with possibly time-dependent linear control set that we are concerning with in this work, as well as a brief explanation on how we recast it as a problem of solving an HJ PDE. We follow discussion in [9, 5], see also [11], about optimal control and also for differential games, and their links with HJ PDE.

2.1 Differential game problems with time dependent control sets

Suppose we are given a fixed terminal time $T \in \mathbb{R}$, an initial time $t < T$, and two balanced (a set C is balanced if $C = -C$) compact convex sets $C, D \subset \mathbb{R}^d$ (which can have a dimension strictly less than d.) We may now consider two families of balanced compact convex sets $\{C(s)\}_{t<s<T}, \{D(s)\}_{t<s<T}$ where

$C(s) \subset C$ and $D(s) \subset D$ for all s. The sets $C(s)$ and $D(s)$ are considered as the possible choices of the controls and errors in a particular system at time s.

Let us denote $\mathcal{A}(t) = \{a : [t,T] \to C \,:\, a \text{ is measurable}, a(s) \in C(s)\}$, which we name it as the admissible set; and $\mathcal{B}(t) = \{b : [t,T] \to D \,:\, b \text{ is measurable}, b(s) \in D(s)\}$, which we can call the error set (or the error tolerence). We call the measurable function $a : [t,T] \to C$ in the set $\mathcal{A}(t)$ a control, and the function $b : [t,T] \to D$ in the set $\mathcal{B}(t)$ represents possible errors in the system.

Now with a fixed T, we consider the finite horizon problem. Suppose we are given an initial state $x \in \mathbb{R}^d$, we consider the (Lipschitz) solution $x : [t,T] \to \mathbb{R}^d$ of the following linear dynamic system with initial condition x at time t:

$$\begin{cases} \frac{dx}{ds}(s) = Mx(s) + N_C(s)a(s) + N_D(s)b(s) \quad \text{in} \quad (t,T) \\ x(t) = x \end{cases} \tag{2.1}$$

where M is a given $d \times d$ matrix independent of time, and $\{N_C(s)\}_{t<s<T}$, $\{N_D(s)\}_{t<s<T}$ are two families of $d \times d$ matrices with real entries. The function $x : [t,T] \to \mathbb{R}^d$ is regarded as the state of the system. To recall, $a(s)$ is the control input and $b(s)$ is the error at time s.

Now we can simplify the above equation by making a change of variable $z(s) = \exp(-sM)x(s)$, which gives the following new equation

$$\begin{cases} \frac{dz}{ds}(s) = \exp(-sM)N_C(s)a(s) + \exp(-sM)N_D(s)b(s) \quad \text{in} \quad (t,T)\,, \\ z(t) = \exp(-tM)x\,. \end{cases} \tag{2.2}$$

Given a running-cost Lagrangian functional $L : [t,T] \times C \times D \to \mathbb{R} \bigcup \{+\infty\}$ which is proper lower-semicontinous 1-coercive convex in the variable a and proper upper-semicontinuous 1-coercive concave in the variable b, together with a proper lower-semicontinuous 1-coercive convex terminal cost functional $J : \mathbb{R}^d \to \mathbb{R} \bigcup \{+\infty\}$, we now consider the following cost functional for a given initial time $t < T$, $z \in \mathbb{R}^d$, control a and error b:

$$K\left(t, z; a(\cdot), b(\cdot)\right) := \int_t^T L(s, a(s), b(s))\, ds + J(z(T))$$

where z is the solution to (2.2). K is now the cost functional to be considered, which is the sum of running cost L and the terminal cost J.

The aim of the control $a(\cdot)$ is to minimize $K\left(t, z; a(\cdot), b(\cdot)\right)$, while the aim of the error $b(\cdot)$ is to maximize $K\left(t, z; a(\cdot), b(\cdot)\right)$, i.e. to search for the greatest extent of error the system can tolerate. This way, the system (2.2) has become a differential game similar to [5, 9, 11].

To follow the standard differential game description and analysis, we now consider the lower value and the upper value of the differential game. For this purpose, we define a strategy for the control as a following map

$$\alpha : \mathcal{B}(t) \to \mathcal{A}(t)$$

provided that, for each $t \le s \le T$, we have

$$b(\tau) = \hat{b}(\tau) \text{ for a.e. } t \le \tau \le s \quad \Rightarrow \quad \alpha[b](\tau) = \alpha[\hat{b}](\tau) \text{ for a.e. } t \le \tau \le s\,.$$

Likewise, we may define a "strategy" for the error as

$$\beta : \mathcal{A}(t) \to \mathcal{B}(t)$$

provided for each $t \le s \le T$, we have

$$a(\tau) = \hat{a}(\tau) \text{ for a.e. } t \le \tau \le s \quad \Rightarrow \quad \beta[a](\tau) = \beta[\hat{a}](\tau) \text{ for a.e. } t \le \tau \le s\,.$$

We remark that a "strategy" for the error actually means finding the worst path that can attain the maximum error of the system.

Now let us denote the set of all strategies for the control as $\Gamma(t)$ and that for the error as $\Delta(t)$ beginning at time t, then we are ready to define the upper and lower values of the differential game. In fact, the lower value $V(t,z)$ is defined as

$$V(t,z) \quad := \quad \inf_{\beta\in\Delta(t)} \sup_{a\in\mathcal{A}(t)} K\left(t,z;a(\cdot),\beta[a](\cdot)\right) \tag{2.3}$$

for a given pair of (t,z). Similarly, we have the upper value $U(t,z)$ defined as

$$U(t,z) \quad := \quad \sup_{\alpha\in\Gamma(t)} \inf_{b\in\mathcal{B}(t)} K\left(t,z;\alpha[b](\cdot),b(\cdot)\right) \tag{2.4}$$

for a given pair of (t,z). We notice that the upper value is only here for completeness, and the main focus is the lower value $V(t,z)$, since it represents the best possible value given the initial and terminal states with starting time t under an error disturbance. Nonetheless, they coincide in some special cases. Our optimal control problem is now to compute $V(t,z)$ in (2.3) as well as finding the control a from the minimum argument of $V(t,z)$.

2.2 Hamilton-Jacobi equation and the generalized Hopf formula

From the dynamic programming optimality conditions and the main theorem in [12], the lower and upper values V and U are the viscosity solutions [3, 4] of some time-dependent HJ PDE. To elaborate, we first define the following two Hamiltonians:

$$H^+(t,p) = \min_{b\in D(T-t)} \max_{a\in C(T-t)} \left\{-\langle \exp(-(T-t)M)N_C(T-t)a + \exp(-(T-t)M)N_D(T-t)b, p\rangle + L(T-t,a,b)\right\},$$
$$H^-(t,p) = \max_{a\in C(T-t)} \min_{b\in D(T-t)} \left\{-\langle \exp(-(T-t)M)N_C(T-t)a + \exp(-(T-t)M)N_D(T-t)b, p\rangle + L(T-t,a,b)\right\}.$$

A simple class of Hamiltonian is obtained when $L(t,a,b)=0$, and in that case, $H^{\pm}(s,p)$ are homogeneous of degree 1 Hamiltonians. In fact, a very important class of problems described by ellipsoidal constraints [10] as follows:

$$\begin{aligned} C(t) \quad &:= \quad \{\langle x-\mathbf{a}_C(t), [\mathbf{Q}_C(t)]^{-1}(x-\mathbf{a}_C(t))\rangle \le 1\} \\ D(t) \quad &:= \quad \{\langle x-\mathbf{a}_D(t), [\mathbf{Q}_D(t)]^{-1}(x-\mathbf{a}_D(t))\rangle \le 1\}. \end{aligned}$$

where $\mathbf{Q}_C(t)$ and $\mathbf{Q}_D(t)$ are some positive definite matrices depending on t, and $\mathbf{a}_C(t), \mathbf{a}_D(t)$ are the respective base point of the moving ellipsoids at the particular time t. In this special case, we can compute that $H^+(t,p) = H^-(t,p)$ and they are of the form of

$$\begin{aligned} & H^{\pm}(t,p) \\ = \quad & \sqrt{\langle [-e^{-(T-t)M}N_C(T-t)]^*p, \mathbf{Q}_C(T-t)[-e^{-(T-t)M}N_C(T-t)]^*p\rangle} \\ & -\sqrt{\langle [-e^{-(T-t)M}N_D(T-t)]^*p, \mathbf{Q}_D(T-t)[-e^{-(T-t)M}N_D(T-t)]^*p\rangle} \\ & +\langle [-e^{-(T-t)M}N_C(T-t)]^*p, \mathbf{a}_C(T-t)\rangle + \langle [-e^{-(T-t)M}N_D(T-t)]^*p, \mathbf{a}_D(T-t)\rangle \end{aligned}$$

From now on we denote $H(t,p) := H^{\pm}(t,p)$ whenever $H^+ = H^-$.

From the dynamic programming principle and the main theorem in [12], together with a change of variable in the time variable by a flip of the axis, we immediately check that that $\varphi(t,z) := V(T-t,z) = U(T-t,z)$ is actually the viscosity solution [3, 4] to the following HJ PDE:

$$\begin{cases} \frac{\partial}{\partial t}\varphi(z,t) + H(t,\nabla_z\varphi(z,t)) = 0 & \text{in } \mathbb{R}^d\times(0,T), \\ v(z,0) = J(z) & \text{in } \mathbb{R}^d. \end{cases} \tag{2.5}$$

The same conclusion holds with the help of the Fenchel-Legendre transform when L is non-zero. To recall, the Fenchel-Legendre transform of a general convex, proper, lower semi-continuous function $f : \mathbb{R}^d \to \mathbb{R} \bigcup \{+\infty\}$, see e.g. [34]

$$f^*(v) := \sup_{x \in \mathbb{R}^d} \{\langle x, v\rangle - f(x)\}. \tag{2.6}$$

2.3 Generalized Hopf formula, reachability indicator and control parameter

Now, the viscosity solution to (2.5) can be explicitly given by the following generalized Hopf formula [11, 21, 35] as

$$\varphi(t,z) = -\min_{p \in \mathbb{R}^d} \left\{ J^*(p) + \int_0^t H(s,p)ds - \langle z, p\rangle \right\}. \tag{2.7}$$

where J^* is the Fenchel transform of J where $H : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ is a continous 1-coercive Hamiltonian function, (with an additional technical assumption that $H(s,p)$ is required to be pseudo-convex w.r.t. the argument p in the set of minimum arguments

$$S_0(t,z) := \text{argmin}_{p \in \mathbb{R}^d} \left\{ J^*(p) + \int_0^t H(s,p)ds - \langle z, p\rangle \right\}$$

for a given (t,z), c.f. [35] for a rigorous treatment of this technical assumption to ensure that the generalized Hopf formula, originally only defining as a lower minimax viscosity solution, to be also an upper minimax viscosity solution in the non-convex case.) This formula comes intuitively from the fact that the integral curves of the Hamiltonian vector field i.e. the bi-characteristics in the phase space, project to give straight line characteristics in x-space. Tracing the definition of the function φ as the lower value provides us the following reachability indicator function for (t,x) given by φ:

$$\mu(t,x) := \chi_{\mathbb{R}^+}\left(\varphi\left(T-t, e^{-tM}x\right)\right),$$

where $\chi_{\mathbb{R}^+}$ is the indicator function of $\mathbb{R}^+ := \{x \geq 0\}$ This function μ returns the value 1 when the initial state x can reach the zero level set J at or before time t following the dynamics described by (2.1), otherwise it will return 0.

Moreover, the control parameter $a(t)$ at (t,x) can also be obtained from this generalized Hopf formula. In fact, when the function φ is differentiable at (t,z), we have that

$$\nabla_z \varphi(t,z) = S_0(t,z) := \text{argmin}_{p \in \mathbb{R}^d} \left\{ J^*(p) + \int_0^t H(s,p)ds - \langle z, p\rangle \right\}. \tag{2.8}$$

(Notice that, in this case, the set $S_0(t,z)$ is a singleton.) Now gazing at the definition of $H(t,p)$ as in (2.5) and using the fact that $H(s,p)$ is of homogeneous degree 1 in our case when $L = 0$, we directly obtain that

$$\begin{aligned}
& \langle \nabla_z \varphi(t,z), \nabla_z H(t, \nabla_z \varphi(t,z))\rangle \\
= & \min_{b \in D(T-t)} \max_{a \in C(T-t)} \left\{ \left\langle [e^{-(T-t)M} N_C(T-t)]a + [e^{-(T-t)M} N_D(T-t)]b, \nabla_z \varphi(t,z) \right\rangle \right\} \\
= & \left\langle [e^{-(T-t)M} N_C(T-t)]a(T-t) + [e^{-(T-t)M} N_D(T-t)]b(T-t), \nabla_z \varphi(t,z) \right\rangle.
\end{aligned}$$

Therefore we have the following formula that $a(t)$ and $b(t)$ shall satisfy at any given time t:

$$\left[-e^{-tM} N_C(t)\right] a(t) + \left[-e^{-tM} N_D(t)\right] b(t) = \nabla_p H\left(\nabla_z \varphi\left(T-t, e^{-tM}x\right), T-t\right),$$

which shall render us the following control parameter $a(t)$:

$$a(t) = \frac{\mathbf{Q}_C(T-t)[-e^{-tM} N_C(t)]^* \nabla_z \varphi\left(T-t, e^{-tM}x\right)}{\|[-e^{-tM} N_C(t)]^* \nabla_z \varphi\left(T-t, e^{-tM}x\right)\|_{\mathbf{Q}_C(T-t)}} + \mathbf{a}_C(T-t).$$

It is similar to obtain $b(t)$, but it is not our purpose to obtain how the error behaves, and therefore unnecessary to compute $b(t)$.

In what follows, we shall apply coordinate descent algorithms, which is different from what we did in [6, 5], for solving the minimization problem (2.7). This gives us a very efficient algorithm for solving (2.5).

3 Optimization Methods and Minimization Techniques

As in [6, 5], we suggest to calculate the solution to (2.5) by directly carrying out the minimization process to obtain (2.7). To evaluate a value of φ at the point (t, z), we solve the d-dimensional minimization problem in (2.7). This way, all the values φ at different points (t, z) are decoupled and therefore they can be implemented in parallel without the necessity of any communication. The minimization problem of dimension d is of a low complexity. In this section, we suggest two methods to directly perform the minimization: the coordinate descent method and the ADMM/split Bregman algorithm.

3.1 Coordinate descent

With a long history in optimization since the 1950s, e.g. in [19, 41, 17, 26, 30, 38], coordinate descent algorithms are iterative optimization methods that update only 1 coordinate or 1 block of coordinates in the variable x at each iteration while fixing the others. Although seemingly unsophisticated, they enjoy major advantages in that the coordinate update has a per-iteration complexity much lower than the full update and that the subproblem dimension is independent of the original problem. These very favorable features have brought forth a resurgence of this method for solving big-data optimization problems in very high dimensions. Moreover, from the theoretical analysis for the smooth case, we can see that the coordinate-wise gradient descent step-size can be chosen considerably larger than we can take in the full gradient descent method.

In this work, we propose to use the coordinate descent algorithm in the minimization process of our non-smooth, possibly non-convex, function in (2.7). It is well known since [41] that, for a general non-smooth non-separable function, coordinate descent may get stuck at a non-stationary point where the function is non-smooth, and may converge to a local minimum in non-convex problems. Therefore, there are a lot of variations in the coordinate descent algorithms suitable for different cases. For instance, for a general smooth but nonconvex functions, a proximal term can be added at each step of the coordinate descent to promote stability [44, 45]. The adding of a proximal term is important because a general nonconvex functions can be very pathological and the algorithm can behave very wildly in this case. Theoretically, the proximal or proximal-gradient coordinate descent methods converge to a stationary point as long as the objective functional is coordinate-wise prox-regular (i.e., becomes convex after adding a sufficiently large proximal term).

Fortunately, our objective function is quite special. It has the form $F_1(p)+F_2(p)-F_3(p)$ (cf. problem (2.7)) where $F_1(p)$ is a smooth convex function, and $F_2(p)$ and $F_3(p)$ are convex homogeneous functions coming from l^2 norms. Although it is still non-separable, non-smooth, and non-convex, our function provides a special structure that allows us to use the standard coordinate descent algorithm in most practical cases.

In the case when our error set $D = \{0\}$, we actually have $F_3 = 0$, and therefore we encounter a convex but non-smooth function. In general, coordinate descent converges in this case, but it is not guaranteed that the limit point is a minimizer where the function is non-smooth. Nonetheless, thanks to the fact that the l^2 norm has only one non-smooth point, the origin, if the algorithm converges to any point other than the origin, we are in good shape because the function is smooth there and therefore we are at a global minimum. The only case when we need to be cautious is when the limit is the origin, but we can directly check if it is optimal at the origin; if not, we restart our algorithm from a different initial guess.

The case where our error set $D \neq \{0\}$ is more difficult since the objective function in (2.7) is non-smooth. In this case, the coordinate descent algorithm does not guarantee convergence to a stationary

point in general. A proper solution would be to apply variable splitting or operator splitting, which will promote stability and guarantee convergence. However, once again, our case is very special in that the origin is the only non-smooth point: whenever our algorithm converges to any point away from the origin, it has arrived at a stationary point. Considering the fact that adding a proximal map to the coordinate descent subproblem adds computational effort and that we use multiple initial guesses in the non-convex case anyway (because even if the algorithm converges to a stationary point, it is not necessarily a minimum), we instead use just the standard coordinate descent algorithm with multiple initial guesses to overcome the possibly pathological behaviour in the non-convex case. Luckily, in our numerical examples, we never observe non-convergence of the algorithm, and multiple initial guesses successfully help us avoid non-optimal stationary points.

3.1.1 The coordinate-descent algorithm

In what follows, we suggest the following coordinate descent algorithm to solve the minimization problem (2.7), which we will use in our numerical experiments:

Algorithm 1. *Take an initial guess of the Lipschitz constant L, and set count $:= 0$. Initialize $j_1 := 1$ and a parameter $\alpha := 1/L$. For $k = 1, ..., M$, do:*

***1**:*

$$\begin{cases} p_i^{k+1} = p_i^k - \alpha \left(\partial_i J^*(p^{k+1}) + \int_0^t \partial_i H(s, p^{k+1}) ds - z_i \right) & \text{if } i = j_k, \\ p_i^{k+1} = p_i^k & \text{otherwise.} \end{cases}$$

***2**:*

$$j_{k+1} := j_k + 1.$$

If $j_{k+1} = d + 1$, then reset $j_{k+1} = 1$.

***3**: If $|p_i^{k+1} - p_i^k| > \varepsilon$, then set count $:= 0$. If $k = M$, then reset $k := 0$ and set $\alpha := \alpha/2$, (i.e. let $L := 2L$.)*

***4**: If $|p^{k+1} - p^k| < \varepsilon$, set count $:=$ count $+ 1$.*

***5**: If count $= d$, stop.*

Return $p_{\text{final}} = p^{k+1}$.

We shall make the two remarks:

1. Step 3 is very crucial because sometimes the coordinate-wise Lipschitz constant is estimated to be too large, and yet we do not wish to spend time evaluating these constants prior to the calculation. Therefore we update our estimate when the method does not converge (even in the convex case).

2. In case the reader is still concerned with possible non-convergence in the non-convex case, we suggest to use the proximal-gradient coordinate descent method, which guarantees convergence to a stationary point, since our functional is coordinate-wise prox-regular, e.g. see [44, 45] for more details. However as previously explained, more computational effort is needed and it might not be worth the effort with our special type of function, especially when we take multiple initial guesses to avoid non-optimal stationary points.

3.1.2 Numerical differentiation and integration

In this subsection, we focus on the evaluation of the the function value and its derivative in **Algorithm 1**. We suggest to compute both the function value and its derivatives by numerical approximations.

For the sake of exposition, let us denote

$$\mathcal{F}(p) := J^*(p) + \int_0^t H(s,p)ds - \langle z, p\rangle \,.$$

To compute the function value $\mathcal{F}(p)$ numerically, we suggest the following standard rectangular quadrature rule to approximate the integral:

$$\mathcal{F}(p) \approx J^*(p) + \sum_i H(p, i\Delta s)\Delta s - \langle z, p\rangle \,. \tag{3.1}$$

with a given choice of Δs. Meanwhile, we suggest approximating the partial derivative $\partial_i \mathcal{F}(p)$ by a finite difference:

$$\partial_i \mathcal{F}(p) \approx \frac{\mathcal{F}(p + \sigma e_i) - \mathcal{F}(p)}{\sigma} \tag{3.2}$$

with a given choice of σ. By using numerical differentiation, we have the advantage of not necessarily handling tedious analytic computations of the derivative of Hamiltonian which might be singular at times. Also, we only have two evaluations of the function value per iteration. However, by performing numerical approximations, either differentiation or integration, we are bound to introduce numerical errors. These errors introduced by numerical approximation can be effectively controlled by choosing appropriate sizes of Δs and σ.

3.2 ADMM/split-Bregman splitting

An optimization algorithm involving splitting schemes might compute faster than a coordinate-descent algorithm in some cases, especially when the Hamiltonian is of some very special form: homogeneous of degree 1. For the sake of completeness and as an alternative approach, therefore we also propose the ADMM/split Bregman algorithm (c.f. [15, 16, 18]) for the optimization procedure, where proximal maps are encountered. This serves as another choice for some special problems when we observe ADMM to be faster than the coordinate descent algorithm.

As explained in our previous work [5], ADMM applied to a general non-smooth non-convex optimization problem may fail due to non-convexity. But it works very well for some practical nonconvex problems, e.g. in matrix completion [36, 37, 46, 47] and phase retrieval [43]. Convergence of ADMM is now known in [42] for either (1) a convex (possibly non-smooth) functional in the second step, or (2) a non-convex functional with some additional regularity assumptions (e.g. strict prox-regularity, piecewise affine, etc.) in the second step. Interested readers may refer to [42] and the references therein for a more thorough discussion of the issue. To avoid any complications caused by non-convexity, we follow our previous work in [5] to minimize a convex functional in Step 2. This way ADMM is guaranteed to converge.

3.2.1 The algorithm

Before we get to the split Bregman algorithm, let us first consider the following two homogeneous degree 1 convex functionals:

$$\begin{aligned}
\Psi_1(t,p) &:= \sqrt{\langle p, \mathbf{Q}_C(T-t)p\rangle} + \langle p, \mathbf{a}_C(T-t)\rangle \\
\Psi_2(t,p) &:= \sqrt{\langle p, \mathbf{Q}_D(T-t)p\rangle} - \langle p, \mathbf{a}_D(T-t)\rangle
\end{aligned}$$

and

$$\begin{aligned} R_1(t) &= [-e^{-(T-t)M} N_C(T-t)]^*, \\ R_2(t) &= [-e^{-(T-t)M} N_D(T-t)]^*. \end{aligned}$$

where $*$ denotes the matrix transpose. Then we get that $H(t,p)$ can actually be written as a difference of the two functionals

$$H(t,p) = \Psi_1(t, R_1(t)p) - \Psi_2(t, R_2(t)p).$$

With this notation in hand, we are ready to propose our optimization ADMM/split-Bregman algorithm for solving (2.7), which now aims to minimize the following functional:

$$\mathcal{J}(p) := J^*(p) + \int_0^t \left\{\Psi_1\left(s, R_1(s)\, p\right) - \Psi_2\left(s, R_2(s)\, p\right)\right\} ds - \langle z, p\rangle.$$

This is equivalent to solving the following minimization problem

$$\begin{aligned} &\min_p \left\{ J^*(p) + \int_0^t \left\{\Psi_1\left(s, \gamma_1(s)\right) - \Psi_2\left(s, \gamma_2(s)\right)\right\} ds - \langle z, p\rangle \right\} \\ &\text{subject to} \quad R_1(s)\, p = \gamma_1(s)\,,\ R_2(s)\, p = \gamma_2(s) \text{ for } 0 < s < t\,. \end{aligned}$$

Therefore we introduce the following augmented Lagrangian functional for a given $(z,t) \in \mathbb{R}^d \times (0,T)$:

$$\begin{aligned} &\mathcal{L}^1_{\lambda_1,\lambda_2,\gamma_1(\cdot),\gamma_2(\cdot)}(v, \gamma(\cdot)) \\ := \quad & J^*(v) - \langle z, v\rangle + \int_0^t \left\{\Psi_1\left(s, \gamma_1(s)\right) - \Psi_2\left(s, \gamma_2(s)\right)\right\} ds \\ & + \frac{\rho_1}{2} \int_0^t \left|\left|\lambda_1(s) - \gamma_1(s) + R_1(s)v\right|\right|^2 ds + \frac{\rho_2}{2} \int_0^t \left|\left|\lambda_2(s) - \gamma_2(s) + R_2(s)v\right|\right|^2 ds \end{aligned}$$

Now we present our splitting algorithm:

Algorithm 2. *For $n = 1, 2, ...,$ do:*

***1**: For all $s \in (0,t)$, compute*

$$\begin{aligned} \gamma_1^{k+1}(s) &= \operatorname{argmin}_{l \in \mathbb{R}^d} \left\{ \Psi_1\left(s, l\right) + \frac{\rho}{2} \left|\left|\lambda^k(s) - l + R_1(s) v^k\right|\right|^2 \right\}, \\ \gamma_2^{k+1}(s) &\in \operatorname{argmin}_{l \in \mathbb{R}^d} \left\{ -\Psi_2\left(s, l\right) + \frac{\rho}{2} \left|\left|\lambda^k(s) - l + R_2(s) v^k\right|\right|^2 \right\}. \end{aligned}$$

(The computation of γ_1^{k+1} can be done directly by applying the shrink operators, and that of γ_2^{k+1} can be computed using the stretch operators. They will be briefly described in the next subsection; see also [6, 5] for more details.)

***2**: Compute*

$$\begin{aligned} v^{k+1} &= \operatorname{argmin}_{v \in \mathbb{R}^d} \left\{ J^*(v) - \langle z, v\rangle + \frac{\rho_1}{2} \int_0^t \left|\left|\lambda^k(s) - \gamma_1^{k+1}(s) + R_1(s)v\right|\right|^2 ds \right. \\ &\qquad \left. + \frac{\rho_2}{2} \int_0^t \left|\left|\lambda^k(s) - \gamma_2^{k+1}(s) + R_2(s)v\right|\right|^2 ds \right\}. \end{aligned}$$

i.e. v^{k+1} solves

$$\begin{aligned} &\left[\partial J^* + \rho_1 \int_0^t R_1^*(s) R_1(s) ds + \rho_2 \int_0^t R_2^*(s) R_2(s) ds\right] v \\ = \quad & z + \rho_1 \int_0^t R_1^*(s)(\gamma_1^{k+1}(s) - \lambda_1^k(s)) ds + \rho_2 \int_0^t R_2^*(s)(\gamma_2^{k+1}(s) - \lambda_2^k(s)) ds \end{aligned}$$

3: For all $s \in (0,t)$,

$$\begin{aligned}\lambda_1^{k+1}(s) &= \lambda_1^k(s) - \gamma_1^{k+1}(s) + R_1(s)v^{k+1}\,,\\ \lambda_2^{k+1}(s) &= \lambda_2^k(s) - \gamma_2^{k+1}(s) + R_2(s)v^{k+1}\,.\end{aligned}$$

4: If $|v^{k+1} - v^k| < \varepsilon$, stop.

Return $v_{\text{final}} = v^{k+1}$.

We have the two following remarks:

1. First, this algorithm is similar to its time-independent counterpart suggested in our previous work [6, 5], but this algorithm is now more involved since we are now dealing with the time-dependent case. This is most apparent when we need to introduce curves $\gamma_1(\cdot)$ and $\gamma_2(\cdot)$ in our splitting scheme, instead of just vectors in $\mathbb{R}^d$. This slows down the algorithm considerably when T becomes large. However the operations in running **Algorithm 2** involves only explicit and easy-to-compute proximal maps (that we will discuss in the next section), it is difficult to say which algorithm, **Algorithm 1** or **Algorithm 2**, performs better in a particular problem. In fact different choices of d, T as well as the hardware structure of the computer may affect the speed of each algorithm and therefore it might be predictable that either **Algorithm 1** or **Algorithm 2** may have an edge in each case. However, we observe in our numerical experiments that **Algorithm 1** seems to work faster in general under our implementation in Section 4. Therefore for the sake of simplicity, we only present results from **Algorithm 1**. Nonetheless, readers are strongly encouraged to try both algorithms on their own computer and choose the faster one for practical purposes and real-life problem solving.

2. The evaluation of the sub-steps in **Algorithm 2** involves both optimization of subproblems and evaluation of integrals. The subproblems will be discussed in the next subsection, while integration can be done numerically as suggested in Section 3.1.2.

3.2.2 Proximal maps, Shrink and Stretch operators

We notice that in both substeps of **Algorithm 2**, we need to compute the following proximal map [29],

$$(I + \alpha\partial f)^{-1}(x) := \text{argmin}_{v\in\mathbb{R}^d}\left\{\alpha f(v) + \frac{1}{2}||v-x||^2\right\}$$

where f is either a convex functional or a homogeneous functional of degree 1 which can be either convex or concave. There is a large library of proximal maps that are known explicitly, e,g. the proximal map of the quadratic functions $f = \frac{1}{2}\langle Ax, x\rangle + \langle b, x\rangle + c$ is $(I+\partial f)^{-1}(x) = (I+A)^{-1}(x-b)$ when -1 is not an eignvalue of A. Other proximal maps can be again obtained by splitting, Newton's Method and Moreau decomposition [29]. A special class of proximal maps that are extensively studied is the case when $f = \pm\Phi$, where Φ is a homogeneous convex functional of degree 1.

Inspired by compressed sensing algorithms, e.g. in [48], we recall the following shrink operators for proximal maps of the positive homogeneous (of degree 1) convex functional Φ. In fact, we notice that Φ is actually the gauge functional of its Wulff set W (see for instance [33, 34, 13, 20] for its definition and more details) as:

$$\Phi(v) = \sup_{p\in W}\langle v, p\rangle\,.$$

As in [6, 5] we recall the definition of the shrink operator of the positive homogeneous (of degree 1) convex functional Φ as

$$\text{shrink}_\Phi(x,\alpha) := (I + \alpha\partial\Phi)^{-1}(x)$$

for any $\alpha > 0$. By the Fenchel-Legendre duality [20, 34] and the Moreau decomposition [29] of a general convex function f, we have, for any $\alpha > 0$,

$$(I + \alpha\partial f)^{-1}(x) + \alpha(I + \alpha^{-1}\partial f^*)^{-1}(x/\alpha) = x\,, \tag{3.3}$$

We get that the proximal map of the positive homogeneous (of degree 1) convex functional Φ is

$$\text{shrink}_\Phi(x, \alpha)(x) = (I + \alpha\partial\Phi)^{-1}(x) = x - \text{Proj}_{\alpha W}(x)$$

for any $\alpha > 0$.

On the other hand, for the proximal maps of $-\Phi$ where Φ is a positive homogeneous (of degree 1) convex functional Φ, we rely on the stretch operators for the function evaluations, [6, 5]. Let us define, for a given non-empty compact set W, the furthest points of a point x to C as the following set-valued function

$$\text{Fur}_C(x) := \text{argmax}_{y\in C}\{\|x - y\|^2\}\,, \tag{3.4}$$

where argmax returns the set of maximizers. With this definition of the Fur_W operator in (3.4) for any compact set W, we can rewrite the above expression of 'proximal map' by the following formula:

$$(I - \alpha\partial\Phi)^{-1}(x) = x + \text{Fur}_{\alpha W}(-x)\,. \tag{3.5}$$

Inspired by the definitions of the shrink operators, we introduce the following stretch operators for proximal maps of a positively homogeneous of degree 1 convex functional Φ:

$$\text{stretch}_\Phi(x, \alpha) := x + \int_{\text{Fur}_{\alpha W}(-x)} v\, d\mathcal{H}_{\text{Fur}_{\alpha W}(-x)}(v)$$

where $\mathcal{H}_W$ is the Hausdorff measure of the set W. For more details and fast evaluations of the shrink and stretch operators, readers may refer to [6, 5] for more detailed discussions.

4 Numerical Experiments

In this section, we provide numerical experiments which compute viscosity solutions to HJ PDE with time-dependent Hamiltonian arisen from control system. For a given set of points (t, z) , we use **Algorithm 1** to compute (2.3). We set $M = 500$ and have a different initial guess of the Lipschitz constant L in each example. We evaluate (t, z) in a given set of grid points over some 2 dimensional cross-sections of the form $\{0\}^a \times [-3, 3] \times \{0\}^b \times [-3, 3] \times \{0\}^c$ where $a + b + c = d - 2$,. We choose our error tolerance in the coodinate descent iteration as $\varepsilon = 0.5 \times 10^{-7}$, which acts as our stopping criterion. The step-size in the numerical quadrature rule in (3.1) is set to $\Delta s = 0.02$, and the step-size for numerical differentiation in (3.2) is $\sigma = 0.01$. In all our examples, we set random initial starting points uniformly distributed in $[-10, 10]^d$. For a convex Hamiltonian, we make one initial guess, and for a non-convex Hamiltonian, we perform 20 independent trials of initial guesses to get rid of possible local minima or in places when the derivative of the viscosity solution does not exist. Our algorithm is implemented in C++ on an 1.7 GHz Intel Core i7-4650U CPU. In what follows, we present some examples.

Example 1 We consider a simple toy example which is in $d = 2$ to check the algorithm. We consider the initial value to be a function with zero level set as an ellipse enclosed by the equation $\langle x, Ax\rangle = 1$ where $A = \text{diag}(1, 25/4)$, i.e. our initial condition for the HJ PDE is $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$. The Hamiltonian is induced by the ODE system in (2.1) with

$$M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad N_C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad N_D = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

and $C(t) = B_1(0) \subset \mathbb{R}^2, D(t) = \{0\}$. The end-time is chosen to be $T = 0.7$ and the Lipschitz constant is chosen as $L = 5$. Figure 1 shows the zero set contours of the solutions $\varphi(t, z) = 0$ using our new

algorithm where $t = 0.1, 0.2...0.7$. A clear comparison is performed with our solution to the solution to Lax-Friedrichs scheme. Indeed a first order Lax-Friedrichs monotone scheme [31] is implemented with $\Delta t = 0.001$ and $\Delta x = 0.005$. We can see from the comparison that the two solutions almost coincide. The solution from the Lax-Friedrichs scheme even has some tiny defects introduced by the boundary of the domain, in that the level sets are a little bit undershot when they approach the boundary of the domain. The computational time per point over the grid points (t, z) where $t \in \{0.1, 0.2...0.7\}, z \in \{(-3+0.1p, -3+0.1q) : p, q = 0, ..., 60\} \subset [-3, 3]^2$ is $2.041 \times 10^{-4} s$ per point. Considering the fact that the algorithm can be implemented in an embarrassingly parallel fashion, this order of magnitude of the run-time per point is satisfying for fast computation of the solution to HJ PDE in a computer that allows parallel computing.

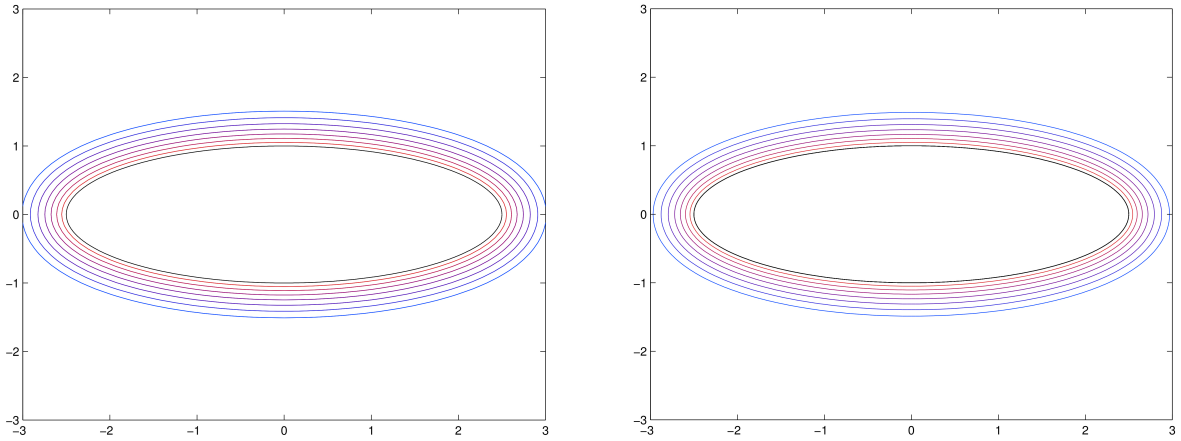

Figure 1: Level sets of the solution from control problems in Example 1 for $t \in \{0.1, 0.2...0.7\}$ with initial data $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$ ($A = \text{diag}(1, 25/4)$) over gridpoints $z \in \{(-3+0.1p, -3+0.1q) : p, q = 0, ..., 60\} \subset [-3, 3]^2$; left: Hopf-Lax formula, right: Lax-Friedrichs.

Example 2 Now we consider another example with a Hamiltonian introduced by two different control sets. We again consider the same initial value $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$. The Hamiltonian is induced by the ODE system in (2.1) with

$$M = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \quad N_C = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \quad N_D = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

and respectively the control sets are $C(t) = B_1(0) \subset \mathbb{R}^2, D(t) = \{0\}$ and

$$\begin{aligned} C(t) &= \{\langle x - \mathbf{a}_C, \mathbf{Q}_C^{-1}(x - \mathbf{a}_C)\rangle \leq 1\} \subset \mathbb{R}^2, \\ D(t) &= \{0\} \subset \mathbb{R}^2 \end{aligned}$$

where

$$\mathbf{Q}_C = \begin{pmatrix} 0.3 & 0.1 \\ 0.1 & 0.3 \end{pmatrix} \quad \mathbf{a}_C = \begin{pmatrix} -0.5 \\ -0.75 \end{pmatrix}$$
$$\mathbf{Q}_D = 0 \quad \mathbf{a}_D = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The end-time is again chosen to be $T = 0.7$ and the Lipschitz constant is chosen as $L = 5$. Figure 2 provides us the zero set contours of the solutions $\varphi(t, z) = 0$ with the respective Hamiltonians using our new algorithm where $t = 0.1, 0.2...0.7$, together with its Lax-Friedrichs counterpart [31] implemented with $\Delta t = 0.001$ and $\Delta x = 0.005$ for comparison. We can see that the two solutions match perfectly. The

computational time per point over the grid points (t, z) where $t \in \{0.1, 0.2...0.7\}, z \in \{(-3 + 0.1p, -3 + 0.1q) : p, q = 0, ..., 60\} \subset [-3, 3]^2$ for the respective Hamiltonians are $5.114 \times 10^{-4}s$ per point and $3.319 \times 10^{-4}s$ per point, both of which are of the same order of magnitude as in the previous example.

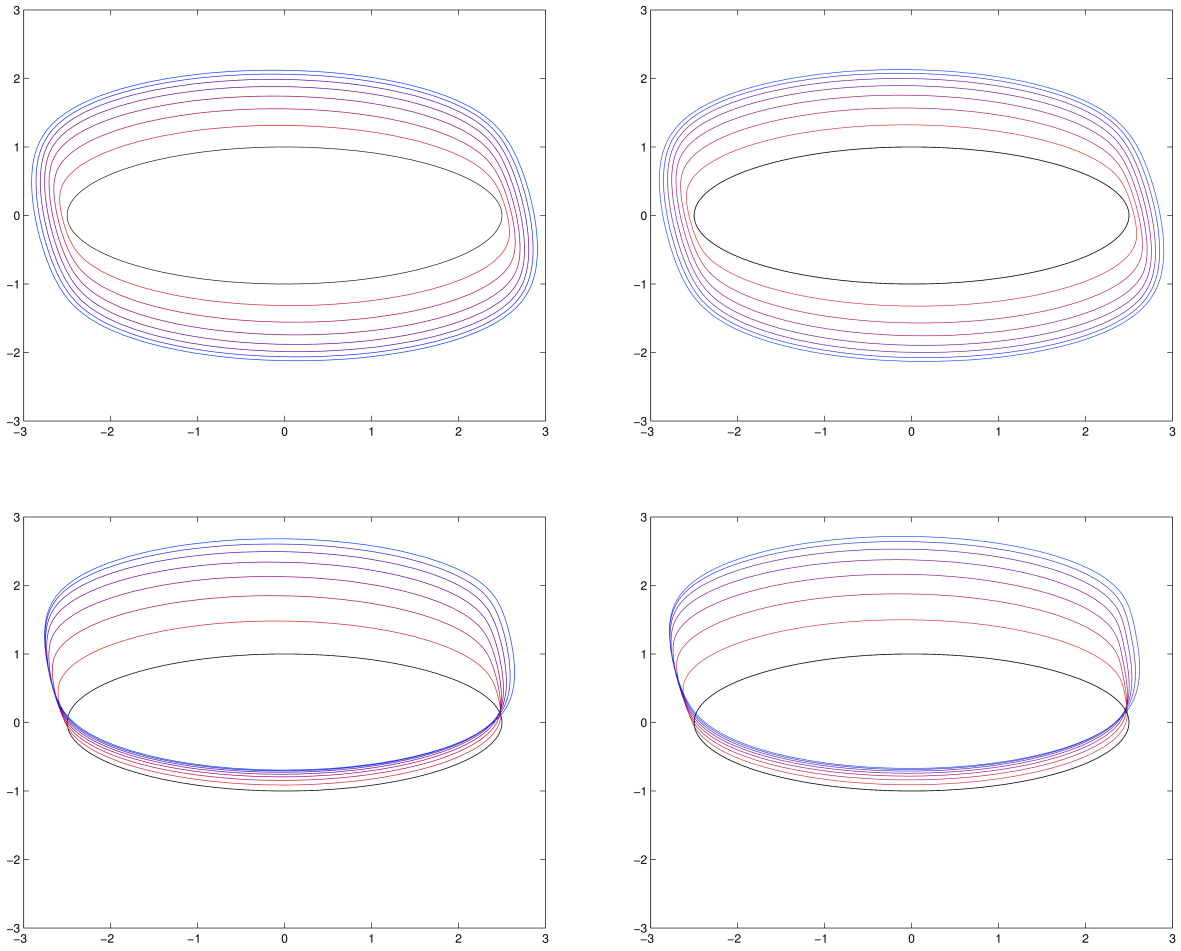

Figure 2: Level sets of the solution from control problems in Example 2 for $t \in \{0.1, 0.2...0.7\}$ with initial data $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$ ($A = \text{diag}(1, 25/4)$) over gridpoints $z \in \{(-3+0.1p, -3+0.1q) : p, q = 0, ..., 60\} \subset [-3, 3]^2$; left: Hopf-Lax formula, right: Lax-Friedrichs; top: $C(t) = B_1(0), D(t) = \{0\}$, bottom: $C(t) = \{\langle x - \mathbf{a}_C, \mathbf{Q}_C^{-1}(x - \mathbf{a}_C)\rangle \leq 1\}, D(t) = \{0\}$

Example 3 We now get to non-convex time-dependent Hamiltonians introduced by time-independent control sets. The initial value is the same as in the previous example. $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$. The Hamiltonian is induced by the ODE system in (2.1) with

$$M = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \quad N_C = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \quad N_D = \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix}$$

and the control sets are taken as

$$\begin{aligned} C(t) &= \{\langle x - \mathbf{a}_C, \mathbf{Q}_C^{-1}(x - \mathbf{a}_C)\rangle \leq 1\} \subset \mathbb{R}^2, \\ D(t) &= \{\langle x - \mathbf{a}_D, \mathbf{Q}_D^{-1}(x - \mathbf{a}_D)\rangle \leq 1\} \subset \mathbb{R}^2 \end{aligned}$$

where

$$\mathbf{Q}_C = \begin{pmatrix} 0.3 & 0.1 \\ 0.1 & 0.3 \end{pmatrix} \quad \mathbf{a}_C = \begin{pmatrix} -0.5 \\ -0.75 \end{pmatrix}$$
$$\mathbf{Q}_D = \begin{pmatrix} 0.4 & 0.2 \\ 0.2 & 0.4 \end{pmatrix} \quad \mathbf{a}_D = \begin{pmatrix} 0.5 \\ 0 \end{pmatrix}.$$

The end-time is also chosen to be $T = 0.7$ and the Lipschitz constant is again chosen as $L = 5$. Figure 3 contains the zero set contours of the solutions $\varphi(t, z) = 0$ with the respective Hamiltonians using our new algorithm where $t = 0.1, 0.2...0.7$, as well as its Lax-Friedrichs counterpart again with $\Delta t = 0.0005$ and $\Delta x = 0.0025$. In this case, we can observe that the solution from our newly-proposed algorithm, which uses directly the exact solution from the Hopf-Lax formula, actually captures the corners of the level sets much more sharply than the Lax-Friedrichs algorithm does. This goes along with the well-known property of the Lax-Friedrichs algorithm that it introduces numerical diffusion and thus a smoothing effect in the solution computed. The computational time per point over the grid points (t, z) where $t \in \{0.1, 0.2...0.7\}, z \in \{(-3+0.1p, -3+0.1q) \, : \, p, q = 0, ..., 60\} \subset [-3, 3]^2$ is $2.196 \times 10^{-4} s \times 20$ per point. It is fairly satisfactory when parallel computers are used to implement our method, especially because our Hamiltonian is now non-convex and is therefore more difficult to optimize.

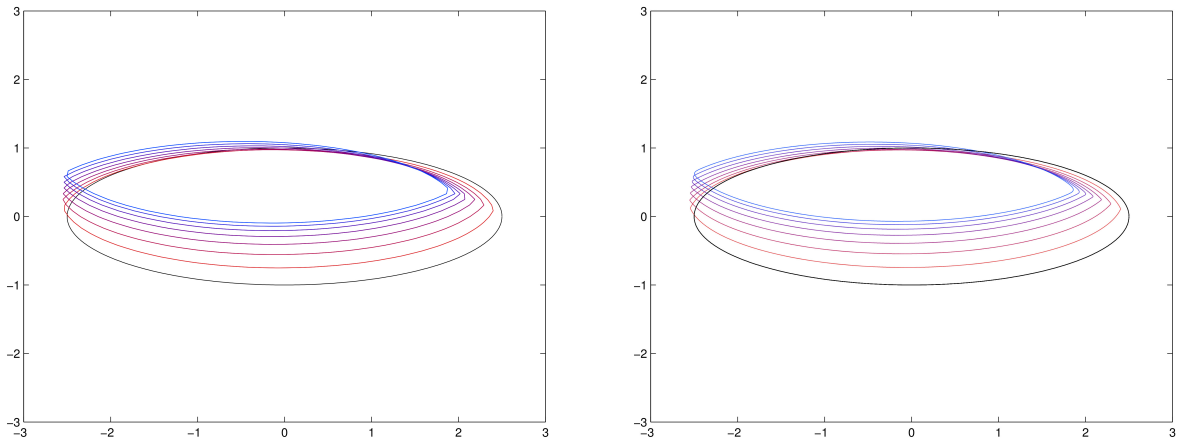

Figure 3: Level sets of the solution from control problems in Example 3 for $t \in \{0.1, 0.2...0.7\}$ with initial data $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$ ($A = \text{diag}(1, 25/4)$) over gridpoints $z \in \{(-3+0.1p, -3+0.1q) \, : \, p, q = 0, ..., 60\} \subset [-3, 3]^2$; left: Hopf-Lax formula, right: Lax-Friedrichs.

Example 4 In what follows, we get to several physical systems introduced by Dr. Hewer of China Lake Naval Station (private communication) with possibly non-convex time-dependent Hamiltonians and possibly time dependent control sets and higher dimensions. The initial value is the same as in the previous example in $d = 2$, and with a function with zero level set as an ellipse enclosed by the equation $\langle x, Ax\rangle = 1$ where $A = \text{diag}(1, 25/4, 0.5, ...0.5)$ when $d > 2$.

For $d = 2$, we first focus on a time-dependent non-convex Hamiltonian is induced by the ODE system in (2.1) with

$$M = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \quad N_C = \begin{pmatrix} 0 & 0 \\ 0.5 & 0 \end{pmatrix}, N_D = \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix}$$

and the time-independent control sets are taken as

$$\begin{aligned} C(t) &= \{\langle x - \mathbf{a}_C, \mathbf{Q}_C^{-1}(x - \mathbf{a}_C)\rangle \le 1\} \subset \mathbb{R}^2, \\ D(t) &= \{\langle x - \mathbf{a}_D, \mathbf{Q}_D^{-1}(x - \mathbf{a}_D)\rangle \le 1\} \subset \mathbb{R}^2 \end{aligned}$$

where

$$\mathbf{Q}_C = \begin{pmatrix} 0.3 & 0.1 \\ 0.1 & 0.3 \end{pmatrix} \quad \mathbf{a}_C = \begin{pmatrix} -0.5 \\ -0.75 \end{pmatrix}$$
$$\mathbf{Q}_D = \begin{pmatrix} 0.04 & 0.02 \\ 0.02 & 0.04 \end{pmatrix} \quad \mathbf{a}_D = \begin{pmatrix} 0.5 \\ 0 \end{pmatrix}$$

The end-time is still $T = 0.7$ and the Lipschitz constant as $L = 5$. Figure 4 shows the zero level set contours of the solutions $\varphi(t, z) = 0$ with the respective Hamiltonians using our new algorithm where $t = 0.1, 0.2...0.7$, as well as its comparison with the Lax-Friedrichs solution computed with $\Delta t = 0.0005$ and $\Delta x = 0.0025$. Again, we can compare the close-up of the respective solutions that the sharp angle created by non-convexity is captured more sharply with the Hopf-Lax formula, since it is free from numerical diffusion. The computational time per point over the grid points (x, t) where $x \in \{(-3 + 0.1p, -3 + 0.1q) : p, q = 0, ..., 60\} \subset [-3, 3]^2$ and $t \in \{0.1, 0.2...0.7\}$ is $4.860 \times 10^{-4} s \times 20$ per point.

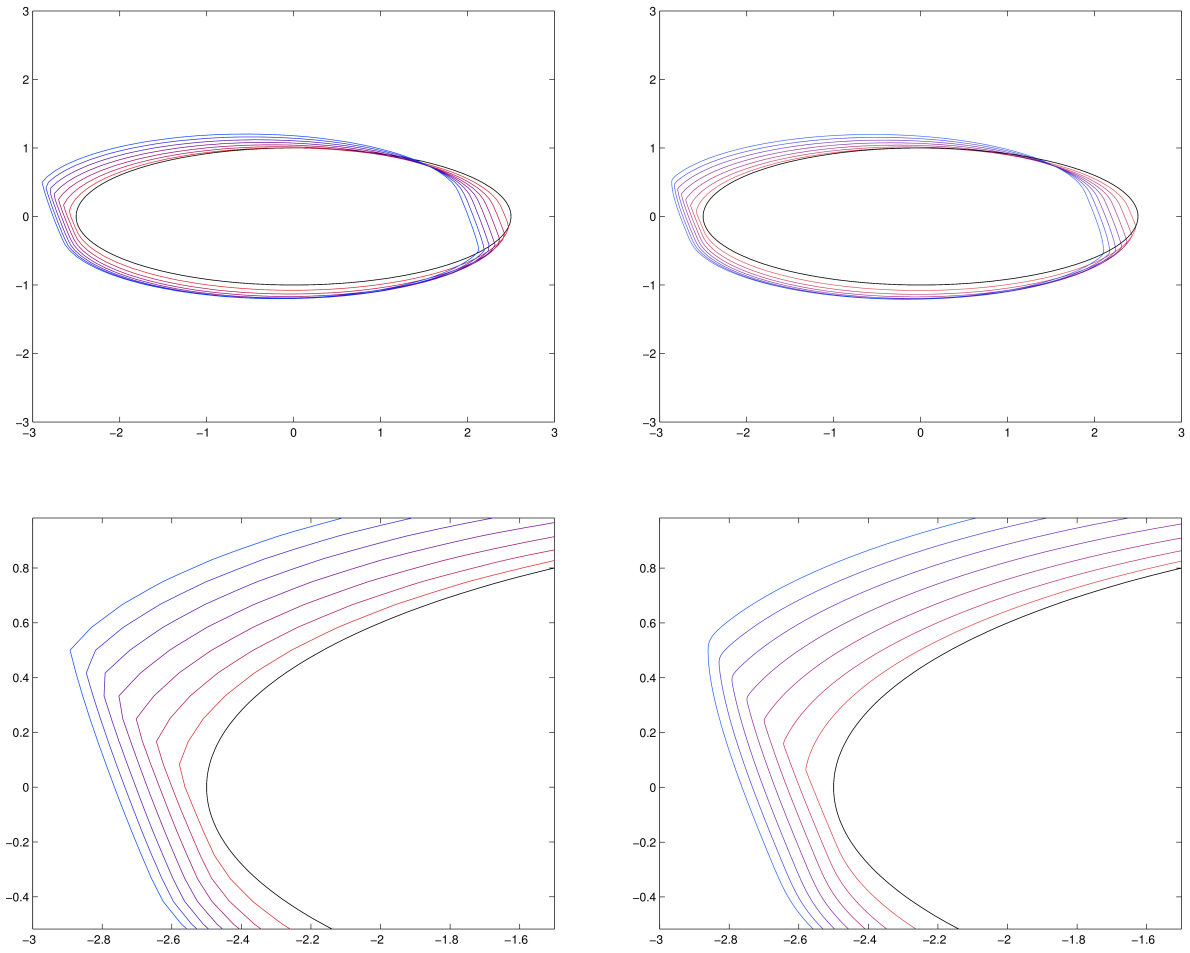

Figure 4: Level sets of the solution from the control problem in Example 4 with $d = 2$ for $t \in \{0.1, 0.2...0.7\}$ with initial data $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$ $(A = \text{diag}(1, 25/4, 0.5, ...0.5))$ over gridpoints $z \in \{(-3+0.1p, -3+0.1q) : p, q = 0, ..., 60\} \subset [-3, 3]^2$; left: Hopf-Lax formula, right: Lax-Friedrichs, top: normal size, bottom: close-up.

To better compare the figures, we also plot an overlap of the two solutions given by our newly suggested algorithm using the Hopf-Lax formula and that of Lax-Friedrichs. Figure 5 shows the overlap of the close-

ups of zero level set contours of the solutions (same as in Figure 4), with colored contours as Hopf-Lax solution and black-and-white as Lax-Friedrichs solution. One can see that they actually almost coincide. The Lax-Friedrichs solution clearly has some smoothing effect introduced by numerical diffision in the scheme.

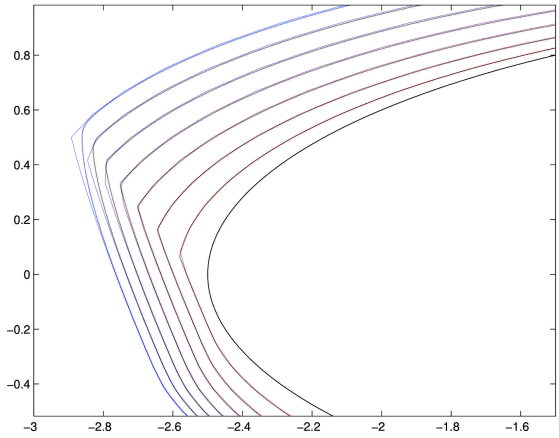

Figure 5: Close-up comparison of level sets of the solution from the control problem in Example 4 with $d = 2$ for $t \in \{0.1, 0.2...0.7\}$ with initial data $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$ ($A = \text{diag}(1, 25/4, 0.5, ...0.5)$) over gridpoints $z \in \{(-3 + 0.1p, -3 + 0.1q) \ : \ p, q = 0, ..., 60\} \subset [-3, 3]^2$; colored: Hopf-Lax formula, black-and-white: Lax-Friedrichs.

Then we get to a time-dependent non-convex Hamiltonian induced by the ODE system in (2.1) the same set of matrices M, N_C and N_D but with time dependent control sets:

$$\begin{aligned} C(t) &= \{\langle x - \mathbf{a}_C(t), \mathbf{Q}_C(t)^{-1}(x - \mathbf{a}_C(t))\rangle \leq 1\} \subset \mathbb{R}^2, \\ D(t) &= \{\langle x - \mathbf{a}_D(t), \mathbf{Q}_D(t)^{-1}(x - \mathbf{a}_D(t))\rangle \leq 1\} \subset \mathbb{R}^2 \end{aligned}$$

where

$$\mathbf{Q}_C(t) = e^{-2t}\begin{pmatrix} 0.3 & 0.1 \\ 0.1 & 0.3 \end{pmatrix} \quad \mathbf{a}_C(t) = \begin{pmatrix} -0.5 \\ -0.75 \end{pmatrix} - 0.1\begin{pmatrix} \cos(\frac{\pi t}{T}) \\ \sin(\frac{\pi t}{T}) \end{pmatrix}$$
$$\mathbf{Q}_D(t) = e^{2t}\begin{pmatrix} 0.04 & 0.02 \\ 0.02 & 0.04 \end{pmatrix} \quad \mathbf{a}_D(t) = \begin{pmatrix} 0.5 \\ 0 \end{pmatrix} - 0.1\begin{pmatrix} \cos(\frac{2\pi t}{T}) \\ \sin(\frac{2\pi t}{T}) \end{pmatrix}$$

The end-time is again $T = 0.7$ and the Lipschitz constant $L = 5$. The zero set contours of the solutions $\varphi(t, z) = 0$ are given in Figure 6 using our new algorithm where $t = 0.1, 0.2...0.7$, and its comparison with Lax-Friedrichs is also provided with $\Delta t = 0.0005$ and $\Delta x = 0.0025$. We can again see from the close-ups the sharp angle created by non-convexity from the Hopf-Lax formula. The computational time per point over the grid points (t, z) where $t \in \{0.1, 0.2...0.7\}, z \in \{(-3+0.1p, -3+0.1q) \ : \ p, q = 0, ..., 60\} \subset [-3, 3]^2$ is $2.360 \times 10^{-4}s \times 20$ per point. This order of magnitude is actually very satisfactory considering the fact that the Hamiltonian is increasingly non-convex with respect to time and the control sets are now time-dependent.

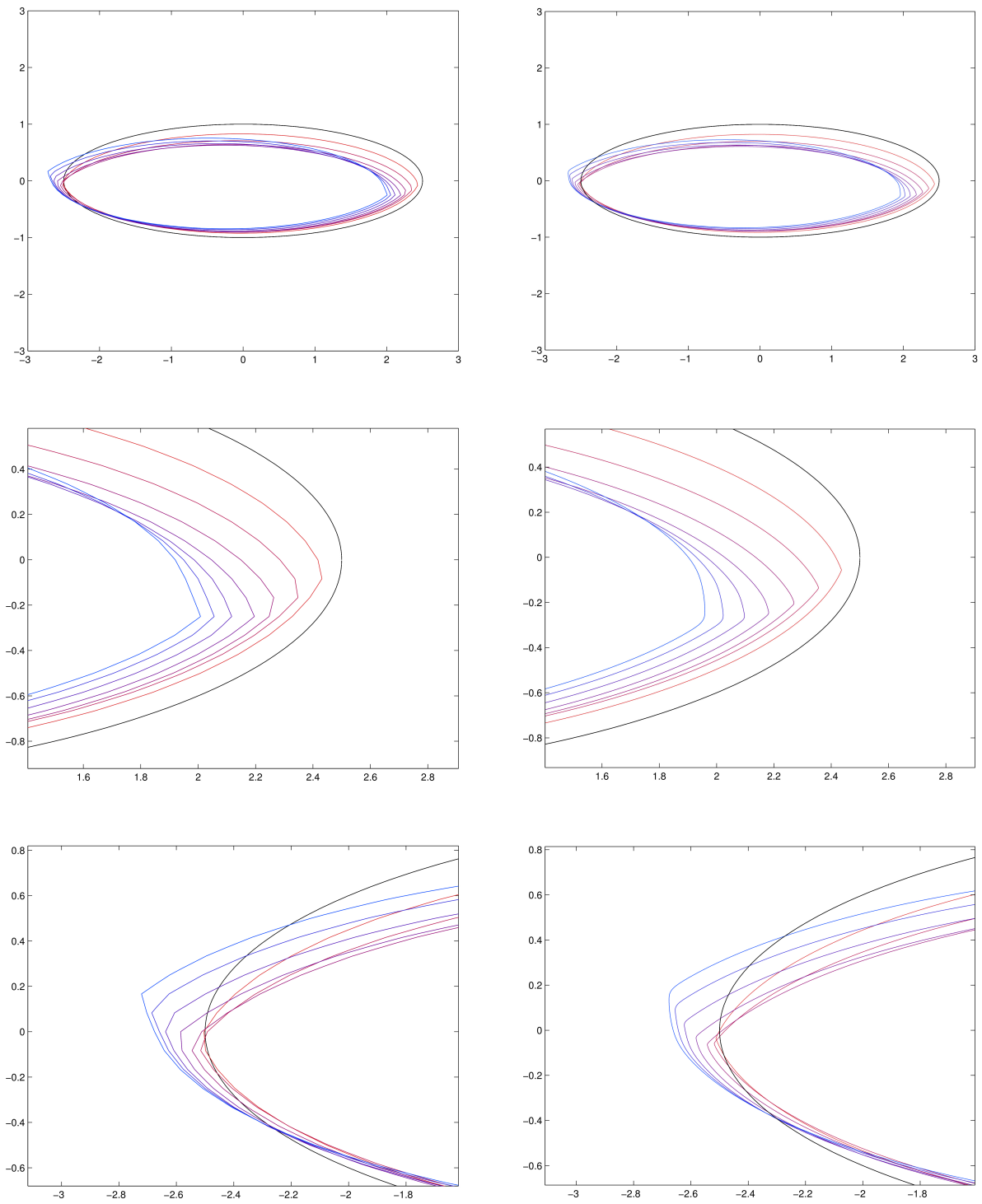

Figure 6: Level sets of the solution from the control problem in Example 4 with $d = 2$ for $t \in \{0.1, 0.2...0.7\}$ with initial data $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$ ($A = \mathrm{diag}(1, 25/4, 0.5, ...0.5)$) over gridpoints $z \in \{(-3+0.1p, -3+0.1q) : p, q = 0, ..., 60\} \subset [-3, 3]^2$; left: Hopf-Lax formula, right: Lax-Friedrichs, top: normal size, middle and bottom: close-ups.

To better compare the figures, we again plot an overlap of the two solutions given by our newly suggested algorithm using the Hopf-Lax formula and that of Lax-Friedrichs. Figure 7 shows the overlaps of the close-ups of the two zero level set contours of the solutions (same as in Figure 6), with colored contours as Hopf-Lax solutions and black-and-white as Lax-Friedrichs solutions. One can now see the more apparent smoothing effect possibly introduced by numerical diffusion in the Lax-Friedrichs solutions, but except that, they still match perfectly.

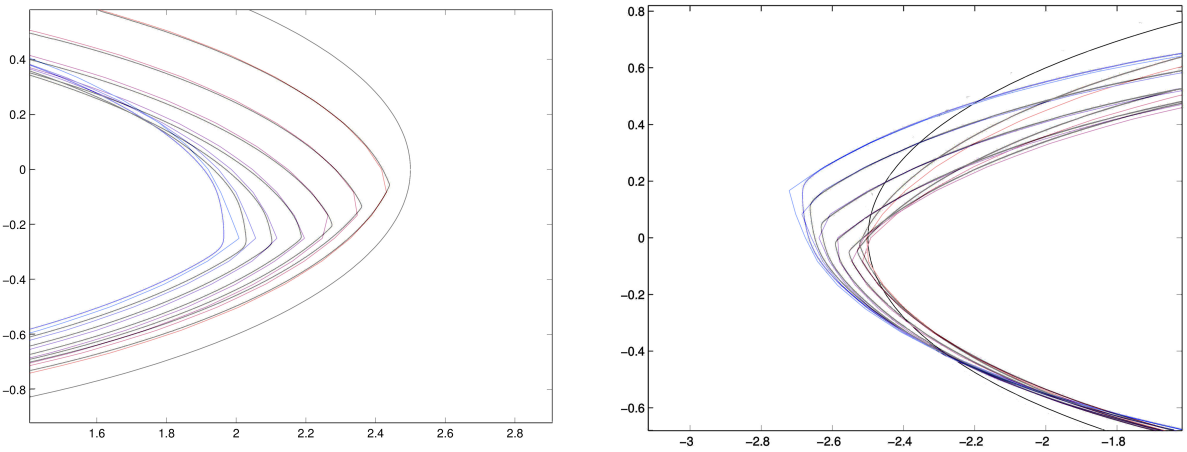

Figure 7: Close-up comparison of level sets of the solution from the control problem in Example 4 with $d = 2$ for $t \in \{0.1, 0.2...0.7\}$ with initial data $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$ ($A = \mathrm{diag}(1, 25/4, 0.5, ...0.5)$) over gridpoints $z \in \{(-3 + 0.1p, -3 + 0.1q) \,:\, p, q = 0, ..., 60\} \subset [-3, 3]^2$; colored: Hopf-Lax formula, black-and-white: Lax-Friedrichs.

Next we get to a case with $d = 3$ with a Hamiltonian which is induced by the ODE system in (2.1) with

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -0.0136 \\ 0 & 0 & -2.0000 \end{pmatrix} \quad N_C = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 20 & 0 & 0 \end{pmatrix} \quad N_D = 0\,.$$

and the time-independent control sets are taken as

$$C(t) = \{\langle x - \mathbf{a}_C, \mathbf{Q}_C^{-1}(x - \mathbf{a}_C)\rangle \leq 1\} \subset \mathbb{R}^3, D(t) = \{0\}\,,$$

where

$$\mathbf{Q}_C = \begin{pmatrix} 0.3 & 0.2 & 0.1 \\ 0.2 & 0.3 & 0.2 \\ 0.1 & 0.2 & 0.3 \end{pmatrix} \quad \mathbf{a}_C = \begin{pmatrix} -0.5 \\ -0.2 \\ 0 \end{pmatrix}$$

The end-time is now chosen as $T = 0.5$ and the Lipschitz constant is still set to $L = 5$. Figure 8 shows the zero level set contours of the solutions $\varphi(t, z) = 0$ with the respective Hamiltonians along the two respective 2-dimensional cross-section $\{0\} \times [-3, 3]^2$ and $[-3, 3] \times \{0\} \times [-3, 3]$ using our new algorithm where $t = 0.1, 0.2...0.5$. The respective computational time per point over the grid points (t, z) where $t \in \{0.1, 0.2...0.5\}, z \in \{(0, -3 + 0.1p, -3 + 0.1q) \,:\, p, q = 0, ..., 60\} \subset \{0\} \times [-3, 3]^2$ and $t \in \{0.1, 0.2...0.5\}, x \in \{(-3+0.1p, 0, -3+0.1q) \,:\, p, q = 0, ..., 60\} \subset [-3, 3]\times\{0\}\times[-3, 3]$ are $6.997\times10^{-3}s$ per point and $9.376 \times 10^{-3}s$ per point, which is still satisfactory in a 3 dimensional case.

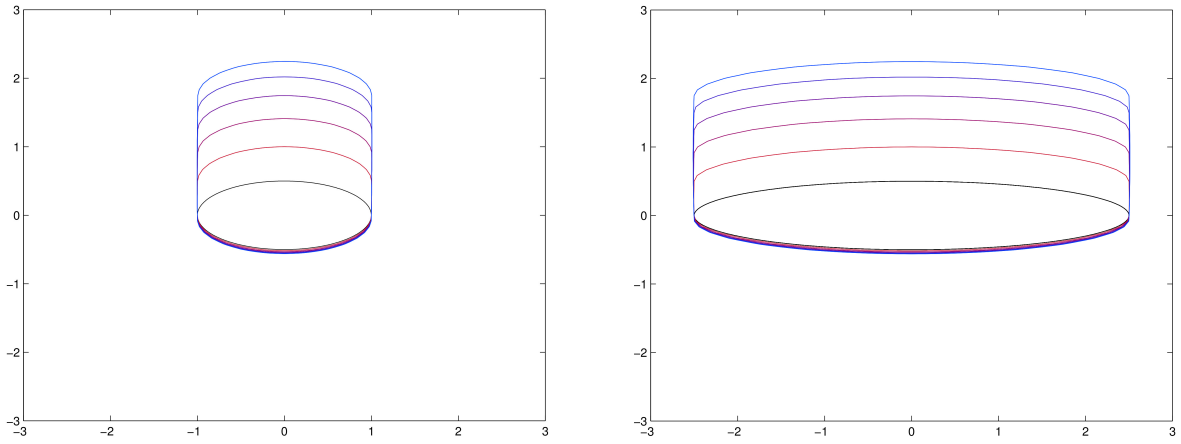

Figure 8: Level sets of the solution from a control problem in Example 4 with $d = 3$ for $t \in \{0.1, 0.2...0.5\}$ with initial data $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$ ($A = \mathrm{diag}(1, 25/4, 0.5, ...0.5)$); left: over gridpoints $z \in \{0\} \times [-3, 3]^2$, right: over gridpoints $z \in [-3, 3] \times \{0\} \times [-3, 3]$.

Example 5 Now we arrive at several physical systems described in [24] with possibly non-convex time-dependent Hamiltonian under possibly time dependent control sets in higher dimensions. The initial value is now again a function with zero level set as an ellipse enclosed by the equation $\langle x, Ax\rangle = 1$ where $A = \mathrm{diag}(1, 25/4, 0.5, ...0.5)$.

For $d = 6$, we consider Hamiltonian induced by the ODE system in (2.1) with

$$
\begin{aligned}
M &= 0.1 \times \begin{pmatrix} -20 & 0 & 0 & 0 & 0 & 0 \\ 0 & -25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.744 & -0.032 & 0 & -0.154 & -0.0052 & 1.54 \\ 0.337 & -1.12 & 0 & 0.249 & -0.1 & -5.2 \\ -0.02 & 0 & 0.0386 & -0.996 & -0.000295 & -0.117 \end{pmatrix} \\
N_C &= 0.1 \times \begin{pmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
N_D &= 0\,.
\end{aligned}
$$

and the control sets taken as

$$C(t) = \{\langle x - \mathbf{a}_C, \mathbf{Q}_C^{-1}(x - \mathbf{a}_C)\rangle \leq 1\} \subset \mathbb{R}^6, D(t) = \{0\}\,,$$

where

$$\mathbf{Q}_C = 0.1 \times \begin{pmatrix} 3 & 1 & 0 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 & 0 \\ 0 & 1 & 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 & 1 & 0 \\ 0 & 0 & 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & 0 & 1 & 3 \end{pmatrix} \quad \mathbf{a}_C = 0.1 \times \begin{pmatrix} -10 \\ -5 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The end-time is now chosen as $T = 0.5$ and the Lipschitz constant is now set as $L = 4$. Figure 9 presents the zero set contours of the solutions $\varphi(t, z) = 0$ along the two respective 2-dimensional cross-section $[-3, 3]^2 \times \{0\}^4$ using our new algorithm where $t = 0.1, 0.2...0.5$. The respective computational time per point over the grid points (t, z) where $t \in \{0.1, 0.2...0.5\}, z \in \{(-3 + 0.1p, -3 + 0.1q, 0, 0, 0, 0) : p, q = 0, ..., 60\} \subset \times[-3, 3]^2 \times \{0\}^4$, is $5.625 \times 10^{-2} s$ per point. This is actually excellent considering the fact that it is usually impossible to compute a time-dependent HJ PDE solution/behaviours of a control system in a 6 dimensional case effectively when conventional algorithms are used.

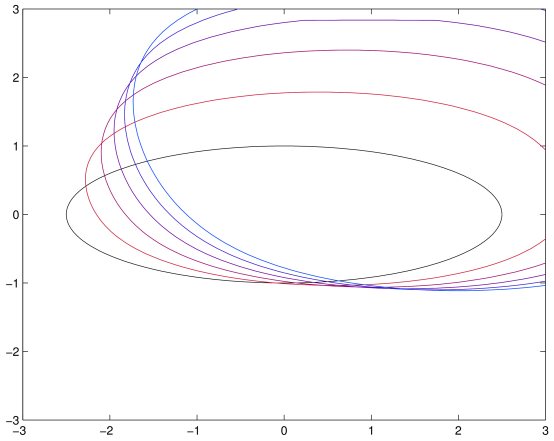

Figure 9: Level sets of the solution from the control problem in Example 5 with $d = 6$ for $t \in \{0.1, 0.2...0.5\}$ with initial data $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$ ($A = \text{diag}(1, 25/4, 0.5, ...0.5)$) and time-independent control sets $C(t) = \{\langle x - \mathbf{a}_C, \mathbf{Q}_C^{-1}(x - \mathbf{a}_C)\rangle \leq 1\}, D(t) = \{0\}$ over gridpoints $z \in \{(-3 + 0.1p, -3 + 0.1q, 0, 0, 0, 0) : p, q = 0, ..., 60\} \subset \times[-3, 3]^2 \times \{0\}^4$.

Next, we consider non-convex Hamiltonians induced by the ODE system in (2.1) with

$$
\begin{aligned}
M &= 0.1 \times \begin{pmatrix} -20 & 0 & 0 & 0 & 0 & 0 \\ 0 & -25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.744 & -0.032 & 0 & -0.154 & -0.0052 & 1.54 \\ 0.337 & -1.12 & 0 & 0.249 & -0.1 & -5.2 \\ -0.02 & 0 & 0.0386 & -0.996 & -0.000295 & -0.117 \end{pmatrix} \\
N_C &= 0.1 \times \begin{pmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
N_D &= 0.1 \times I_6 \, .
\end{aligned}
$$

and the time-dependent control sets taken as

$$
\begin{aligned}
C(t) &= \{\langle x - \mathbf{a}_C(t), \mathbf{Q}_C(t)^{-1}(x - \mathbf{a}_C(t))\rangle \leq 1\} \subset \mathbb{R}^6 \, , \\
D(t) &= \{\langle x - \mathbf{a}_D(t), \mathbf{Q}_D(t)^{-1}(x - \mathbf{a}_D(t))\rangle \leq 1\} \subset \mathbb{R}^6 \, ,
\end{aligned}
$$

where

$$\mathbf{Q}_C(t) = 0.1e^{-2t}\begin{pmatrix} 3 & 1 & 0 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 & 0 \\ 0 & 1 & 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 & 1 & 0 \\ 0 & 0 & 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & 0 & 1 & 3 \end{pmatrix} \quad \mathbf{a}_C(t) = 0.1\begin{pmatrix} -10 \\ -5 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} - 0.1\begin{pmatrix} \cos(\frac{\pi t}{T}) \\ \sin(\frac{\pi t}{T}) \end{pmatrix}$$

$$\mathbf{Q}_D(t) = 0.1e^{2t}\begin{pmatrix} 1 & 0.25 & 0 & 0 & 0 & 0 \\ 0.25 & 1 & 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 1 & 0.25 & 0 & 0 \\ 0 & 0 & 0.25 & 1 & 0.25 & 0 \\ 0 & 0 & 0 & 0.25 & 1 & 0.25 \\ 0 & 0 & 0 & 0 & 0.25 & 1 \end{pmatrix} \quad \mathbf{a}_D(t) = 0.1\begin{pmatrix} -10 \\ -5 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} - 0.1\begin{pmatrix} \cos(\frac{\pi t}{T}) \\ \sin(\frac{\pi t}{T}) \end{pmatrix}$$

The end-time is still chosen as $T = 0.5$ and the Lipschitz constant is again set as $L = 4$. Figure 10 gives the zero set contours of the solutions $\varphi(t, z) = 0$ along the two respective 2-dimensional cross-sections $[-3, 3]^2 \times \{0\}^4$ using our new algorithm where $t = 0.1, 0.2...0.5$. The respective computational time per point over the grid points (t, z) where $t \in \{0.1, 0.2...0.5\}, z \in \{(-3 + 0.1p, -3 + 0.1q, 0, 0, 0, 0) : p, q = 0, ..., 60\} \subset \times[-3, 3]^2 \times \{0\}^4$ is $7.862 \times 10^{-2}s \times 20$ per point. This is impressive considering the high dimension of the problem and the increasing non-convexity of the problem.

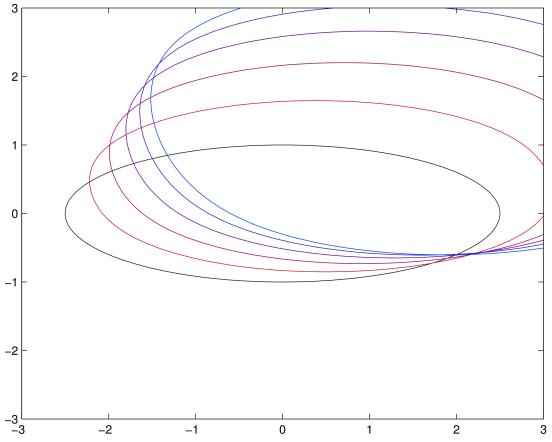

Figure 10: Level sets of the solution from the control problem in Example 5 with $d = 6$ for $t \in \{0.1, 0.2...0.5\}$ with initial data $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$ ($A = \text{diag}(1, 25/4, 0.5, ...0.5)$) with time-dependent control sets $C(t) = \{\langle x - \mathbf{a}_C(t), \mathbf{Q}_C(t)^{-1}(x - \mathbf{a}_C(t))\rangle \leq 1\}, D(t) = \{\langle x - \mathbf{a}_D(t), \mathbf{Q}_D(t)^{-1}(x - \mathbf{a}_D(t))\rangle \leq 1\}$over gridpoints $z \in \{(-3 + 0.1p, -3 + 0.1q, 0, 0, 0, 0) : p, q = 0, ..., 60\} \subset \times[-3, 3]^2 \times \{0\}^4$.

The last example is chosen with $d = 10$ and Hamiltonian induced by the ODE system in (2.1) with

$$M = 0.001 \times \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -62500 & -250 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2450000 & 0 & -130 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -10000 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 900 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -7260000 & 0 & -62500 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 40000 & 40000 & 0 & -40000 & -200 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$N_C = 0.001 \times \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$N_D = 0\,.$$

The control sets are now given as

$$C(t) = \{\langle x - \mathbf{a}_C, \mathbf{Q}_C^{-1}(x - \mathbf{a}_C)\rangle \leq 1\} \subset \mathbb{R}^{10}, D(t) = \{0\}\,,$$

where

$$\mathbf{Q}_C = 0.001 \times \begin{pmatrix} 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 \end{pmatrix} \quad \mathbf{a}_C = 0.001 \times \begin{pmatrix} -10 \\ -5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The end-time is again as $T = 0.5$ and the Lipschitz constant is now set to be $L = 2^7$. Figure 11 shows the zero set contours of the solutions $\varphi(t, z) = 0$ along the two respective 2-dimensional cross-section $[-3, 3]^2 \times \{0\}^8$ using our new algorithm where $t = 0.1, 0.2...0.5$. The respective computational time per point over the grid points (t, z) where $t \in \{0.1, 0.2...0.5\}, z \in \{(-3 + 0.1p, -3 + 0.1q, 0, 0, 0, 0, 0, 0, 0, 0) : p, q = 0, ..., 60\} \subset \times[-3, 3]^2 \times \{0\}^8$ is $2.995 \times 10^{-1}s$ per point, which is still good considering the high dimension of the problem.

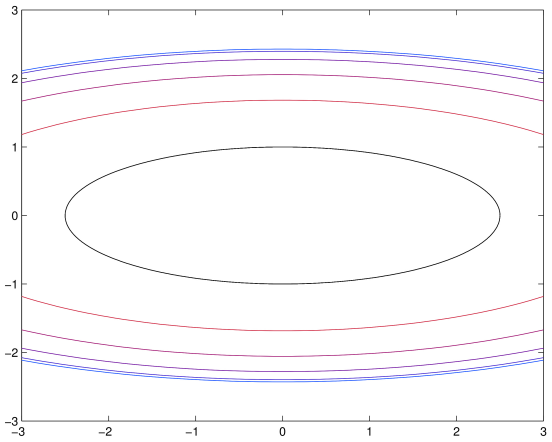

Figure 11: Level sets of the solution from the control problem in Example 5 with $d = 10$ for $t \in \{0.1, 0.2...0.5\}$ with initial data $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$ $(A = \mathrm{diag}(1, 25/4, 0.5, ...0.5))$ over gridpoints $z \in \{(-3 + 0.1p, -3 + 0.1q, 0, 0, 0, 0) : p, q = 0, ..., 60\} \subset \times[-3, 3]^2 \times \{0\}^8$.

Example 6 We now make an explicit comparsion for the computational effort of our newly proposed Hamilton-Jacobi solver and control problem using a graph of computational time per dimension. We again consider the same initial value $J(x) = \frac{1}{2}(\langle x, Ax\rangle - 1)$, and an example with a Hamiltonian introduced by two different control sets. Now for a given dimension d, the Hamiltonian is induced by the ODE system in (2.1) with

$$M = \begin{pmatrix} 1 & 1 & 0 & ... & 0 & 0 & 0 \\ 1 & 1 & 1 & ... & 0 & 0 & 0 \\ 0 & 1 & 1 & ... & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & ... & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & 1 & 1 & 0 \\ 0 & 0 & 0 & ... & 1 & 1 & 1 \\ 0 & 0 & 0 & ... & 0 & 1 & 1 \end{pmatrix} \quad N_C = \begin{pmatrix} 1 & 0.5 & 0 & ... & 0 & 0 & 0 \\ 0.5 & 1 & 0.5 & ... & 0 & 0 & 0 \\ 0 & 0.5 & 1 & ... & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & ... & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & 1 & 0.5 & 0 \\ 0 & 0 & 0 & ... & 0.5 & 1 & 0.5 \\ 0 & 0 & 0 & ... & 0 & 0.5 & 1 \end{pmatrix} \quad N_D = 0.1 \times I$$

and respectively the control sets is taken as

$$\begin{aligned} C(t) &= \{\langle x - \mathbf{a}_C, \mathbf{Q}_C^{-1}(x - \mathbf{a}_C)\rangle \leq 1\} \subset \mathbb{R}^d, \\ D(t) &= \{\langle x - \mathbf{a}_D, \mathbf{Q}_D^{-1}(x - \mathbf{a}_D)\rangle \leq 1\} \subset \mathbb{R}^d \end{aligned}$$

where

$$\mathbf{Q}_C = \begin{pmatrix} 0.3 & 0.1 & 0 & ... & 0 & 0 & 0 \\ 0.1 & 0.3 & 0.1 & ... & 0 & 0 & 0 \\ 0 & 0.1 & 0.3 & ... & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & ... & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & 0.3 & 0.1 & 0 \\ 0 & 0 & 0 & ... & 0.1 & 0.3 & 0.1 \\ 0 & 0 & 0 & ... & 0 & 0.1 & 0.3 \end{pmatrix} \quad \mathbf{a}_C = 0$$

$$\mathbf{Q}_D = 0.01 \times I \quad \mathbf{a}_D = 0$$

The Lipschitz constant is chosen as $L = 4$. Figure 12 shows the computational time per point over the grid points (t, z) where $t = 0.1, z \in \{(-3 + 0.1p, -3 + 0.1q) : p, q = 0, ..., 60\} \subset [-3, 3]^2$ with respect

to the dimension d where $d \in (0, 20] \bigcap \mathbb{N}$. We can see the mild growth of computational time with respect to the dimension of the problem.

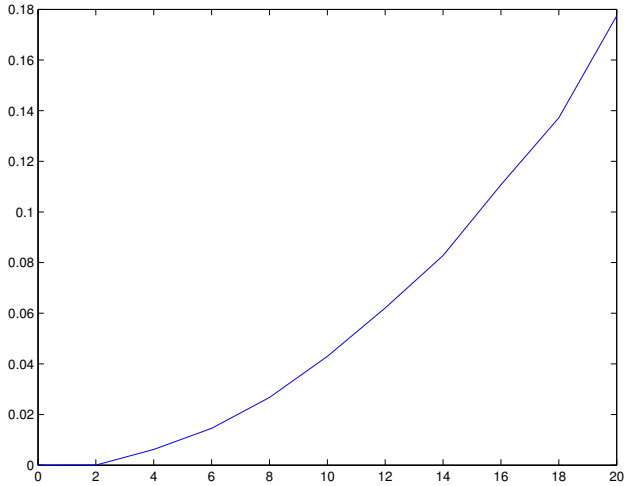

Figure 12: Computation time per point over the grid (t, z) with $t = 0.1, z \in \{(-3 + 0.1p, -3 + 0.1q) : p, q = 0, ..., 60\} \subset [-3, 3]^2$ with dimension $d \in (0, 20] \bigcap \mathbb{N}$ in Example 6.

5 Acknowledgements

We sincerely thank Dr. Gary Hewer and his colleagues (China Lake Naval Center) for providing help and guidance in practical optimal control and differential game problems. We also thank Prof. Jianliang Qian for reminding us of the literature [35] and the technical condition for the generalized Hopf formula.

References

[1] R. Bellman, *Adaptive Control Processes, a Guided Tour*, Princeton U. Press, (1961).

[2] R. Bellman, *Dynamic Programming*, Princeton U. Press, (1957).

[3] M.G. Crandall, P.-L. Lions, *Some Properties of Viscosity Solutions of Hamilton-Jacobi Equations*, Trans. AMS 282 (2), pp. 487-502, (1984).

[4] M.G. Crandall, P.-L. Lions, *Viscosity Solutions of Hamilton-Jacobi Equations*, Trans. AMS 277 (1), pp. 1-42, (1983).

[5] Y.T. Chow, J. Darbon, S. Osher, W. Yin, *Algorithm for Overcoming the Curse of Dimensionality for Certain Non-convex Hamilton-Jacobi Equations, Projections and Differential Games*, UCLA CAM report 16-27, (2016).

[6] J. Darbon, S. Osher, *Algorithms for Overcoming the Curse of Dimensionality for Certain Hamilton-Jacobi Equations Arising in Control Theory and Elsewhere*, preprint, UCLA CAM report 15-50, (2015).

[7] D. Davis, W. Yin, *A Three-Operator Splitting Scheme and its Optimization Applications*, preprint, UCLA CAM report 15-13, (2015).

[8] E.W. Dijkstra, *A Note on Two Problems in Connexion with Graphs*, Num. Math. 1, pp. 269-271, (1959).

[9] I.C. Dolcetta, *Representation of Solutions of Hamilton-Jacobi Equations*, Nonlinear Equations: Methods, Models and Applications, pp. 79-90, (2003).

[10] I. Elishakoff (ed.), Whys and Hows in Uncertainty Modelling, Springer-Verlag Wien 1999.

[11] L.C. Evans, Partial Differential Equations, Grad. Studies in Math. 19, AMS, (2010).

[12] L.C. Evans and P.E. Souganidis, *Differential Games and representation formulas for solutions of Hamilton-Jacobi Isaacs Equations*, Indiana U. Math. J. 38, pp. 773-797, (1984).

[13] M.P. Friedlander, I. Macedo, T.K. Pong, *Gauge optimization and duality*, SIAM Journal on Optimization 24 (4), pp 1999–2022, (2014).

[14] J. Friedman, T. Hastie, H. Hofling and R. Tibshirani, *Pathwise coordinate optimization*, Annals of Applied Statistics 1, pp. 302-332, 2007.

[15] D. Gabay, B. Mercier *A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximation*, Computers & Mathematics with Applications 2(1), pp.17-40, (1976).

[16] R. Glowinski, A. Marroco, *On the Approximation by Finite Elements of Order One, and Resolution, Penalisation-Duality for a Class of Nonlinear Dirichlet Problems*, ESAIM: Mathematical Modelling and Numerical Analysis - Mathematical Modelling and Numerical Analysis 9(R2), pp 41-76, (1975).

[17] L. Grippo, M. Sciandrone. *On the convergence of the block nonlinear Gauss–Seidel method under convex constraints.* Operations Research Letters, 26(3), pp 127–136, (2000).

[18] T. Goldstein, S. Osher, *The split Bregman method for L1-regularized problems*, SIAM Journal on Imaging Sciences, 2(2), pp 323–343, (2009).

[19] C. Hildreth. *A quadratic programming procedure.* Naval research logistics quarterly, 4(1), pp 79–85, (1957).

[20] J.-B. Hiriart-Urruty, C. Lemaréchal, Fundamentals of Convex Analysis, Grundlehren Text Editions, Springer, (2001).

[21] E. Hopf, *Generalized Solutions of Nonlinear Equations of the First Order*, J. Math. Mech. 14, pp. 951-973, (1965).

[22] M. B. Horowitz, A. Damle, J. W. Burdick, *Linear Hamilton Jacobi Bellman Equations in High Dimensions*, arXiv:1404.1089, (2014).

[23] W. Kang, L.C. Wilcox, *Mitigating the Curse of Dimensionality: Sparse Grid Characteristics Method for Optimal Feedback Control and HJB Equations*, preprint, arXiv:1507.04769, (2015).

[24] C.S. Kenney, R.B. Lepnik, Integration of the Differential Matrix Riccati Equation, lEEE Transactions on Automatic Control, AC-30 (10), pp. 962-970, (1985).

[25] Y. Li, S. Osher, *Coordinate Descent Optimization for l^1 Minimization with Application to Compressed Sensing; a Greedy Algorithm*, Inverse Probl. Imaging 3.3 (2009): 487-503.

[26] Z.-Q. Luo, P. Tseng. *On the convergence of the coordinate descent method for convex differentiable minimization.* Journal of Optimization Theory and Applications, 72(1), pp 7–35, (1992).

[27] I.M. Mitchell, A. M., Bayen, C. J. Tomlin, *A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. Automatic Control*, IEEE Trans. on Auto. Control, 50(7), pp. 947-957, (2005).

[28] I.M. Mitchell, C. J. Tomlin, *Overapproximating reachable sets by Hamilton-Jacobi projections*, J. Sci Comp. 19 (1-3), pp. 323-346, (2003).

[29] J.J. Moreau, *Proximité et dualité dans un espace hilbertien*, Bulletin Spc. Math. France 93, pp. 273-299, (1965).

[30] J.M. Ortega, W.C. Rheinboldt, Iterative solution of nonlinear equations in several variables. Academic Press, New York and London (1970).

[31] S. Osher, C.-W. Chu, *High Order Essentially Non-oscillatory Schemes for Hamilton-Jacobi Equations*, SIAM J. Num. Anal. 28 (4), pp. 907-922, (1991).

[32] S. Osher, J.A. Sethian, *Fronts Propagating with Curvature Dependent Speech: Algorithms Based on Hamilton-Jacobi Formulations*, J. Comput. Phys. 79, (1), pp. 12-49, (1988).

[33] S. Osher and B. Merriman, *The Wulff Shape as the Asymptotic Limit of a Growing Crystalline Interface*, Asian J. Math. 1 (3), pp. 560-571, (1997).

[34] R.T. Rockafellar, Convex Analysis. Princeton Landmarks in Mathematics, Princton University press, (1997).

[35] I.V. Rublev, *Generalized Hopf formulas for the nonautonomous Hamilton-Jacobi equation*, Computational Mathematics and Modeling 11.4,pp. 391-400 (2000).

[36] Y. Shen, Z. Wen, Y. Zhang, *Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization*, Optimization Methods Software 29(2), pp. 239-263, (2014).

[37] D.L. Sun, C. Fevotte, *Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence*, 2014 IEEE ICASSP, pp. 6201-6205, (2014).

[38] P. Tseng, P., *Convergence of a block coordinate descent method for nondifferentiable minimization*, Journal of Optimization Theory and Applications 109(3), pp.475-494 (2001).

[39] Y.H.R. Tsai, L.T. Cheng, S. Osher, H.K. Zhao, *Fast sweeping algorithms for a class of Hamilton–Jacobi equations*, SIAM J. Num. Anal 41 (2), pp. 673-694, (2003).

[40] J. N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Transactions on Automatic Control 40(9), pp. 1528-1538, (1995).

[41] J. Warga. *Minimizing certain convex functions.* Journal of the Society for Industrial & Applied Mathematics, 11(3), pp 588–593, (1963).

[42] Y. Wang, W. Yin, J. Zeng, *Global Convergence of ADMM in Nonconvex Nonsmooth Optimization*, preprint, UCLA CAM report 15-62, (2015).

[43] Z. Wen, C. Yang, X. Liu, S. Marchesini, *Alternating direction methods for classical and ptychographic phase retrieval*, Inv. Prob. 28(11), 115,010, (2012)

[44] Y. Xu, W. Yin. *A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion.* SIAM Journal on imaging sciences, 6(3), pp 1758–1789, (2013).

[45] Y. Xu, W. Yin. *A globally convergent algorithm for nonconvex optimization based on block coordinate update.* arXiv preprint arXiv:1410.1386, 2014.

[46] Y. Xu, W. Yin, Z. Wen, Y. Zhang, *An alternating direction algorithm for matrix completion with nonnegative factors* Frontiers of Mathematics in China 7(2), pp. 365-384, (2012).

[47] L. Yang, T.K. Pong, X. Chen, *Alternating direction method of multipliers for nonconvex background/foreground extraction*, arXiv:1506.07029, (2015).

[48] W. Yin, S. Osher, D. Goldfarb, J. Darbon, *Bregman Iterative Algorithms for l_1 Minimization with Applications to Compressed Sensing*, SIAM J. Imag. Sci. 1 (1), pp. 143-168, (2008).