UNIVERSITY OF CALIFORNIA
Santa Barbara

Efficient Guidance of Underpowered Vehicles in Time-Varying Flow Fields

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Mechanical Engineering

by

Blane Andrew Rhoads

Committee in Charge:

Professor Igor Mezić, Chair

Professor Frederic Gibou

Professor Jeffrey Moehlis

Professor Andrew Teel

December 2013

The Dissertation of
Blane Andrew Rhoads is approved:

Professor Frederic Gibou

Professor Jeffrey Moehlis

Professor Andrew Teel

Professor Igor Mezić, Committee Chairperson

November 2013

Efficient Guidance of Underpowered Vehicles

in Time-Varying Flow Fields

by

Blane Andrew Rhoads

To my family; past, present, and future

Acknowledgements

First and foremost, I'd like to thank my advisor, Professor Igor Mezić, who first introduced me to concept of an optimal trajectory of a nonlinear system, which continues to fascinate me, and who immediately provided me with the job of computing such trajectories, which continues to occupy me. Igor's continual support, confidence in my success, and flexibility in accommodating my career interests have been invaluable to me and have made my experience as a graduate student truly enjoyable.

Second, I'd like to thank my other committee members: Professor Frederic Gibou, for introducing me to Level Set Methods and adaptive grids, Professor Jeffrey Moehlis, for his interest in applying adaptive grids to the computation of isochrons for neuron models, and Professor Andrew Teel, for his excellent lectures on nonlinear systems, and for directly improving my analytical skills by so carefully grading my work.

Third, I'd like to thank my collaborators: Professor Andrew Poje from the Department of Mathematics at the City University of New York's College of Staten Island, and my labmate Dr. Alexandre Mauroy.

Fourth, I'd like to thank all the other wonderful people at UCSB who have enriched my experience here. In particular, Marko Budišić, who I consider a model labmate, my other labmates and colleagues over the years, the Mechanical Engineering Departmental staff, Professors Brad Paden, Bassam Bamieh, Gustavo Ponce (Math), Raya Feldman (Statistics), and countless others.

Fifth, I gratefully acknowledge funding from The Office of Naval Research through grants N00014-07-1-0587 and N00014-10-1-0611.

Last but not least, I'd like to thank my friends here and elsewhere, my mother Kathy, brother Brett, sister Stephanie, stepfather Tom, and my fiancée Charlotte, whose support and encouragement have been crucial to my success.

Curriculum Vitæ

Blane Andrew Rhoads

Education

2007	Bachelor of Science in Mechanical Engineering, University of Tulsa.
2007	Bachelor of Science in Applied Mathematics, University of Tulsa.

Experience

2008 – 2013	Graduate Student Researcher.
2007 – 2012	Teaching Assistant: "Statics", "Dynamics", "Vibrations", "Control Systems", "Robot Design".
Fall 2011	Research Assistant, *United Technologies Research Center, East Hartford, CT*; Multi-objective path planning for autonomous helicopters.
Summer 2009	Research Assistant, *Sandia National Lab, Combustion Research Facility, Livermore, CA*; Computational Singular Perturbation with Tabulation.
Summer 2007	Engineering Intern, *Intel Corporation, Chandler, AZ*; Avoiding First Wafer Delay in next-generation (450 mm wafer) process tools, Standard set of 2D Data Matrix ID laser marks for camera testing.

Honors

2012	Best Teaching Assistant for 2011-2012 academic year.
2007	Summa Cum Laude (4.00 GPA).
2007	Faculty Honors Medalist.
2007	Phi Beta Kappa.

Publications and Conference Participation

2013	Mauroy, A., Rhoads, B., Moehlis, J., Mezić, I., "Global isochrons and phase sensitivity of bursting neurons", *SIAM Journal of Applied Dynamical Systems*, accepted Oct 2013.

2013	Rhoads, B., Mezić, I., Poje. A., "Efficient Guidance of Underpowered Vehicles in Time-Varying Flows", *Proceedings of the 52nd IEEE Conference on Decision and Control (CDC)*, Florence, Italy.
2013	Rhoads, B., Mezić, I., Poje, A., "Minimum Time Heading Control of Underpowered Vehicles in Time-Varying Ocean Currents", *Ocean Engineering*, vol. 66, pp. 12-31, Elsevier.
2013	Presentation, "Navigating the Lagrangian Flow Map: Globally Optimal Feedback Control of Underpowered Vehicles in Time-Varying Flow Fields", *SIAM Conference on Applications of Dynamical Systems*, Snowbird, UT.
2012	Debusschere, B., Marzouk, Y., Najm, H., Rhoads, B., Goussis, D., Valorani, M., "Computational Singular Perturbation with Non-Parametric Tabulation of Slow Manifolds for Time Integration of Stiff Chemical Kinetics", *Combustion Theory and Modeling*, vol. 16, pp. 173-198, Taylor and Francis.
2011	Poster, "Globally Optimal Trajectories and Lagrangian Structures in a Time-Varying Flow Field"; *SIAM Conference on Applications of Dynamical Systems*, Snowbird, UT.
2010	Rhoads, B., Mezić, I., Poje. A., "Minimum time feedback control of autonomous underwater vehicles", *Proceedings of the 49th IEEE Conference on Decision and Control (CDC)*, pp. 5828-5834, Atlanta, GA.
2009	Presentation, "Coastal Ocean Flows and Nonlinear Optimal Control", *SIAM Conference on Applications of Dynamical Systems*, Snowbird, UT.
2009	Radloff, S., Abravanel, M., Rhoads, B., Steeg, D., van der Meulen, P., Petraitis, M., "First wafer delay and setup: How to measure, define, and improve first wafer delays and setup times in semiconductor fabs". *Advanced Semiconductor Manufacturing Conference (ASMC)*, pp. 86-90, IEEE/SEMI.

Abstract

Efficient Guidance of Underpowered Vehicles in Time-Varying Flow Fields

Blane Andrew Rhoads

In this dissertation we study high-level trajectory planning for underpowered vehicles in spatially complex, 2D, time-varying flow fields. In particular, we consider a minimum time problem and a minimum energy problem. These problems are difficult because locally optimal trajectories abound and currents stronger than the vehicle can push it far off course. Nevertheless, globally optimal trajectories can be obtained by numerical solution of a dynamic Hamilton Jacobi Bellman (HJB) partial differential equation (PDE) for the time-varying optimal cost-to-go function—the gradient of which yields an optimal feedback control law. Locally optimal trajectories are associated with shocks—discontinuities in the gradient of the cost-to-go. Strong currents are associated with discontinuities in the cost-to-go itself.

Existing work is primarily concerned with the proper capturing of shocks, and is mostly limited to weak, time-invariant flows. But strong, time-varying flows play a large role in the real-life problem. Thus the characterization of solutions for realistic flows has involved significant experimentation with novel solution approaches. A key theme has been the complementary nature of Eulerian or semi-Lagrangian finite difference methods and Lagrangian particle methods. The former methods

are associated with implicit front tracking methods such as Level Set Methods and Fast Marching, and rely on shock-capturing (e.g. Godunov) schemes. The latter methods are associated with explicit front tracking methods but compute particle trajectories in a higher-dimensional state-"costate" space using the well-known Euler Lagrange ordinary differential equations; these are variants of the so-called "extremal field" method. Other themes include the use of adaptive grids and the dichotomy between backward-in-time methods for closed-loop optimal trajectories and forward-in-time methods for open-loop optimal trajectories. In all cases, the resulting trajectories are globally optimal.

First, we present a forward-in-time Lagrangian method that exploits a special property of minimum time control to obtain open-loop optimal trajectories, without actually solving the dynamic HJB equation. The algorithm proved highly dependent on adaptive remeshing of the Lagrangian marker particles along the tracked front and a rather complicated trimming procedure for local optima. Examples include a numerically defined strong, time-varying flow field from a model of the Adriatic Sea. Second, we present a backward-in-time semi-Lagrangian method on an adaptive triangle grid for the fixed final time minimum energy problem. The algorithm effectively transforms the optimal control problem into a point wise-optimization problem, which we choose to solve exactly by keeping the discretizations first-order in space and time. The adaptive triangle grid proves very effective at capturing shocks, but the point-wise optimization is found to

be more computationally complex than hoped. Third, we present a coordinate transformation based on the Jacobian of the so-called flow map—the state at the fixed final time if one were to turn the control off. This suggests a local, greedy control scheme, which we compare to the minimum energy control in 1D and 2D. Finally, we conclude with a 1D proof of concept for a hybrid Lagrangian-Eulerian minimum energy algorithm that combines the best of the Lagrangian minimum time algorithm and the semi-Lagrangian minimum energy algorithms.

Contents

List of Figures

List of Tables

Chapter 1

Background and Overview

Fleets of robots known as AUVs (autonomous underwater vehicles) are now used around the world for tasks as diverse as commercial exploration, military surveillance, and scientific research. One class of AUV that has gained widespread popularity among oceanographers is that of *underwater gliders*, such as the Seaglider, the Spray, and the Slocum, described in [1], [2], and [3], respectively. A glider propels itself forward with fixed wings in a "saw tooth" shaped alternating sequence of long dives and ascents, interspersed by short actuations of a battery-powered piston to change its buoyancy. These vehicles are ideal for ocean sampling because of their long range and duration. However, this efficiency comes at the expense of controllability. Their speed relative to the water is often comparable to, if not smaller than, that of the flow fields they explore. This fact, along with increasing measurement and forecasting capabilities for ocean flow fields, suggest a new paradigm, in which the flow is viewed not as a disturbance, but as a time-varying nonlinear system to be incorporated into motion planning algorithms.

While much progress has been made developing efficient and robust model-based strategies for the control of coordinated groups of gliders—see for instance [4]—as well as for individual vehicles—see [5] and [6], the role of strong, spatially complex, time-varying environmental currents in path planning has received comparatively less attention. As stated in [4], "a methodology to exploit available estimates or predictions of the flow field is of significant interest".

As far as this dissertation is concerned, the primary problem is one of high-level trajectory planning, not low-level trajectory tracking or stabilization. At the spatiotemporal scales of interest (kilometers and hours), the dynamics in the horizontal plane play a much larger role than those in the vertical direction, and hydrodynamics and inertia may be ignored. The result is the 2D, time-varying, nonlinear, affine control system

$$\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}, t) + \mathbf{u}, \tag{1.1}$$

that is, the velocity of the vehicle is equal to the sum of its velocity $\mathbf{u}$ relative to the flow and the velocity $\mathbf{v}$ of the flow. The AUVs of interest typically operate at fixed speeds; this leads to a minimum time problem. Alternatively, we also consider a variable (but bounded) speed "minimum energy" problem, in which the cost of a trajectory is a combination of the integral of $\mathbf{u} \cdot \mathbf{u}$ and a fixed final time end cost.

The low dimensionality of (1.1) makes globally optimal control feasible. However, several challenges persist which are inherent in the real-life problem but often

avoided in the literature. First of all, $\mathbf{v}$ is spatially complex; generally, the number of locally optimal trajectories grows exponentially with respect to the spatial complexity and the length of the time interval of optimization. Second, $\mathbf{v}$ is time-varying; this essentially increases the dimensionality of the problem from two to three. Last but not least, the control input $\mathbf{u}$ is bounded and often smaller than $\mathbf{v}$; this makes "controllable sets" and "reachable sets" bounded and trajectory costs unbounded.

Globally optimal trajectories are typically obtained by numerical solution of a dynamic Hamilton Jacobi Bellman (HJB) partial differential equation (PDE) for the time-varying optimal cost-to-go function—the gradient of which yields an optimal feedback control law. The methods explored in this dissertation draw from two main types of methods: Eulerian or semi-Lagrangian finite difference methods, and Lagrangian methods that make use of the Euler Lagrange ordinary differential equations (ODEs)—the necessary conditions of optimal control. This chapter now goes on to present background on these methods and on the other topics that have proven relevant to the work at hand, in Sections 1.1-1.3, then a brief review of other more application-specific literature, in Section 1.5, and finally a detailed overview of the contributions of the dissertation, in Section 1.6.

1.1 Nonlinear Optimal Control

This section reviews the basic principles and equations of general nonlinear optimal control. There are two main approaches: the variational calculus approach, and the dynamic programming approach. The former is associated with the Pontryagin (minimum or maximum) Principle and the ODEs known as the Euler Lagrange equations, which serve as a necessary condition for optimal control. The latter approach is associated with Bellman's Principle of Optimality and the PDE known as the Hamilton Jacobi Bellman equation, which serves as a necessary and sufficient condition for optimal control. The relationship between the two approaches is important to the numerical methods that will be presented in this dissertation. All of these things will be described in the context of the following fairly general *optimal trajectory* problem.

Given an initial state $\mathbf{x}_0 \in \mathrm{R}^n$ and an initial time $t_0 \in [t_{\min}, t_{\max}]$, minimize the *cost functional*

$$J(\mathbf{u}, t_f; \mathbf{x}_0, t_0) := \int_{t_0}^{t_f} \mathcal{L}(\mathbf{x}(t), \mathbf{u}(t), t)dt + h(\mathbf{x}(t_f), t_f) \tag{1.2}$$

with respect to $\mathbf{u} : [t_0, t_f] \to \Omega$ and $t_f \in [t_0, t_{\max}]$, subject to some end condition(s)

$$\mathbf{m}(\mathbf{x}(t_f), t_f) = 0, \tag{1.3}$$

where the *state* trajectory $\mathbf{x} = \mathbf{x}(\cdot)$—implicitly dependent on the *control input* signal $\mathbf{u} = \mathbf{u}(\cdot)$—is the unique solution of the *state equation*

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{1.4}$$

with the initial conditions

$$\mathbf{x}(t_0) = \mathbf{x}_0. \tag{1.5}$$

From now on $\mathbf{x}$ may be used to mean either the state trajectory $\mathbf{x}(\cdot) : [t_0, t_f] \to \mathrm{R}^n$, as above, or an instantaneous state $\mathbf{x}(t) \in \mathrm{R}^n$, as implied by the context, and $\mathbf{u}$ may be used similarly. In Equation (1.2), the function $\mathcal{L}$ acts as a running cost function and is called the *Lagrangian*, and the function h is the *end cost* function. It will be assumed throughout that these and other functions such as the vector function $\mathbf{f}$ (including certain derivatives yet to be mentioned) are well-defined on the appropriate spaces.

The nature of this problem is largely dependent on the form of the vector function $\mathbf{m} : \mathrm{R}^n \times [t_{\min}, t_{\max}] \to \mathrm{R}^k$ $(0 \leq k \leq n)$ in Equation (1.3). The constraint $\mathbf{m} = 0$ may be thought of as a hypersurface where optimal trajectories must terminate in the $n + 1$ dimensional extended state space or "space-time". In general this terminal hypersurface spans a range of values of t_f, and thus t_f is free to vary along with $\mathbf{u}$, as the definition of J in (1.2) suggests. This is the case in "free final time" problems, including "minimum time" problems, which are the topic of Chapter 2. In Chapter 3 and beyond, we will also consider "fixed final time" problems, in which t_f is fixed, and removed as an argument of J in Equation

(1.2). In these problems, the hypersurface $\mathbf{m} = 0$ is contained within (or equal to) a planar time slice of the space-time domain. Note that we do not consider the more well-studied case of $t_f = \infty$ or "infinite horizon" optimal control because of the finite time nature of our motivating application.

In addition to the input constraint $\mathbf{u}(t) \in \Omega$, nonlinear optimal control theory also allows more general state/input constraints of the form $\psi(\mathbf{x}(t), \mathbf{u}(t), t) \leq 0$, or at least constraints of the state $\mathbf{x}$ to some subset of R^n; moreover, in our numerical methods we will explicitly consider state constraints due to coastlines. However, this introduction is sufficiently general without this complication. Note that in our application $\Omega \subset \mathrm{R}^n$ but that presently the dimensionality of Ω is intentionally unspecified.

1.1.1 Variational calculus and the Euler Lagrange equations

Far and away the most common approach to solving the above optimal trajectory problem, for individual trajectories, is to make use of the well-known necessary conditions of optimal control, derived from the calculus of variations. Namely, if $\mathbf{u}$ and t_f are optimal, then, in addition to the conditions (1.3), (1.4), and (1.5), there exists a *costate* trajectory $\mathbf{p} : [t_0, t_f] \to \mathrm{R}^n$ satisfying the *costate equation*

$$\dot{\mathbf{p}} = -\mathcal{H}_{\mathbf{x}}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t), \tag{1.6}$$

the *Pontryagin principle*

$$\mathcal{H}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t) = \min_{\mathbf{u}' \in \Omega} \mathcal{H}(\mathbf{x}, \mathbf{u}', \mathbf{p}, t), \tag{1.7}$$

the end condition

$$h_{\mathbf{x}}(\mathbf{x}(t_f), t_f) - \mathbf{p}(t_f) = \mathbf{d} \cdot \mathbf{m}_{\mathbf{x}}(\mathbf{x}(t_f), t_f), \tag{1.8}$$

for some vector $\mathbf{d} \in \mathrm{R}^k$, and, if and only if t_f is indeed free to vary (i.e. if $\mathbf{m}(\mathbf{x}, t) = 0$ does not imply $t = t_f$), the additional "free final time" end condition

$$\mathcal{H}(\mathbf{x}(t_f), \mathbf{u}(t_f), \mathbf{p}(t_f), t_f) + h_t(\mathbf{x}(t_f), t_f) = \mathbf{d} \cdot \mathbf{m}_t(\mathbf{x}(t_f), t_f), \tag{1.9}$$

where $\mathcal{H}$ is called the *Hamiltonian* and is defined by

$$\mathcal{H}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t) := \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \cdot \mathbf{p} + \mathcal{L}(\mathbf{x}, \mathbf{u}, t). \tag{1.10}$$

Throughout this dissertation subscript notations like the those above will be used to indicate partial derivatives or Jacobian matrices. For example, $\mathbf{f}_{\mathbf{x}}$ is the Jacobian matrix of the original control system $\mathbf{f}$. Note that differentiating (1.10) yields $\mathcal{H}_{\mathbf{x}} = \mathbf{f}_{\mathbf{x}}\mathbf{p} + \mathcal{L}_{\mathbf{x}}$.

Expressing the state equation (1.4) in terms of $\mathcal{H}$ and coupling it with the costate equation (1.6) yields the Hamiltonian system of ODEs

$$\dot{\mathbf{x}} = \mathcal{H}_{\mathbf{p}}; \quad \dot{\mathbf{p}} = -\mathcal{H}_{\mathbf{x}}. \tag{1.11}$$

These are known as the *Euler Lagrange equations*. The initial condition (1.5) specifies the initial state but not the initial costate. Similarly, the end conditions

(1.8) and (1.9) yield only partial information about the final state $\mathbf{x}(t_f)$, the final costate $\mathbf{p}(t_f)$, and the final time t_f. Thus the original optimal trajectory problem is transformed into a so-called *two-point boundary value problem* (TPBVP), consisting of these boundary conditions and the Euler Lagrange equations (1.11), in which it is assumed $\mathbf{u}$ can be determined point-wise as the argument of the minimum (1.7).

The typical approach is to solve this TPBVP using an iterative method. There are many well-known problems with this approach. Most importantly, a solution of the TPBVP is merely a candidate optimal trajectory, sometimes called an *extremal trajectory* or an *extremal*; it may or may not be a local optimum, and if it is a local optimum, it may or may not be the desired global optimum. Secondly, a given iterative method may or may not even converge to an extremal for a given problem.

One such method is the *shooting method*. The (forward-in-time) shooting method searches iteratively for solutions of the TPBVP by numerically solving the initial value problem (IVP) defined by (1.11), (1.5), and $\mathbf{p}(t_0) = \mathbf{p}_0$ for various initial costates $\mathbf{p}_0 \in \mathrm{R}^n$. (Alternatively, one can also work backward in time, in which case the "initial" values of the IVP are $\mathbf{x}(t_f) = \mathbf{x}_f$ and $\mathbf{p}(t_f) = \mathbf{p}_f$ satisfying the end conditions instead of $\mathbf{x}(t_0) = \mathbf{x}_0$ and $\mathbf{p}(t_0) = \mathbf{p}_0$.) In addition to the cost J of each of these IVP solutions, one can presumably define a secondary cost K based on how "far" one is from satisfying the end conditions. Thus the

infinite dimensional optimization of the signal $\mathbf{u}(\cdot)$ is transformed into the nD optimization of $\mathbf{p}_0$. If n is large or K is too complex, global optimization is indeed hopeless. However, the shooting method highlights an alternative possibility: where n is small and the solution is not overly complex, and global optimization is feasible. This leads to the idea of the *extremal field* approach—a missing link of sorts between the typical TPBVP approach and the dynamic programming approach.

1.1.2 The extremal field approach

Though apparently not as popular in the literature as its counterparts, the extremal field approach is at least briefly described in the popular textbook [7]. Here and throughout, *extremal trajectory* or *extremal* means a solution of one of the family of Euler Lagrange IVPs involved in the shooting methods described above, whether forward-in-time or backward-in-time, and whether or not it also solves the TPBVP. The idea of the approach is simple: one *can* declare a solution of the TPBVP to be globally optimal if one has explored the entire "field of extremals" sufficiently. For example, if one is working forward-in-time, and $n = 2$, one just needs a sufficiently resolved (and ideally adaptive) grid of points $\mathbf{p}_0 \in \mathrm{R}^2$. Although the forward-in-time approach turns out to have a distinct advantage over the backward-in-time approach for the special case of minimum time control, as will be shown in Chapter 2, the backward approach is generally

better, in that it allows the definition of an optimal feedback control law. Whereas forward extremals are parameterized by initial costates $\mathbf{p}_0$, backward extremals are parameterized by the one or more final costates $\mathbf{p}_f$ associated with various final state-times $(\mathbf{x}_f, t_f)$ on the target surface $\mathbf{m}(\mathbf{x}, t) = 0$. The effect is to "flood" space-time with extremals as densely as is reasonable, such that initial state-time $(\mathbf{x}_0, t_0)$ is on, or at least very close to, one or more of the extremals; then by some combination of direct minimization and interpolation one can evaluate the optimal feedback control law locally and/or extract the entire optimal trajectory. Note that for the "time-invariant problem", where $\mathcal{L}$, h, $\mathbf{m}$, and $\mathbf{f}$ are time-invariant and t_f is free, the space to be filled by extremals is simply the nD state space instead of the $(n+1)$D extended state-time space.

One of the few good examples of this method found in the literature is that of [8]. The application is actually for the infinite time horizon optimal control problem of (fully nonlinear, but time-invariant) quadratic regulation of a pendulum on a cart about its unstable equilibrium. By linear approximation of the solution in a small region containing the target equilibrium, the infinite horizon problem is transformed to a time-invariant, free final time problem, in which the field of extremals is parameterized by final states $\mathbf{x}_f$ on the 1D surface $\mathbf{m}(\mathbf{x}, t) = \mathbf{m}(\mathbf{x}) = 0$ bounding the small region of linearization (where each $\mathbf{x}_f$ readily yields one final costate $\mathbf{p}_f$). The resulting (backward-in-time) extremals are computed one-at-a-time, adaptively, using a simple bisection algorithm. We'll refer to methods

like this that compute extremals independently as "pure extremal field" methods. We'll use "extremal field" or "field of extremals" much more loosely, to refer to any method which computes some continuum of extremal trajectories in n-D space by embedding them in the $2n$-dimensional state-costate space.

One very effective such method is the continuation method of [9], which treats the same cart-pendulum problem as [8]. Like [8], [9] identifies the desired continuum of trajectories as a stable invariant manifold of the Hamiltonian Euler Lagrange system about the fixed point. But whereas [8] "explores" this manifold (which we will later call the *extremal surface*) one trajectory at a time, specifying the time interval a priori, [9] computes this manifold one "level set" at a time. Specifically, the manifold is computed using level sets of a geodesic distance defined in $\mathrm{R}^{2n} = \mathrm{R}^4$. The more typical alternative is to use level sets of time or of the cumulative cost J; this will be the case in what we call "front" based methods or front-tracking methods, e.g. in Section 1.2.

Pure extremal field methods like that of [8] and other extremal field methods like that of [9] are like depth-first and breadth-first methods, respectively, e.g. for a path in a discrete graph, with the latter being farther removed from TPBVP shooting methods and closer to dynamic programming. Pure extremal field methods have the advantages of algorithmic simplicity and parallelizability. On the other hand, level set based or front based extremal field methods are slightly better equipped to deal with inherent sensitivity of extremal trajectories with respect

to the given initial (or terminal) conditions. The sensitivity is partly due to the Hamiltonian nature of the Euler Lagrange equations. The length of the level set and thus the "number" of extremal trajectories grow exponentially with respect to the length of the time interval of computation and to the spatial complexity of the underlying system $\mathbf{f}$.

This exponential growth is reflected in the fact that projecting the extremal surface from R^{2n} into R^{n} (or, in the time-varying case, from R^{2n+1} to R^{n+1}), yields a mess of intersecting locally optimal extremal trajectories and self-intersecting level sets. While it is interesting to observe the variety of local optima, the real interest is in the global optima. These can be obtained, as mentioned, by direct minimization of the multi-valued graph of J vs. $\mathbf{x}$ along the computed surface. This minimum of J is the so called value function—the solution of the HJB equation. Hence the extremal field method is really a method of characteristics for the HJB equation. The challenge is to make the extremal field approach efficient by not computing the suboptimal part of the extremal surface. One of the goals of this dissertation is to propose algorithms for doing exactly this, and point out their inherent advantages and disadvantages. Also of interest is to compute the value function—to solve the HJB equation—directly, on a structured, Eulerian grid in the (extended) state space, rather than indirectly on the unstructured, Lagrangian grid yielded by the projection of the extremal surface.

1.1.3 Dynamic programming and the Hamilton Jacobi Bellman equation

In the most general sense, the term *dynamic programming*, first used by Richard Bellman in the 1940s, refers to the strategy of breaking down a complex, sequential decision-making problem into simpler, nested or recursive subproblems. More specifically, it refers to the global solution of optimal trajectory problems like ours in terms of the so-called *value function*. In the present, continuous time, time-varying case the *value function* V is defined (in terms of the trajectory cost (1.2)) by

$$V(\mathbf{x}_0, t_0) := \min_{\mathbf{u}, t_f} J(\mathbf{u}, t_f; \mathbf{x}_0, t_0). \tag{1.12}$$

Also called the (optimal) *cost-to-go*, this function is really only meaningful if the problem satisfies Bellman's *Principle of Optimality* [10], namely, if

$$J(\mathbf{u}, t_f; \mathbf{x}_0, t_0) = V(\mathbf{x}_0, t_0) \implies J(\mathbf{u}, t_f; \mathbf{x}(t), t) = V(\mathbf{x}(t), t), \quad \forall t \in [t_0, t_f],$$

that is, if $(\mathbf{x}, \mathbf{u})$ being optimal implies that the restriction of $(\mathbf{x}, \mathbf{u})$ to any subinterval $[t, t_f]$ is also optimal. If an optimal trajectory problem satisfies the Principle of Optimality, a necessary and sufficient condition for the existence of the value function (1.12) is that it is the solution of the time-varying *(dynamic) Hamilton Jacobi Bellman (HJB) equation*

$$V_t(\mathbf{x}, t) = -\min_{\mathbf{u} \in \Omega} \mathcal{H}(\mathbf{x}, \mathbf{u}, V_{\mathbf{x}}(\mathbf{x}, t), t), \tag{1.13}$$

with the boundary condition

$$V(\mathbf{x}, t) = h(\mathbf{x}, t)$$

along the terminal hypersurface $\mathbf{m}(\mathbf{x}, t) = 0$. Moreover, the point-wise argument $\mathbf{u}$ of the minimum in (1.12) defines an optimal feedback control law, whose simulation directly yields solutions of the optimal trajectory problem for as many initial state-times $(\mathbf{x}_0, t_0)$ as desired.

The problem of obtaining V and $\mathbf{u}$ on (a subset of) $\mathrm{R}^n \times [t_{\min}, t_{\max}]$ will be referred to as the *optimal feedback control* problem. Note that for HJB equations as with more general Hamilton Jacobi (HJ) equations, "solution" technically means *viscosity solution* [11, 12]. Roughly speaking, ∇V and V itself are generally discontinuous, and not well-defined along their discontinuities. Discontinuities in ∇V or "cusps" in V include both *shocks* and *rarefactions*, terms borrowed from the study of conservations laws. For our purposes, shocks are where equally optimal trajectories collide and terminate in backward time—including but not limited to locally maximal ridges in V—and rarefactions are where optimal trajectories merge in forward time—including but not limited to locally minimal valleys in V. The proper capturing of rarefactions is a requirement for all of the prior and present methods described in this dissertation, but is not emphasized, since rarefactions don't persist in time like shocks and since our numerical examples do not include non-smooth h. More importantly, discontinuities in V itself (e.g due to lack of local controllability as in the problem at hand) are not considered in

the more popular numerical methods for the HJB equation, which focus on fast methods for the time-invariant case, as will be explained in in Sections 1.2.2 and 1.2.3.

The most relevant prior methods for the time-varying HJB equation are the backward-in-time semi-Lagrangian methods of [13] and [14]. These methods are quite relevant to Chapter 3. Roughly speaking, they transform the optimal control problem into a point-wise optimization problem, which they solve iteratively, and they capture shocks properly but not very precisely since they don't use adaptive grids. In [13] the basic semi-Lagrangian method is shown to be equivalent, at least in 1D, to the Eulerian Godunov unwinding schemes often used in the Level Set Methods to be described in Section 1.2.1.

Finally, the most recent and perhaps the most promising method for the time-varying HJB equation is the implicit Eulerian method of [15]. This method applies the Fast Marching method to be described in Section 1.2.2 at each time slice, and is shown to be superior to explicit Eulerian methods for certain cases. However, as of now the results are limited to cases where the flow is zero and the vehicle speed is spatially dependent.

For time-invariant problems, $V_t = 0$, $\mathcal{H} = \mathcal{H}(\mathbf{x}, \mathbf{u}, V_x(\mathbf{x}))$, and (1.13) reduces to a so-called *static HJB equation*, which is analogous to the discrete time *Bellman equation*. Though the present setting is more rare, and dynamic programming is more often associated with discrete time and discrete space settings, the approach

is the same, and it is easy enough to see that the Principle of Optimality holds. The Principle of Optimality does not typically hold, for instance, in the case of receding horizon control. For proofs of the Principle of Optimality, the derivation of the HJB equation, and the derivation of the Euler Lagrange equations from the HJB equation, we recommend [7]. The approximation of a (time-invariant) continuous space problem by a discrete space problem will be discussed in Section 1.2.2 in the context of the Fast Marching method.

Having completed the introduction to general nonlinear optimal control theory, from now on we will restrict to affine systems (1.1), i.e. assume $\mathbf{f}(\mathbf{x}, \mathbf{u}, t) = \mathbf{v}(\mathbf{x}, t) + \mathbf{u}$ in (1.4).

1.2 Minimum Time Control and Front Tracking

A *minimum time control* problem is one where t_f is left free to vary, $\mathcal{L} = 1$, and $h = 0$ in (1.2), and where $\mathbf{u}$ is somehow bounded. For our purposes, $\Omega = \{\mathbf{u} \in \mathrm{R}^n, |\mathbf{u}| \leq s\}$, and $s > 0$ is the (maximum) "speed". In turn $\mathcal{H} = (\mathbf{v} + \mathbf{u}) \cdot \mathbf{p} + 1$ and is minimized by $\mathbf{u} = -s\mathbf{p}/|\mathbf{p}|$, which yields $\mathcal{H} = \mathbf{v} \cdot \mathbf{p} - s|\mathbf{p}| + 1$ and thus the *dynamic minimum time HJB equation*

$$V_t(\mathbf{x}, t) = -\mathbf{v}(\mathbf{x}, t) \cdot V_x(\mathbf{x}, t) + s|V_x(\mathbf{x}, t)| - 1. \tag{1.14}$$

If $\mathbf{m}$ and $\mathbf{v}$ are time-invariant, then so is V, i.e. $V_t = 0$, and (1.14) reduces to the *static minimum time HJB equation*

$$s|V_x(\mathbf{x})| - \mathbf{v}(\mathbf{x}) \cdot V_x(\mathbf{x}) = 1. \tag{1.15}$$

Finally, if $\mathbf{v} = 0$, then (1.15) reduces to the *Eikonal equation*

$$s|V_x(\mathbf{x})| = 1. \tag{1.16}$$

Now the cost-to-go V is simply the "time-to-go", and the boundary condition is that $V = 0$ along the terminal surface $\mathbf{m} = 0$. A variety of prior methods exist for numerically solving variants of these three equations. The better known methods are Level Set Methods, Ordered Upwind Methods, and Fast Marching Methods, respectively. These are all Eulerian or semi-Lagrangian finite difference methods for implicit front tracking, as will be explained in the remainder of this section.

To our knowledge prior Lagrangian methods for minimum time control are limited to the static case, i.e. (1.15) and (1.16). The best example is that of [16], which we classify as an extremal field method in that it uses the Euler Lagrange equations. However, we also classify this method as a front tracking method because it explores the extremal surface more efficiently than [8] or [9], by constantly trimming the level sets of J, projected from R^4 into R^2, to directly yield the level sets of V. This type of "trimming" is numerically challenging, and is one of the key components of the method we present in Chapter 2, for the time-varying case.

1.2.1 Level Set Methods

Inspired by numerical methods for conservation laws, *Level Set Methods* (LSMs) were introduced by Osher and Sethian in 1988 for the application of curvature-dependent propagation of flame fronts [17], and have since been extended to a variety of applications. The general problem is to simulate the time evolution of some $(n-1)$D surface or front in nD space. The basic strategy is to define the front *implicitly* as the zero level set of a scalar *level set function* $\phi(\mathbf{x}, t)$ on an grid in nD space, instead of *explicitly* with a grid fitted to the surface. Often the term *front capturing* is used to distinguish implicit methods like LSMs from explicit front tracking methods. Additionally, we tend to refer to implicit methods as "Eulerian" and explicit methods as "Lagrangian". Ideally ϕ is the *signed distance function*—the distance to the front times +1 if "outside" front or times -1 if "inside" the front—and much if not most of the computational effort goes toward *reinitialization*—pausing every few time steps to numerically correct Φ for the natural deformation that generally accumulates. The front is evolved by evolving ϕ according to a *level set equation* such as

$$\phi_t + \mathbf{V}(\mathbf{x}, t) \cdot \phi_x = 0, \tag{1.17}$$

where $\mathbf{V}$ (not to be confused with V or $\mathbf{v}$) is the velocity of a particle on the front (or in our case a controlled vehicle) LSMs rely on non-standard spatial discretization methods such as Godunov unwinding schemes and essentially non-oscillatory (ENO) schemes. They offer the advantage of properly capturing changes in topol-

ogy (e.g. bubbles merging) as well as singularities such as *shocks* and *rarefactions* more naturally than explicit methods. Basic LSMs are algorithmically simpler but computationally (and memory-wise) more complex than explicit methods, since they work in nD instead of $(n-1)$D space. However, a variety of more efficient versions have been developed, including quadtree and octree adaptive grid based LSMs, introduced in [18] and [19], respectively, and discussed further in Section 1.4, as well as hybrid particle LSMs [20]. Two popular texts on Level Set Methods are [21] and [22].

Level Set Methods (or perhaps more aptly the related Ordered Upwind Methods to be described in Section 1.2.3) could conceivably be used to solve the dynamic minimum time HJB equation (1.14) if not more general dynamic HJB equations. The tracked front would be defined by increasing level sets of $V(\mathbf{x}, t)$ and initialized at the terminal hypersurface. In [23], to be summarized in Chapter 2, we presented a backward-in-time extremal field method that essentially tracks this same front using Lagrangian marker particles, for the case of minimum time.

But one can obtain minimum time trajectories more efficiently without solving the HJB equation. In this case the front of interest is what we will call the "reachability front"—the boundary of the subset of R^n that is reachable at time $t \geq t_0$ from a particular point $(\mathbf{x}_0, t_0)$, and $\mathbf{V}(\mathbf{x}, t) = \mathbf{v}(\mathbf{x}, t) - s\phi_{\mathbf{x}}(\mathbf{x}, t)/|\phi_{\mathbf{x}}(\mathbf{x}, t)|$ in (1.17), i.e. $\phi_t = -\mathbf{v} \cdot \phi_x + s|\phi_x|$. Note that the role of ϕ here is much like that of V in (1.14), but that the -1 does not appear. In addition, the reachability front

evolves forward in time from $\mathbf{x}_0$, and yields an open-loop trajectory, whereas the solution of the HJB equation yields closed-loop trajectories; this contrast is one of the topics of Chapter 2, the focus of which is a forward-in-time method published in [24]. Recently, the exact same problem was solved via an LSM in [25]. The Eulerian LSM approach is as useful for the reachability front as for other more general fronts and in fact is closely related to the semi-Lagrangian HJB method of Chapter 3. Moreover, the method of [25] is not very sophisticated. For example, it makes no mention of a reinitialization procedure or the lack thereof. However, LSMs for the reachability front are not pursued in this dissertation.

Another quite relevant application of standard Level Set Methods to control is in the solution of a Hamilton Jacobi Isaacs equation for backward reachable sets in [26]. The same author provides a popular Matlab toolbox for the associated Eulerian schemes—including total variation diminishing (TVD) Runge-Kutta (RK) explicit time-stepping schemes, and essentially non-oscillatory (ENO) and Godunov unwinding spatial discretization schemes.

The larger impact of LSMs on optimal control has been through the offshoot of Fast Marching and Ordered Upwind Methods for the solution of static HJB equations, e.g. for the case of time-invariant flows $\mathbf{v}(\mathbf{x}, t) = \mathbf{v}(\mathbf{x})$.

1.2.2 Fast Marching

Fast Marching is a finite difference method for approximating the solution of the Eikonal equation (1.16), with the added generality of a spatially-dependent front propagation speed $s = s(\mathbf{x})$ (assuming $s(\mathbf{x})$ is positive and bounded away from 0). It was presented by Sethian with the name "fast marching" in 1996, as a simplified level set method for the special case of monotonically advancing fronts, using an Eulerian discretization [27]. But the same basic algorithm was also presented by Tsitsiklis in 1995, from a control theoretic perspective, using a semi-Lagrangian discretization [28]. The front of interest propagates backward in time from the target set (a discrete version of the terminal surface $\mathbf{m} = 0$ of (1.3)), and its snapshots give the level sets of the time-to-go V. Alternatively, the front propagates forward in time from the initial point $\mathbf{x}_0$, and V is the "time-to-arrive"; in fact this forward approach is more common in the literature, but offers no practical advantage over the backward approach, which offers the advantage of feedback control. An all-purpose level set method would need to compute each snapshot on a separate nD grid in terms of the level set function ϕ; the fast marching method simply computes all snapshots on a single nD grid, in terms of V itself.

The Fast Marching Method is closely related to Dijkstra's method—a well-known dynamic programming algorithm for the shortest path on a discrete graph [29]. Both algorithms are referred to as a *single pass* algorithms, to distinguish

them from iterative algorithms where V may be updated many times for each node or grid point. Roughly speaking these single pass algorithms work by maintaining estimates of V at "neighbors" of "accepted" points and "accepting" the "neighbors" in order of increasing V. In this sense they take advantage of the inherent *causality* of the problem: V must be decreasing along optimal paths. Assuming the estimates of V are kept sorted in an efficient data structure such as a *heap*, both Fast Marching and Dijkstra's algorithm are $O(N \log N)$ in the number of nodes or grid points N (versus $O(N^2)$ if the sorting is done using a simple array).

It's not unreasonable to attempt to approximate solutions of the continuous problem with solutions of the discrete problem. Whereas the HJB equation captures the continuous problem exactly and its solution is inherently approximate, the discrete Bellman equation is an approximation of the continuous problem and its solution is exact. The problem is that the discrete problem is a very bad approximation of the continuous problem. In particular, the number of possible directions of motion is limited by the connectivity of the discrete graph. This principle is well-established, and explained very clearly for example in [30]. Fast Marching does not constrain the direction of motion and thus generally yields smooth trajectories, while still maintaining the efficiency of Dijkstra's method. The difference is in the estimation of V at a particular point from the values at neighboring points i—in both cases a minimum of the form $V_i + \Delta V_i$. Dijkstra's

algorithm uses the discrete minimum from among all the neighbors. Fast Marching simply considers the continuum of values obtained by linear interpolation between neighboring pairs of neighbors. Fast Marching has also been extended to higher-order Eulerian schemes [31] and triangular grids [32].

Although not reflected in (1.16), fast marching works for spatially dependent speeds of the form $s = s(\mathbf{x})$. Problems like these, e.g. minimum time control of a vehicle in a zero flow field, are called *isotropic*. Ordered Upwind Methods generalize Fast Marching to the case of *anisotropic* problems—where the speed of propagation also depends on the front *normal*: the vector $\frac{V_x}{|V_x|}$.

1.2.3 Ordered Upwind Methods

Ordered Upwind Methods (OUMs) are a class of finite difference methods for Hamilton Jacobi equations of the form

$$S\left(\mathbf{x}, \frac{V_x}{|V_x|}\right)|V_x| = 1, \tag{1.18}$$

with the speed of propagation S positive and bounded away from zero. OUMs were introduced in 2001 [33] and 2004 [30] by Sethian and Vladimirksy. They include both semi-Lagrangian and Eulerian versions, thus generalizing the methods of [28] and [27], respectively. The static minimum time HJB equation (1.15) corresponds to the special case of (1.18) for which $S = s - \mathbf{v} \cdot \frac{V_x}{|V_x|}$. And the Eikonal equation (1.16) (solvable by Fast Marching) corresponds to the special case of (1.15) for which the flow $\mathbf{v}$ is zero and thus $S = s$.

One of the main cases of interest in the application at hand is the case of (1.15) where the flow is "strong" (i.e. $\exists \mathbf{x}$ s.t. $|\mathbf{v}(\mathbf{x})| >= s$). In this case, OUMs suffer from the limitation that S is not guaranteed to be bounded away from zero. Furthermore, the closer S is to zero, the worse the anisotropy of the problem, and the worse the performance of the algorithm. In particular, if $0 < S_{\min} \leq S(\mathbf{x}, \mathbf{p}) \leq S_{\max} < \infty$ for all states $\mathbf{x}$ and unit vectors $\mathbf{p}$, the *anisotropy coefficient* is defined as $\frac{S_{\max}}{S_{\min}}$, and the computational complexity is $O\left(\frac{S_{\max}}{S_{\min}} N \log N\right)$ in the number of grid points N (whereas Fast Marching is $O(N \log N)$).

Finally, semi-Lagrangian OUMs have been extended to the case of time-varying minimum time problems in [34]. The added efficiency of this method over time-marching methods for solution of the time-varying HJB equation is thought to be comparable to that of the minimum time algorithm of 2, but the method requires (small-time) local controllability.

1.3 The Flow Map

The *flow map* $\mathbf{X}$ maps states $\mathbf{x}$ along the passive particle trajectories of the flow field $\mathbf{v}$ forward or backward from one time t to another time T. In other words, the flow map is defined by solutions of

$$\frac{d}{dT}\mathbf{X}(\mathbf{x}, t, T) = \mathbf{v}\left(\mathbf{X}(\mathbf{x}, t, T), T\right). \tag{1.19}$$

for which $\mathbf{X}(\mathbf{x}, t, t) = \mathbf{x}$.

The flow map and its Jacobian are now widely used in the visualization of time-varying nonlinear dynamical systems, e.g. for the study of transport in geophysical flows. Most often, the Jacobian is visualized in terms of Finite Time Lyapunov Exponent (FTLE)—a measure of the average hyperbolicity or stretching along a given particle trajectory; "ridges" in the FTLE field are often used to approximate transport barriers known as Lagrangian Coherent Structures (LCS). The term "flow map" was popularized in [35], in the context of LCS and FTLEs. More recently, the flow map Jacobian has been visualized in terms of a quantity known as Mesohyperbolicity [36], which captures hyperbolicity but also rotation.

The flow map and its Jacobian are relevant to control, especially when the control is bounded and occurs over short time intervals, and thus the ability to change one's fate is limited. Attempts to exploit this connection seem to be limited to the context of LCS and FTLEs, as in [37] and [38]. This is understandable, since control objectives for small vehicles such as gliders often involve exploring the structure of the flow itself and/or transitioning between the so-called almost invariant sets often delineated by LCS. But it's a bit disappointing, since the flow map and its Jacobian can also be used more directly to estimate trajectory costs and their sensitivity to instantaneous control inputs, regardless of where a target state lies relative to the LCS and invariant sets. This is the topic of Chapter 4.

A variety of numerical methods have been developed for the efficient computation of the flow map $\mathbf{X}$ and its Jacobian $\mathbf{X_x}$ from a given flow field $\mathbf{v}$, especially

for the purpose of visualizing LCS via FTLE fields. Often a 2D time-varying flow $\mathbf{v}$ is defined numerically on a 3D grid, for instance from the numerical solution of the Navier Stokes equations, and thus must be interpolated; a popular tricubic interpolation scheme was presented in [39]. Typically $T - t$ is fixed (rather than T) and the Jacobian $\mathbf{X}_\mathbf{x}$ is computed via finite differences. In the simplest case, $\mathbf{X}$ is computed for a uniform 3D grid of points $(\mathbf{x}, t)$, and there is significant overlap in the time intervals of trajectory integration. Besides higher-order ODE solvers, there are two main ways of making this more efficient: adaptive 2D grids, to be discussed in Section 1.4, and flow map composition methods like those of [40], which uses spatial interpolation to avoid redundant time integration. Another strategy which tends to go hand-in-hand with adaptive grids is to compute the Jacobian $\mathbf{X}$ by time integration of n^2 additional ODEs, in addition to the n ODEs (1.19), instead of by finite differencing, as in [41]. This strategy has obvious advantages even for uniform grids but seems to be underutilized, apparently because the numerical diffusion introduced by finite differencing is considered desirable when attempting to resolve smooth but very narrow ridges in FTLE fields.

1.4 Adaptive Grids

Adaptive grids of various kinds play a large role in many of the numerical methods described in this Chapter and in the remainder of this dissertation. The general idea is to focus the computation of some scalar field in the area of greatest

interest. For general global optimization, this would presumably be in the vicinity of the candidate optima, and/or where large variations are observed. Of particular interest is the ability to adaptively capture cusps in a given surface (discontinuities in the gradient, not necessarily extremal), as well as large gradients or even discontinuities.

For example, the bisection algorithm of [8] is a 1D adaptive grid algorithm for an extremal field. The hypothetical extremal field discussed in Section 1.1.2, parameterized by costates $\mathbf{p}_0 \in \mathrm{D} \subset \mathrm{R}^2$, would ideally use some kind of 2D adaptive grid.

As mentioned, the adaptive grids of choice for LSMs are quadtree [18] and octree [19] grids. A quadtree (octree) grid begins as a uniform grid and is refined where needed by dividing a grid cell into four (eight) quadrants. Cell objects can then be stored in tree data structure where each node of the tree points to either four (eight) "child" nodes or none; hence the name quadtree (octree). Typically the grid is refined to some maximum level of resolution uniformly along the tracked interface. In addition, a grid may be *graded*, that is, one may enforce the requirement that neighboring grid cells differ in size by no more than a factor of two; this is not the case in [19]. Adaptive grids significantly complicate the discretizations and, notably, makes them less parallelizable, but effectively reduce the dimensionality of the problem by approximately one.

The primary goal of Ordered Upwind Methods for static HJB equations (including Fast Marching) is to properly capture shocks—cusps in the cost-to-go V. Unfortunately, the adaptive placement of grid points based directly on V here is not straightforward, due to the single-pass nature of the algorithms. One option that has been explored, e.g. in [42] and [43], is to focus on large gradients of V by placing more grid points where the speed or propagation S is small. Both of these use adaptive triangular grids, applying the Fast Marching method of [44] for unstructured triangular grids.

Adaptive grid methods for computing the smooth but highly sensitive flow map (and FTLE field) such as those of [45], [46], and [41] are relevant for two reasons. First, they may be directly useful for computing the less sensitive but non-smooth cost-to-go function. Second, the flow map itself is useful for understanding optimal trajectories and for generating good suboptimal trajectories. To summarize, [45] partially inspired the adaptive grid refinement criterion to be used in Chapters 3 and 4, and indeed avoids unnecessary trajectory computations, but does nothing to alleviate the memory costs; it merely subsamples the adaptively computed flow map onto a very large uniform rectangular grid. [46] and [41] are of less interest, since they require more complicated unstructured triangle grids. However, the latter method is impressive in its own right, and demonstrates the suitability of anisotropic adaptive grids (long skinny triangles) for FTLEs.

In general structured triangle and higher dimensional tetrahedral grids offer an attractive alternative to quadtree and octree grids. In particular, [47] presents a general class of triangle and tetrahedral grids that extend to any dimension. According to [47], one of the advantages of tetrahedral grids is that they contain fewer vertices per volumetric grid element.

1.5 Other Relevant Literature

Work in the area of path planning specifically for AUVs and UAVs is somewhat limited, even for 2D, time-invariant flow fields, but includes some interesting results that provide insight into more general cases. For instance, [48] uses geometric arguments to extend Dubins's results for minimum time paths of a car with a bounded turning radius to the case of a UAV in time-invariant, spatially-uniform wind. [49] characterizes the various locally optimal minimum time trajectories of gliders—for time-invariant flow fields that model the nonlinear eddies and shear typical in the ocean—obtained by integrating "ray equation[s]" related to nonlinear light and sound propagation. These ray equations are essentially the same as the Euler Lagrange equations.

Work in the area of path planning for AUVs in realistically complex flow fields is mostly limited to more heuristic methods. For example, [50] and [51] optimize trajectories in model output for a 2D, time-varying, coastal flow and in an analytic but 3D and time-varying estuarine flow, respectively. One class of

iterative sampling-based methods that somewhat mirrors the idea of our forward-in-time approach is that of RRTs (rapidly exploring random trees), which are used in [52] to compute energy efficient paths in models of (2D, time-invariant) ocean flow fields.

A couple of exceptions are [53], which considers spatially complex, time-invariant flow fields, and [54], which considers spatially-complex, time-varying flow fields, as well as variable departure times. These use dynamic programming algorithms—the A* algorithm [55], and a generalization of Dijkstra's method called symbolic wavefront expansion, respectively—to obtain globally optimal paths on discrete graphs. In both of these the connectivity of the (interior) graph nodes is eight. While the problems are of interest, the constraint of the vehicle to discrete graphs in such complex flows is a severe limitation of the solution approaches. This was discussed in Section 1.2.2.

Work in the area of general robot path planning is more extensive, and largely benefits from the development of Ordered Upwind Methods. In addition to the foundational work on Fast Marching and OUMs, discussed in Sections 1.2.2 and 1.2.3 respectively, an heuristically-guided OUM called the FM* algorithm (inspired by A*), was presented in [56] specifically for the application of AUVs. [57] applies an OUM to solve a static HJB equation for the time-to-go function and globally optimal feedback trajectories of UAVs (unmanned aerial vehicles) in a 2D, time-invariant, but realistic model of a tornadic storm. Notably, the wind

speeds in fact exceed the speed of the UAVs, making the time-to-go function discontinuous, despite the fact that this is not allowed in any of the theory published by the inventors of OUMs. A rigorous investigation of how this effects the accuracy, and how OUMs might be formally extended to the discontinuous case, is of interest, and some introductory work in this direction is undertaken in [14].

The case of UAVs in realistically complex, strong, but time-invariant winds, in addition to obstacles, is also considered in [58], which carefully analyzes the effect of the strong winds on controllability. However the focus is ultimately on reducing the problem to a graph search by approximating the wind speed as constant on polygonal patches.

Finally, the work of [59], from the image processing literature, is highly relevant but does not fit well into any of the categories discussed thus far. Essentially, it uses Lagrangian marker particles to solve the Eikonal equation (1.16) for the distance from the boundary of an image, revealing the "skeleton" of the image defined by the shocks (cusps in the distance function) that result. The algorithm includes remeshing of particles along the tracked boundary/front, and allows for removal/trimming of particles at the shocks. More importantly, the algorithm determines the direction of motion of the particles stably via ENO discretizations of the gradient of the distance function on an auxiliary 2D Eulerian grid. This type of Lagrangian/Eulerian hybrid approach is of great interest, and is relevant for Chapter 5, but is reserved for future work.

1.6 Overview of Dissertation

Chapter 2 presents the minimum time results published in [24]. The algorithm of choice is an explicit (Lagrangian) front tracking method like that of [16]. Like in [16], the algorithm tracks the "reachability front" forward in time from a specified initial state-time $(\mathbf{x}_0, t_0)$, with the help of a "trimming" procedure for locally optimal trajectories. The resulting surface is the 2D boundary of the reachable subset of the 3D time-varying state space. Unlike in [16], it also includes a simple but important "remeshing" procedure. Most importantly, the flow is time-varying; the algorithm obtains globally optimal minimum time trajectories without solving the time-varying HJB equation or otherwise computing a 3D grid. We prove this rigorously and also demonstrate it empirically, by validating the results against those of the backward-in-time extremal field method of [23]. The main theme is the ability of unstructured Lagrangian meshes to capture singularities without numerical diffusion, via direct minimization of the multi-valued solution surface, and without the requirement of local controllability. This is in contrast for instance to the Eulerian and semi-Lagrangian OUMs, as discussed in Section 1.2.3 and even in [34], the final paragraph of which cites the case of locally non-controllable dynamics as one of key interest in future work.

Chapter 3 presents 1D and 2D results for the fixed final time "minimum energy" problem defined by $\mathcal{L} = W\mathbf{u} \cdot \mathbf{u}$ in (1.2), where the $W > 0$ is a weighting parameter. Some of the 1D results are described briefly in [60]. The 2D results

are unpublished. The HJB equation is solved directly in a semi-Lagrangian backward time-stepping approach similar to that of [13]; the boundary condition is simply $V(\mathbf{x}, t_f) = h(\mathbf{x}, t_f)$. Like in [13], our method effectively transforms the optimal control problem into a point-wise optimization problem. Unlike in [13], our method does not include higher than first-order discretizations for $V_{\mathbf{x}}$ and V_t. The purpose of this is to avoid iterative optimization methods solve the resulting (nonlinear constrained quadratic) point-wise optimization problem exactly. In addition, the main feature of our algorithm is the ability to resolve the shocks (cusps in V) very precisely using a structured adaptive triangular grid and a very simple refinement criterion. The grid is equivalent to that of [47], which extends to 3D and higher dimensions. Our method is used to demonstrate the basic features of minimum energy control in the context of analytically defined flows. However the computational complexity of the exact point-wise optimization proves fairly limiting, and motivates alternate approaches, semi-Lagrangian, Eulerian, and Lagrangian.

Chapter 4 presents a more in depth study of how the minimum energy control of Chapter 3 relates to the structure of the flow itself and how it might be approximated by greedy, local control laws. It includes 1D numerical results published in [60] and unpublished 2D numerical results. First, we propose a natural coordinate transformation for the optimal control problem based on the fixed final time flow map $\mathbf{X}$ defined in Section 1.3. In the transformed coordinates the

flow is zero, but the control input is multiplied by the Jacobian of the flow map $\mathbf{X}_{\mathbf{x}}$—a time-varying matrix. We then discuss the "pulled back end cost" function $H(\mathbf{x}, t) := h(\mathbf{X}(\mathbf{x}, t, t_f))$, its approximation of V for the case of $W = 0$ (not considered elsewhere in this dissertation), and the resulting class of greedy, local, suboptimal control laws. These laws are of the form $\mathbf{u}(\mathbf{x}, t) = -\tilde{s}\frac{H_{\mathbf{x}}}{|H_{\mathbf{x}}|}$ and referred to as "local Lagrangian control" (LLC) laws. We propose three LLC laws and focus on one in particular, comparing it to the optimal control computed in Chapter 3. Although our examples are only preliminary, we claim that the pulled-back end cost offers a much more practical alternative than the proposed use of finite time Lyapunov exponents of [37] and [38].

Chapter 5 presents a very brief 1D proof of concept for one additional algorithm: a Lagrangian alternative to the backward-in-time semi-Lagrangian algorithm of Chapter 3 for the minimum energy HJB equation. Like in the semi-Lagrangian algorithm, the core of the Lagrangian algorithm is the computation of a given time slice $V(\cdot, t)$ from the slice $V(\cdot, t + \Delta t)$, on an adaptive grid. The Lagrangian approach is to do this via a backward-in-time extremal field method—the method of characteristics described in Section 1.1.2. In order initialize the extremal field, we use a cubic/quadratic interpolation scheme that preserves the equivalence of $V_{\mathbf{x}}$ and the costate $\mathbf{p}$, subsampling the adaptive grid of $V(\mathbf{x}, t+\Delta t)$ extensively along the shocks. In order to simultaneously "remesh" and "trim" the unstructured extremal field (along which V is generally multi-valued) back onto

the structured grid of $V(\mathbf{x}, t)$, we again use cubic/quadratic interpolation (and direct minimization). In theory the Lagrangian approach is more efficient than the semi-Lagrangian approach in that it takes advantage of the a priori knowledge of $V_{\mathbf{x}} = \mathbf{p}$ and thus $\mathbf{u}$ at time $t + \Delta t$, but we have not been able to demonstrate this in practice. The challenge appears to lie in efficiently identifying the unstructured grid elements of the extremal field containing a given vertex of the structured grid at time t. It was also assumed that the Lagrangian approach has a fundamental advantage over Eulerian schemes such as Godunov methods due to Courant-Friedrichs-Lewy-like time step restrictions, but this remains unclear.

Chapter 6 briefly summarizes our conclusions and presents recommendations for future work.

Chapter 2

Minimum Time Trajectories: a Forward-in-Time Lagrangian Approach

2.1 Overview

This chapter presents the minimum time results published in [24]. The minimum time problem is equivalent to the one defined in Section 1.2, but with a couple of minor differences. First, the target is a fixed state $\mathbf{x}_f$; in other words, the terminal hypersurface in (1.3) assumes the form of the line $\mathbf{m}(\mathbf{x}, t) = |\mathbf{x} - \mathbf{x}_f|$. And second, the control $\mathbf{u}$ is defined in terms of the heading angle Θ (since $|\mathbf{u}| = s$ is known), and the vectors $\mathbf{x}$ and $\mathbf{v}$ are expressed using scalars. Example solutions focus on different combinations of spatial complexity, time-dependence, and strong currents, and include a numerically-defined flow field from a high resolution model of the Adriatic Sea.

The algorithm of choice is an explicit (Lagrangian) front tracking method like that of [16] and thus also related to the extremal field approach of Section 1.1.2. Like in [16], the algorithm tracks the "reachability front" forward in time from a specified initial state-time $(\mathbf{x}_0, t_0)$, with the help of a "trimming" procedure for locally optimal trajectories. Unlike in [16], it also includes a simple but important "remeshing" procedure, in which marker particle are frequently redistributed along the tracked front. Most importantly, the algorithm obtains globally optimal minimum time trajectories without solving the time-varying HJB equation or otherwise computing a 3D grid. We confirm the global optimality of the trajectories in three ways. We demonstrate it intuitively, by interpreting the computed 2D "extremal surface" as the boundary of the reachable subset of the 3D time-varying state space. We prove it rigorously, through the definition of a "relaxed" minimum time necessary condition. And we show it empirically by validating the resulting open-loop minimum time trajectories against the closed-loop minimum time trajectories of [23], obtained by actually solving the time-varying HJB equation (1.14) in a backward-in-time extremal field approach.

As mentioned previously, the advantage of the backward approach is the optimal feedback control law—essentially the negative gradient of the solution of the HJB equation. The disadvantage is the computational complexity. First of all, as shown in Figure 2.1, the backward-in-time, "controllability" approach of [23] is computationally equivalent to repeating the present forward-in-time "reachabil-

ity" approach for an entire family of arrival times t_f. Alternatively, the "controllability front" (not shown) is a 2D object, initialized at the line $\mathbf{x} = \mathbf{x}_f$, whereas the "reachability front" is a 1D object initialized at the point $(\mathbf{x}, t) = (\mathbf{x}_0, t_0)$. In addition to increased dimensionality, [23] does "remesh" the 2D controllability front (along 1D slices), but it does not attempt to "trim" away local optima; this further limits its efficiency. In short, for underwater gliders, which are incapable of communicating with the central controller more than once or twice an hour, it makes more sense to just recompute the solution using the more efficient algorithm (and updated flow field data) than to try to use continuous feedback control.

The main weakness of present forward-in-time algorithm is the trimming procedure for local optima. This procedure makes the algorithm significantly more efficient but also significantly more complicated, at least in the current implementation. This is the main disadvantage of Lagrangian methods like ours versus Eulerian methods such as Level Set Methods (LSMs) and Ordered Upwind Methods (OUMs), as discussed in Sections 1.2.1 and 1.2.3 respectively.

Though efficiency is necessary if the present time-varying method is to be competitive with other methods, the main theme of this chapter is the same as that of [23]: the unique ability of unstructured Lagrangian meshes to capture singularities with no numerical diffusion, via direct minimization of the multi-valued surface. In [23] the surface of interest is actually a time-slice of the solution of the dynamic HJB equation. Presently, it is simply the 2D boundary of the

reachable set in the 3D time-varying state space. In both cases, discontinuities—due to currents stronger than the vehicle—are just as easy to capture as the cusps known as shocks—due to locally optimal trajectories. This is in contrast for instance to the Eulerian and semi-Lagrangian OUMs.

The remainder of this chapter proceeds as follows. Section 2.2 states the minimum time problem. Section 2.3 describes the forward-in-time Lagrangian solution algorithm. Section 2.4 presents example solutions, including for a numerically-defined flow field from a high resolution model of the Adriatic Sea. Finally, Section 2.5 concludes the chapter.

2.2 Minimum Time Problem

Consider the following simple model for a vehicle moving in the 2D, time-varying vector flow field $(u, v) : \mathrm{R}^2 \times \mathrm{R} \to \mathrm{R}^2$:

$$\begin{aligned} \dot{x}(t) &= u(x(t), y(t), t) + s\cos(\Theta(t)); \\ \dot{y}(t) &= v(x(t), y(t), t) + s\sin(\Theta(t)), \end{aligned} \tag{2.1}$$

where the state $(x, y) = (x(t), y(t)) \in \mathrm{R}^2$ is the position of the vehicle at time $t \in \mathrm{R}$, the control input $\Theta = \Theta(t) \in \mathrm{S}^1$ is its "heading"—the angle of its motion relative to the water, and the fixed parameter s is the vehicle speed. Note that the vehicle speed s could alternatively be thought of as a second control input $s(t) \in [0, s_{max}]$, where s_{max} is the maximum speed. Minimum time control, however, would yield $s(t) = s_{max}$ for all time.

While simple, the model realistically captures the essential features of the given application. It does not include velocities as state variables because the spatio-temporal scales of interest are kilometers and hours. It assumes the vehicle speed s is fixed because underpowered vehicles such as gliders typically do operate at fixed speeds. It does not include z as a state variable, and the solution examples that we will present are all for 2D flow fields; however, if the vertical speed of the ambient flow is negligible compared that of the vehicle and the vehicle's depth is a pre-determined function of time $z(t)$, as with gliders, then one can readily extend both the model and solution techniques by considering

$$\begin{aligned} u(x,y,t) &:= \hat{u}(x,y,z(t),t); \\ v(x,y,t) &:= \hat{v}(x,y,z(t),t), \end{aligned} \tag{2.2}$$

which directly accounts for the effects of vertical shear on a given horizontal 3D flow field $(\hat{u},\hat{v})$. This idea will be addressed again in Section 2.5.

The problem of interest is to steer the vehicle from a given initial position (x_0,y_0) at time t_0 to a fixed target position (x_f,y_f) in minimum time, if it is reachable within some maximum allowed time $T^{\max}$. In other words, find a control input signal $\Theta : [t_0,t_f] \to \mathrm{S}^1$ and the resulting state trajectory $(x,y) : [t_0,t_f] \to \mathrm{R}^2$ of the system (2.1) that minimizes the cost functional

$$\int_{t_0}^{t_f} 1 dt, \tag{2.3}$$

or the final time $t_f \in [t_0, t_0 + T^{\max}]$, while satisfying the initial condition $(x(t_0),y(t_0)) = (x_0,y_0)$ and the end condition $(x(t_f),y(t_f)) = (x_f,y_f)$. This is

the classical "Zermelo's navigation problem". Note that the target is not really a point in the plane but rather a line in space-time, which will generally be referred to as the "target set". Moreover, as should become evident, the core of the present method could also be used with more general, time-varying target sets; however, this will not be discussed further, since it does contribute much conceptually and technically requires some derivation.

Although low-dimensional and conceptually simple, this problem captures the key difficulties of more general nonlinear optimal control problems. Most of these stem from the fact that the input vector is bounded. This makes the problem nonlinear, even if the flow field or the system being controlled is linear. In general, the only hope for a solution is a numerical one. Locally optimal trajectories abound, especially if the flow field itself is nonlinear and spatially complex. Currents exceeding the input bound mean loss of local controllability. Sometimes this makes it impossible to arrive at the target, much less follow a specific path. The time-variability of the flow field effectively increases the dimensionality of the state space from 2 to 3. The present method for minimum time trajectories is designed to treat such situations efficiently.

2.3 Algorithm

The present method may be thought of as an extremal field algorithm with remeshing, or as a marker particle method for front tracking. As mentioned, the

front of interest will be referred to as the *reachability front*, $\partial\Omega_{xy}(t_f)$. It is the boundary of the reachable set $\Omega_{xy}(t_f)$—the set of all positions that can be reached or passed through at a given time $t_f \in [t_0, t_0 + T^{\max}]$ by a vehicle released from the fixed position (x_0, y_0) at the fixed time t_0. In other words

$$\Omega_{xy}(t_f) := \{(x_f, y_f) \in \mathrm{R}^2 \quad | \quad \exists \Theta : [t_0, t_f] \to S^1 \quad | \quad (x(t_f), y(t_f)) = (x_f, y_f)\},$$

where $(x, y) : [t_0, t_f] \to \mathrm{R}^2$ is defined here as the solution of (2.1) with $(x(t_0), y(t_0)) = (x_0, y_0)$. Similarly, one has the set of reachable position-times

$$\Omega_{xyt} := \{(x_f, y_f, t_f) \in \mathrm{R}^2 \times [t_0, t_0 + T^{\max}] \quad | \quad (x_f, y_f) \in \Omega_{xy}(t_f)\}$$

and its boundary $\partial\Omega_{xyt}$ (less the interior of $\Omega_{xy}(t_0 + T^{\max})$), which we will call the *reachability surface*.

We state without proof the rather obvious fact that a minimum time trajectory must terminate at (one of the) temporally first point(s) of intersection between Ω_{xyt} and the target set. Moreover, such a point must lie in $\partial\Omega_{xyt}$. So $\partial\Omega_{xyt}$ is both the union of various snapshots in time of the reachability front $\partial\Omega_{xy}(t_f)$, embedded in (x, y, t) space, and the union of (the graphs of) minimum time trajectories. A schematic of $\partial\Omega_{xyt}$, its intersection with a linear target set, and the resulting minimum time trajectory was shown in Figure 2.1(a). The challenge is to track and record the reachability front in such a way that it yields a good numerical representation of the reachability surface and allows extraction of minimum time trajectories.

This will be done using marker particles. Particles evolve according to (2.1) with Θ chosen to make the trajectories optimal. The required Θ, as it turns out, is simply the angle of the front's normal direction. A naive particle method approximates Θ by finite differences between neighboring particles. This finite difference strategy breaks down in the presence of shocks—cusps on the inside of the reachability surface where minimum time trajectories collide and thus terminate. These appear as cusps in the reachability front, making the normal direction undefined and the finite difference approximation of Θ unstable. This instability is described thoroughly in [16], and in general the problem of shocks is well understood, especially in the context of the Level Set Methods and OUMs.

The key to a viable marker particle method is to track Θ along with x and y using the appropriate ODE. The resulting system of equations is

$$\begin{aligned} \dot{x} &= u(x,y,t) + s\cos\Theta; \\ \dot{y} &= v(x,y,t) + s\sin\Theta; \\ \dot{\Theta} &= -u_y\cos^2\Theta + (u_x - v_y)\sin\Theta\cos\Theta + v_x\sin^2\Theta, \end{aligned} \tag{2.4}$$

where, for readability, the dependencies of x, y, and Θ on time t are hidden, and $u_x = \frac{\partial u}{\partial x}(x,y,t)$, etc. are the partial derivatives of the flow field (u,v). The $\dot{\Theta}$ equation is derived in the Appendix 7.1, using Pontryagin's minimum principle from the calculus of variations. It constitutes a reduced-order version of the more familiar 2D co-state equation, and thus (2.4) are the well-known Euler Lagrange equations, for the special case of minimum time. Very similar $\dot{\Theta}$ equations are

derived in [7], in the context of minimum time control of a boat, and in [61], in the context of simulating the propagation of a flame.

Thus instead of tracking the (x, y) reachability front to compute the reachability surface we will track a curve in (x, y, Θ) space, which we will call the *extremal front*, to compute the aforementioned *extremal surface*. Whereas the reachability surface can be thought of as a set in the 3D (x, y, t) space, the extremal surface can be thought of as a set in the 5D $(\theta, t, x, y, \Theta)$ space—the graph of a function from the 2D space of initial angles $\theta = \Theta(t_0)$ and final times $t \in [t_0, t_0 + T^{\max}]$ to the 3D (x, y, Θ) space; individual trajectories of (2.4)—extremal trajectories—give θ slices of this function, and snapshots of the extremal front give time (t) slices. Alternatively, "extremal surface" may be used more loosely to refer to the projection of the hard-to-visualize 5D graph into one of the lower-dimensional subspaces, or to refer to the function itself, depending on the context. In particular, the (x, y, t) projection of the extremal surface is (a superset of) the desired reachability surface $\partial\Omega_{xyt}$. Having computed it on a triangular mesh, one can directly compute its intersection with the target set and extract the desired minimum time trajectory and open-loop control input Θ.

The difference between (this projection of) the extremal surface and the reachability surface is that the former also contains locally optimal trajectories, or, equivalently, that it contains curves of self-intersection points—exactly where the latter has shocks; this will be shown most clearly in Figure 2.5. Although in-

teresting, the space of locally optimal trajectories is much larger than that of the desired globally optimal trajectories and, as will be shown numerically, grows roughly exponentially instead of linearly with respect to $T^{\max}$. To improve the efficiency for larger $T^{\max}$, our method includes an optional procedure for, every few time steps, trimming the extremal front and thus computing a small subset of the full extremal surface that still contains the reachability surface. Let "extremal front" and "extremal surface" also refer to the trimmed versions. Immediately after trimming, (the (x, y) projection of) the extremal front is equivalent to the reachability front.

For conceptual purposes, it is useful to define the (minimum) *arrival time function*

$$T(x_f, y_f) := \min\{t_f \in [t_0, t_0 + T^{\max}] \quad | \quad (x_f, y_f, t_f) \in \partial\Omega_{xyt}\},$$

in other words as a minimization in time of the generally multi-valued reachability surface. For time-invariant flows, the arrival time T is the solution of the static HJB equation

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} \cdot \nabla T(x,y) + s|\nabla T(x,y)| = 1$$

with the boundary condition $T(x_0, y_0) = 0$. In this case the minimum time trajectory could be extracted via gradient descent from the arrival time function alone, without the need for the multi-valued reachability surface. This fact will be mentioned in some of the example solutions. But it is is really not useful, and

does not play a role in the present algorithm. This is in contrast to the optimal feedback control framework of the analogous static HJB equation (1.15) or the dynamic HJB equation (1.14) for the optimal time-*to-go* function, used in [23].

In summary, the present algorithm consists of two main parts—computing the triangular extremal surface, and extracting the minimum time trajectory for a specific target. These are described via pseudocode in Algorithm 1 and 2, respectively. For simplicity, Algorithm 1 does not include the trimming procedure, and Algorithm 2, as noted therein, lacks some of the optimizations included in the actual implementation, which have to do with minimizing the number of triangles of the extremal surface through which one loops. The comments provided in the pseudocodes, along with the description up to this point, are intended to be complete enough and clear enough to convey the overall flow of the algorithm. Thus we proceed to describe each of the procedures called in the pseudocodes, and then the trimming procedure and the modifications required for it. Note that the "IntegrateEulerLagrange", "Remesh", and "TrimExtremalFront" procedures are also conveyed schematically in Figure 2.2.

Procedure IntegrateEulerLagrange

This procedure is the core of the algorithm. It simply solves the minimum time Euler Lagrange ODEs (2.4) numerically over a given time interval for a vector of particles in the (x, y, Θ) state-costate space. Its role within the overall algorithm

is conveyed schematically in Figure 2.2(b). In Algorithm 1, it is used to evolve the entire extremal front over relatively short time intervals, to allow frequent enough remeshing. For example, in Section 2.4.5, the time interval used is 1 hour, and the overall time interval of the extremal surface is $T^{\max} = 54$ hours. We used the Matlab function "ode45"—a 4th order adaptive time step Runge-Kutta solver; however, such a high-order solver was probably unnecessary, given the frequency and simplicity of the current remeshing procedure.

For simpler flow fields and for the purpose of learning, the pure extremal field approach is nice. It doesn't include remeshing or trimming, so the ODEs can be integrated from t_0 to $t_0 + T^{\max}$ all at once. In that case whole extremal trajectories can be adaptively "inserted" into the computed extremal surface, the overall algorithm is simpler, and any error can be traced to the ODE solver. But in our experience this approach is not generally sufficient to accurately and efficiently resolve every part of the extremal surface. This is due to the sensitive nature of the extremal trajectories, which will be demonstrated thoroughly in Section 2.4.

Procedure Remesh

The purpose of this procedure is to control the distance between neighboring marker particles along the extremal front. It is necessary because, in a spatially complex flow field, the front can stretch dramatically over time. If trimming is used, the remeshing is applied one front segment at a time; this is conveyed in

Figure 2.2(c), though the trimming is not accounted for in the Algorithm 1 pseudocode. It is natural to use a distance metric defined in (x, y, Θ) space as opposed to (x, y) space. Then areas of high curvature in (x, y) space are automatically resolved by the higher density of particles placed there. In fact, the distance metric we use also depends on the time t. For particles 1 and 2 (or any two neighboring particles) at time t this distance is given by

$$d(x_1, y_1, \Theta_1, x_2, y_2, \Theta_2, t) := \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + W(\Theta_1 - \Theta_2)^2 \frac{st}{(st)^2 + 1}},$$

where W is a weighting parameter that may be tuned to the spatial scale of the problem. All examples in this paper will use $W = 1$. The scaling factor $\frac{st}{(st)^2+1}$ is not absolutely necessary, but was found very helpful in keeping the number of marker particles reasonable. It is depicted in Figure 2.3. The exact shape is not important, just the end behavior. The idea is to keep the $(\Theta_1 - \Theta_2)$ term from playing too much of a role in the initial stages, near where $(x_1 - x_2) = (y_1 - y_2) = 0$, and in the final stages, where resolving the rapidly growing, high-curvature areas of the front is not as important. Note that in the absence of flow st is exactly proportional to the length of the reachability front.

Two opposing remeshing schemes are proposed. The so-called "redistribution" scheme completely redistributes the particles along the curve, uniformly, with enough new particles added to maintain the same density as before. The so-called "insertion" scheme inserts particles only where needed, by bisection, in order to maintain an upper bound on the distances between them. The advantage of

the redistribution scheme is that it can require less overall particles, especially if there is any negative stretching in the front. One advantage of the insertion scheme is that it allows one to explicitly capture the actual extremal trajectories along which the marker particles evolve. Another advantage is that it performs far less interpolation, thus requiring less computational overhead and probably introducing less error.

Interpolation is performed with respect to the curve parameter given by the cumulative sums of distances along the curve. Either of the two remeshing schemes may be paired with linear, cubic, or any other interpolation scheme. That said, all of the present results used the cubic option. One option that has been explored but not rigorously tested is a special cubic interpolation scheme that preserves the consistency between the angle Θ and the actual normal direction of the interpolating curve in (x, y) space. Note that in addition to x, y, and Θ, the value θ of the initial heading associated with a given particle must also be stored and interpolated during the remesh step; this is needed for the purpose of extracting trajectories from the extremal surface.

Procedure GetConstrainedDelaunayTriangulation

This procedure computes a Delaunay triangulation of the given array of points (θ, t), constraining the triangles to the rectangular strips between which the points are assumed to lie. In Algorithm 1 the points lie along the the $N^t + 1$ time slices

for which the extremal front is recorded. The desired extremal surface function $(x, y, \Theta)(\theta, t)$ is defined via quadratic interpolation from the resulting triangular mesh. This will be shown for the numerical examples of Sections 2.4.3, 2.4.4, and 2.4.5 in Figures 2.7(a), 2.10(a), and 2.14(a), respectively. A more sophisticated approach might be to place the triangle edges between successive time slices by somehow minimizing the distance metric described in Section 2.3 along them. But Delaunay triangulation has proved sufficient so far and implementations are readily available.

Procedure GetBarycentricCoordinates

This procedure simply computes the solution $\mathbf{b} = (b_1, b_2, b_3)$ of a linear system of the form $(\mathbf{b} \cdot (\theta_1, \theta_2, \theta_3), \mathbf{b} \cdot (t_1, t_2, t_3), \mathbf{b} \cdot (1, 1, 1)) = (\theta, t, 1)$—i.e. the barycentric coordinates of the point (θ, t) with respect to the vertices of the given triangle. As should be evident from Algorithm 2, the barycentric coordinates are all in the interval $[0, 1]$ if and only if the point (θ, t) lies in or on the triangle, and in that case provide the interpolant needed to evaluate a function defined on the mesh containing that triangle.

Note that whereas the triangles of the extremal surface are well-defined in the (θ, t) plane, as suggested here, that is not the case in the (x, y) plane, since, recall, the reachability surface Ω_{xyt} is generally multi-valued in t. The triangle may lie on line, and in that case the output $\mathbf{b}$ is undefined. For simplicity algorithm 2

assumes **b** is defined. In the actual implementation all cases are considered and the proper (minimum) value $t(x, y)$ is obtained.

Procedure TrimExtremalFront

This procedure trims away the suboptimal segments of a given extremal front based on its projection into (x, y) space, as depicted schematically in Figure 2.2(d). To be clear, a point on the extremal front is defined as suboptimal if and only if it lies in the interior of the reachable set $\Omega_{xy}(t_f)$. And trimming is something done to extremal fronts, not to extremal surfaces. Hence the rules for trimming do not depend on whether the associated extremal surface is multi-valued. That said, it just so happens that the self-intersecting extremal fronts that get trimmed correspond to extremal surfaces that are multi-valued due to shocks, not the strength of the flow. And, finally, the more general reason for not trying to make the extremal surface single-valued or to otherwise trimming *it*, is that a general, time-varying target set like that alluded to in Section 2.2 could intersect it at *any* point.

The trimming procedure is optional and is omitted from the pseudocode of Algorithm 1 because it significantly complicates the overall algorithm without adding much conceptually. Instead of a single curve—an array of vectors, the extremal front object must be defined as a constantly growing sequence of smaller curves—an array of an array of vectors. If this abstraction is made, then the additional

pseudocode might include something like "$\left(\theta^{ef}, \mathbf{x}^{ef}, \mathbf{y}^{ef}, \Theta^{ef}\right)$ = TrimExtremalFront $\left(\theta^{ef}, \mathbf{x}^{ef}, \mathbf{y}^{ef}, \Theta^{ef}\right)$", enclosed in an "if" statement, and placed just before the call to "IntegrateEulerLagrange". The only algorithm parameter required is the number of times to trim N^{trims}, which is assumed to go evenly into the total number of time steps $N^t - 1$.

The benefit of trimming is that it can make the growth over time of the extremal front effectively linear instead of exponential. This will be shown for the numerical examples of Sections 2.4.4 and 2.4.5 in Figures 2.15 and 2.16, respectively. One benefit of not trimming, besides the algorithmic simplicity, is the ability to observe the local optimal trajectories.

The procedure is summarized as follows:

1. Compute the intersection points (x, y), both between and within the individual curves composing the (already trimmed) extremal front.

2. Insert one copy of each such point into each of the two line segments involved in the intersection, interpolating the two distinct values of Θ (and θ) separately.

3. Define the front, with new points added, as a directed graph, with unidirectional edges between the neighboring points within individual curves (including from the last point of the last curve to the first point of the first curve, if the values of θ there are still 2π and 0 respectively) and bidirectional edges between the newly added pairs of intersection points.

4. Find all cycles of the the directed graph. Compute the area encompassed by each of these "candidate" fronts using (a simple discretization of) the formula $A = \frac{1}{2}\int(y\frac{dx}{dl} - x\frac{dy}{dl})dl$, where l is any path parameter along it.

5. Assume the candidate front encompassing the largest area to be the desired reachability front. Declare all curves and curve segments not lying on it as suboptimal and remove them.

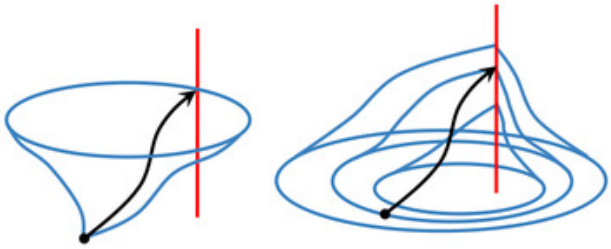
Note that the current procedure does not check the orientation of the candidate fronts in the final step above. This means that it cannot detect a candidate front that forms a "hole" in the reachable set, and will improperly remove such a hole. For flow fields much stronger than those considered in Section 2.4, this is a possibility.

Input: (x_0, y_0, t_0), // Initial position-time.
$T^{\max}$, // Maximum allowed arrival time.
N^t, // # of times t at which to remesh and record.
N_0^θ. // Initial # of marker particles and thus headings θ.
Output: $\left\{N_1^\theta, ..., N_{N^t}^\theta\right\}$, // # of marker particles at each time t.
N^{tri}, // # of triangles.
$\theta^{es}, \mathbf{t}^{es}, \mathbf{x}^{es}, \mathbf{y}^{es}, \Theta^{es}$, // Extremal surface vertices (vectors).
$\mathbf{v}^1, \mathbf{v}^2, \mathbf{v}^3$. // Triangles' vertex indices (vectors).
Data: t, t^{prev}, // Instantaneous time and previous time.
$\theta^{ef}, \mathbf{x}^{ef}, \mathbf{y}^{ef}, \Theta^{ef}$. // Instantaneous extremal front values (vectors).
/* Initialize extremal front with given initial # of particles, */
/* and headings from 0 to 2π: */
for $i = 0$ **to** N_0^θ **do**
  $\left(\theta_i^{ef}, x_i^{ef}, y_i^{ef}, \Theta_i^{ef}\right) = \left(2\pi\frac{i}{N_0^\theta}, x_0, y_0, 2\pi\frac{i}{N_0^\theta}\right)$;
/* Initialize extremal surface with initial extremal front: */
$(\theta^{es}, \mathbf{t}^{es}, \mathbf{x}^{es}, \mathbf{y}^{es}, \Theta^{es}) = \left(\theta^{ef}, [t_0; ...; t_0], \mathbf{x}^{ef}, \mathbf{y}^{ef}, \Theta^{ef}\right)$;
/* Track extremal front from t_0 to $t_0 + T^{\max}$, remeshing/recording: */
$t = t_0$;
while $t < t_0 + T^{\max}$ **do**
  $t^{prev} = t$;
  $t = t + \frac{T^{\max}}{N^t}$;
  $i = \frac{t - t_0}{T^{\max}} N^t$;
  $\left(\mathbf{x}^{ef}, \mathbf{y}^{ef}, \Theta^{ef}\right) = \texttt{IntegrateEulerLagrange}\left(\mathbf{x}^{ef}, \mathbf{y}^{ef}, \Theta^{ef}, t^{prev}, t\right)$;
  $\left(N_i^\theta, \theta^{ef}, \mathbf{x}^{ef}, \mathbf{y}^{ef}, \Theta^{ef}\right) = \texttt{Remesh}\left(N_{i-1}^\theta, \theta^{ef}, \mathbf{x}^{ef}, \mathbf{y}^{ef}, \Theta^{ef}, t\right)$;
  /* Record by appending vectors: */
  $(\theta^{es}, \mathbf{t}^{es}, \mathbf{x}^{es}, \mathbf{y}^{es}, \Theta^{es}) =$
  $\left(\left[\theta^{es}; \theta^{ef}\right], \left[\mathbf{t}^{es}; t; ...; t\right], \left[\mathbf{x}^{es}; \mathbf{x}^{ef}\right], \left[\mathbf{y}^{es}; \mathbf{y}^{ef}\right], \left[\Theta^{es}; \Theta^{ef}\right]\right)$;
/* Define extremal surface w/ triangles in (θ, t) plane, */
/* constrained to time slices: */
$(\mathbf{v}^1, \mathbf{v}^2, \mathbf{v}^3) = \texttt{GetConstrainedDelaunayTriangulation}\left(\theta^{es}, \mathbf{t}^{es}\right)$;

Algorithm 1: Compute triangular extremal surface.

Input: $(x_f, y_f), \Delta t^*,$ `// Target position, trajectory step size.`
$N^{tri}, N^{vert},$ `// # of extremal surface triangles, vertices.`
$\theta^{es}, \mathbf{t}^{es}, \mathbf{x}^{es}, \mathbf{y}^{es}, \Theta^{es},$ `// Vertices (vectors).`
$\mathbf{v}^1, \mathbf{v}^2, \mathbf{v}^3.$ `// Triangle vertex indices (vectors).`
Output: $\theta^*, N^*,$ `// Initial heading angle, # of time steps.`
$\mathbf{t}^*, \mathbf{x}^*, \mathbf{y}^*, \Theta^*.$ `// Trajectory (vectors).`
`/* Loop through triangles to find min` t `(and` θ`) for which`
$(x, y) =$ `*/`
`/*` (x_f, y_f) `(omitting sorting of triangles by` t `value): */`
$t^{min} = \infty$;
for $i = 0$ **to** $N^{tri} - 1$ **do**
  $(p, q, r) = (v_i^1, v_i^2, v_i^3)$;
  $\mathbf{b} =$ `GetBarycentricCoordinates` $\left(\left(x_p^{es}, x_q^{es}, x_r^{es}\right), \left(y_p^{es}, y_q^{es}, y_r^{es}\right), x_f, y_f\right)$;
  if $b_1 \in [0,1]$ & $b_2 \in [0,1]$ & $b_3 \in [0,1]$ **then** `// if` $(x_f, y_f) \in$ `triangle:`
    $t = \mathbf{b} \cdot \left(t_p^{es}, t_q^{es}, t_r^{es}\right)$; `// Get arrival time.`
    if $t < t^{min}$ **then** `// if needed update min` t `(and` θ`):`
      $t^{min} = t$;
      $\theta^* = \mathbf{b} \cdot \left(\theta_p^{es}, \theta_q^{es}, \theta_r^{es}\right)$;
`/* Loop through triangles to interpolate` $(x, y, \Theta)(\theta, t)$ `along */`
`/* trajectory (again, omitting sorting of triangles by` t
`value): */`
if $t^{min} < \infty$ **then**
  $N^* =$ `floor` $\left(\frac{t^{min}}{\Delta t^*}\right) + 1$;
  $\mathbf{t}^* = [0; \Delta t^*; 2\Delta t^*; ...; (N^* - 1)\,\Delta t^*; t^{min}]$;
  for $i = 0$ **to** N^* **do**
    $j = -1$;
    repeat `// loop until` $(\theta^*, t_i^*) \in$ `triangle:`
      $j = j + 1$;
      $(p, q, r) = (v_j^1, v_j^2, v_j^3)$;
      $\mathbf{b} =$
      `GetBarycentricCoordinates` $\left(\left(\theta_p^{es}, \theta_q^{es}, \theta_r^{es}\right), \left(t_p^{es}, t_q^{es}, t_r^{es}\right), \theta^*, t_i^*\right)$;
    until $b_1 \in [0,1]$ & $b_2 \in [0,1]$ & $b_3 \in [0,1]$;
    $(x_i^*, y_i^*, \Theta_i^*) =$
    $\left(\mathbf{b} \cdot \left(x_p^{es}, x_q^{es}, x_r^{es}\right), \mathbf{b} \cdot \left(y_p^{es}, y_q^{es}, y_r^{es}\right), \mathbf{b} \cdot \left(\Theta_p^{es}, \Theta_q^{es}, \Theta_r^{es}\right)\right)$;
else
  `display` ("Target position is not reachable within maximum allowed time");

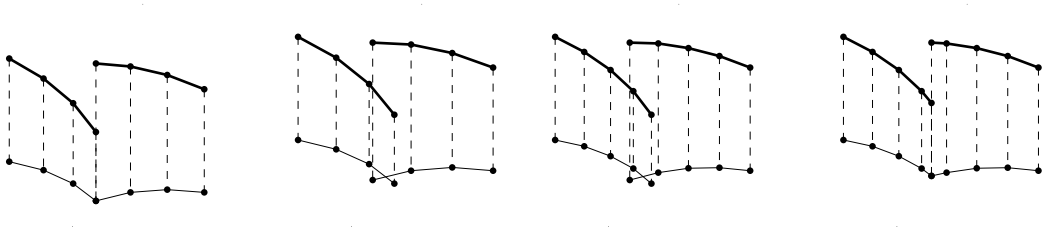
Algorithm 2: Extract min time trajectory from triangular extremal surface.



(a) Forward, reachability approach.

(b) Backward, controllability approach.

Figure 2.1: Schematics of the more computationally intensive backward method and the more efficient forward method. Both (a) and (b) show the same minimum time trajectory (in black, with an arrow), initial position-time (x_0, y_0, t_0) (as a black dot), and fixed target position (x_f, y_f) as a vertical line in space time (in red). (a) shows the reachability surface (in blue)—which contains a one parameter family of forward-in-time minimum time trajectories. As shown in (b), the prior method involves a two parameter family of backward-in-time trajectories, since the minimum time of arrival to the target set is not known a priori. The only benefit of the prior method is that it yields a feedback control law.



(a) Initial front.

(b) After solution of the Euler Lagrange ODEs, as described in Section 2.3.

(c) Remeshing (of the "redistribution" type), as described in Section 2.3.

(d) Trimming, as described in Section 2.3.

Figure 2.2: Schematics illustrating one time step of the front tracking algorithm. In each frame, the thick solid lines show (a segment of) the extremal front in (x, y, Θ) space, and the thin solid lines show its projection (along the thin dashed lines) into (x, y) space, i.e. the reachability front. The initial front here has already been trimmed and thus already contains a shock i.e. cusp, as indicated by the two marker particles with the same x and y values but distinct heading angles Θ. Note that these schematics do not attempt to convey the effect of the $\dot{\Theta}$ component of the Euler Lagrange ODEs, or the exact (linear or higher-order) interpolation scheme involved in the remeshing step.

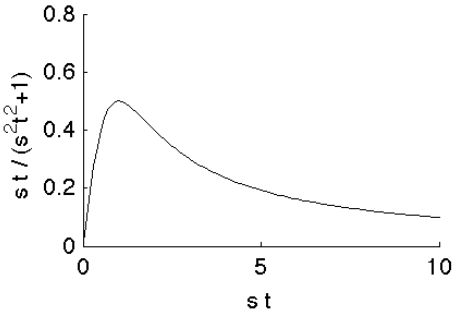


Figure 2.3: A scaling factor used in computation of distance along reachability front, described in Section 2.3.

2.4 Results and Discussion

This section contains example solutions for a few key flow fields. All of the flows considered are strong, i.e. faster than the vehicle speed s. Section 2.4.1 considers a time-invariant, spatially-uniform flow to demonstrate loss of controllability/reachability. Section 2.4.2 considers a time-varying, spatially-uniform flow for which controllability/reachability is lost locally but not globally. These first two examples are simple enough that the extremal trajectories $x = x(\theta, t)$, $y = y(\theta, t)$, and $\Theta = \Theta(\theta, t)$ are expressed in closed form. Section 2.4.3 considers a time-invariant, spatially varying "jet" flow field which produces a shock, i.e. locally optimal trajectories. Section 2.4.4 considers a time-invariant, spatially varying "gyre" flow which tests the limits of the trimming procedure with its abundance of locally optimal trajectories and tests the remeshing procedure with its extreme stretching of the front. Aspects of the each of the example flows are seen in Section 2.4.5 where we directly test our algorithm in the context of a spatially and tem-

porally varying ocean flow produced by a data-assimilating NCOM model of the Adriatic Sea. We verify the equivalence of the forward algorithm presented here and the backward algorithm presented in [23] in the ocean flow field by directly reproducing one of the minimum time trajectories.

2.4.1 Time-invariant, spatially-uniform flow

For spatially-uniform flow fields, the partial derivatives u_x, u_y, v_x, and v_y are all identically zero. Thus for all times t, $\dot{\Theta}(t) = 0$ and $\Theta(t) = \theta$, and the reachability front is simply a circle of radius $s \cdot (t - t_0)$ centered about the trajectory of a passively advected particle released from (x_0, y_0) at time t_0. This is true for all spatially uniform flows, even for aperiodically time-varying flows like that of Section 2.4.2.

Consider the time-invariant, spatially-uniform flow field given by

$$u(x, y, t) = u(x, y) = u = 1.5;$$

$$v(x, y, t) = v(x, y) = v = 0.0$$

and the problem parameters $(x_0, y_0) = (0, 0)$, $s = 1$, $T^{\max} = 2$, $t_0 = 0$, and $(x_f, y_f) = (0.83, 0.74)$. Solving the Euler Lagrange equations (2.4) analytically yields the following closed-form expression for the extremal trajectories

$$x(t) = (1.5 + \cos\theta)t;$$

$$y(t) = (\sin\theta)t;$$

$$\Theta(t) = \theta.$$

These were obtained numerically using the algorithm parameters $N_0^\theta = 50$, $N^{\text{trims}} = 1$ (since no trimming is necessary), and $N^t = 20$ with the "insertion" remeshing scheme.

Figure 2.4(a) shows the extremal surface in space-time (x, y, t), in this case equivalent to the reachability surface. For strong flows like this one, the surface is multi-valued. Figure 2.4(b) shows the arrival time function T, with dashed lines to show the "hidden" segments of the reachability front. Figure 2.4(a) shows the target as a vertical line. Both figures show the extracted optimal trajectory. Its length in time is $T = t_f = 1$, and the optimal heading angle is $\Theta = \theta \approx 2.3$. Due to the strong eastward flow T is infinite except for in a cone opening to the right. The extracted trajectory was chosen to lie precisely on the boundary of this cone.

Due to the time-invariance of the flow T is still the solution of a static HJB equation in the interior of the cone, and thus the control vector is known to be $s\frac{\nabla T}{|\nabla T|}$. On the boundary of the cone, however, $s\frac{\nabla T}{|\nabla T|}$ is only defined in the limit from the inside, in which it is normal to the contours of T, and thus normal to the extracted trajectory. The practical significance of this singularity is that the cost T is finite but not robustly so: the slightest perturbation would cause the vehicle to miss the target completely. The present method captures these special trajectories correctly and thus nicely highlights this key property of minimum time control in strong flows.

More important practically is that it captures the infinitely many other trajectories in the interior of the cone. As explained thoroughly in [30], OUMs would break down here, since the gradient ∇T is unbounded. One could use another Eulerian method, but would have to discretize in both space and time, even though the problem is time-invariant. The present, particle-based method is well-suited for the application at hand because it computes the arrival time function just as easily in a strong or time-varying flow as in a weak, time-invariant flow.

Lastly, this example reveals extremal trajectories that are not locally optimal trajectories: those corresponding to the dashed lines. These are similar to locally optimal trajectories, and we tend to refer loosely to all extremals as "locally optimal", but they are not locally optimal here. In fact, they may be thought of as "maximum time" trajectories, since, for instance, the leftmost of these trajectories ($\Theta = \pi$) corresponds to the strategy of swimming directly away from its target while still passing through it. However, we cannot trim away these hidden segments of the reachability front. If the flow were to reverse direction, the reachability surface would double back on itself and these extremal trajectories would become optimal for targets in the left half plane. This is the case in the next example, which focuses on a more typical oceanic field where the instantaneous flow may exceed the vehicle speed, even though the mean flow does not.

2.4.2 Sinusoidally time-varying, spatially-uniform flow

Consider the sinusoidally time-varying, spatially uniform flow field given by

$$u(x, y, t) = u(t) = 0.5 + 2\cos(2\pi t);$$

$$v(x, y, t) = v(t) = 0.5 + 2\sin(2\pi t),$$

and the exact same problem and algorithm parameters as in the example of Section 2.4.1 except with the target position $(x_f, y_f) = (0.0, -0.5)$. In this case, the extremal trajectories are given by

$$x(t) = (0.5 + s\cos\theta)t + 2\tfrac{1}{2\pi}\sin(2\pi t);$$

$$y(t) = (0.5 + s\sin\theta)t + 2\tfrac{1}{2\pi}\left[1 - \cos(2\pi t)\right];$$

$$\Theta(t) = \theta.$$

Figure 2.5 shows the solution for this example just as Figure 2.4 did for the example of Section 2.4.1. The minimum time is $T = t_f \approx 1.77$ and the optimal control is $\Theta = \theta \approx 4.38$. The optimal trajectory is initially pushed away from the target, goes through 1.77 periods of the time-varying flow, and eventually arrives, on the second pass (although the control is constant throughout).

The arrival time function exhibits several discontinuities but is finite everywhere. Holding a given course is impossible, but reaching the target and reaching it quickly is merely a matter of finding the right θ.

The effect of a discontinuity in this case is that a trajectory may come very close to the target then be forced back again before finally arriving. As in Section 2.4.1, this implies singular trajectories, but here perturbations will only result in

finite jumps in cost. Unlike in Section 2.4.1, the arrival time function is not the solution of a static HJB equation. This is evidenced by the fact that example trajectory crosses over one of the discontinuities.

2.4.3 Time-invariant, spatially varying "jet" flow

Spatial non-uniformity in a flow field produces deformation in the otherwise circular reachability front. These deformations may lead to shocks in the reachability surface—the cusp-like singularities associated with locally optimal trajectories.

Consider the flow field

$$u(x, y, t) = 1 + B(y);$$

$$v(x, y, t) = 0,$$

where B is the smooth bump function given by $1 - \cos(2\pi y)$ if $y \in [0, 1]$ and 0 otherwise. This flow field is illustrated in Figure 2.6(b) by the sequence of black streamlines spanning the y range of the plot near $x = 1$.

The problem parameters are $(x_0, y_0) = (0, -0.25)$, $s = 2$, $T^{\max} = 1.5$, and $t_0 = 0$, and the two target positions (x_f, y_f) are $(0, 1.25)$ and $(3.25, -0.25)$. The algorithm parameters are $N_0^\theta = 50$, $N^{\text{trims}} = 1$ (although some trimming could be done), and $N^t = 15$ with the "insertion" remesh scheme.

Figure 2.6(a) shows the extremal surface and the two optimal trajectories in (x, y, t) space. Unlike in Figure 2.5(a), the surface does not "fold over" itself, because the flow is not time-varying, but it does intersect itself, due to the spatial

variability of the flow, and thus it is not equivalent to the reachability surface. In addition to the streamlines of the flow, Figure 2.6(b) shows the arrival time function, reachability front (solid), extremal front (dashed), two optimal trajectories and a locally optimal trajectory (a straight line) to the second target $(3.25, 0)$. Figures 2.7(a)—2.7(d) show (θ, t) plots of the triangular mesh and the x, y, and Θ values on it, overlaid with (projections of) the 2 extracted (globally) optimal trajectories.

The first target point, $(0, 1.25)$, is located directly across the jet, to the north of the initial point, (0,-0.25). The arrival time is $t_f \approx 1.20$. The initial control angle is $\theta \approx 2.6$. Over the course of the trajectory the heading control smoothly decreases to a minimum of about 2.0 at the center of the jet (i.e. the vector shown points north-northwest) and then smoothly increases back to 2.6. The resulting path nicely anticipates the effect of the jet. It is qualitatively and mathematically similar to the paths shown in Figure 3(A) of [49].

The second target point, $(3.25, -0.25)$, is located directly downstream or to the east of the initial point, (0,-0.25). The arrival time is $t_f \approx 1.02$. The control angle is initially $\theta \approx 1.0$ but decreases smoothly to a final value of about -1.0, as the resulting trajectory enters and leaves the jet. The strategy is clearly to use the jet to get downstream faster. Note that the vehicle is initially well outside of the jet, and therefore cannot sense it locally. It must know the entire flow field and plan ahead, based on how fast the jet is moving versus how far off course one

must go to use it. The alternate strategy is to stay on course and direct all of one's control effort towards the target. This is the strategy of the depicted locally optimal trajectory, which arrives in time $t_f = (x_f - x_0)/(s+1) = 3.25/3 \approx 1.08$. The first strategy is a slight favorite, since $1.02 < 1.08$. Note that Figure 2.7 shows how extremal trajectories using the jet to get downstream means extreme sensitivity of x, y, and Θ to the initial angle θ. The practical consequence of this is the need for remeshing.

The shock is the curve of target points for which both strategies are equally optimal. This curve is in the middle of an entire region of target points for which both strategies exist (but for which one is globally optimal and the other is only locally optimal). Here, this is the cone-shaped region containing the dashed segments. The shock is not shown explicitly but is indicated by the cusps where the reachability front is not smooth. Although the flow field is faster than the vehicle at the center of the jet, the arrival time function does not have discontinuities. This is because the initial position is located away from the region of strong flow and the reachability front advances such that it is never directly opposed to the jet. Therefore the component of the flow that is directed down the gradient of the arrival time function never exceeds the vehicle speed.

2.4.4 Time-invariant, spatially complex "gyre" flow field

Consider the time-invariant, spatially complex "gyre" flow field given by

$$\begin{aligned} u(x,y,t) = \quad u(x,y) = \quad & -\sin(\pi x)\cos(\pi y); \\ v(x,y,t) = \quad v(x,y) = \quad & \cos(\pi x)\sin(\pi y) \end{aligned}$$

and shown in Figure 2.8. It consists of a grid of saddle points connected by heteroclinic orbits and unit grid cells containing nonlinear gyres of alternating orientation.

The problem parameters are $(x_0, y_0, t_0) = (0.5, 0.5, 0)$ (the center of a gyre), $s = 0.75$ (compared to a maximum flow magnitude of 1.00), and $T^{\max} = 3$. Twelve target positions (x_f, y_f) are spaced evenly along the circle of radius two centered at (x_0, y_0). The algorithm parameters are $N_0^\theta = 50$, $N^{\text{trims}} = 12$, and $N^t = 60$, with a "redistribution" remesh scheme.

Figure 2.8 shows a close up of the flow field itself and 3 of the 12 optimal trajectories. Figure 2.9(b) shows the arrival time function, the 12 instances of the reachability front obtained by trimming the extremal front, and all 12 optimal trajectories. Figure 2.9(a) shows the (trimmed) extremal surface, the 12 linear target sets, and the optimal trajectories, in (x, y, t) space. Figures 2.10(a)—2.10(d) show (θ, t) plots of the triangular mesh and the x, y, and Θ values on it, overlaid with (projections of) the 12 extracted (globally) optimal trajectories.

The optimal trajectories match intuition. Initially, they spiral out from the elliptic fixed point, where the arrival time function resembles the stream func-

tion. Then they systematically work their ways outwards, taking advantage of the fast flow along the separatrices but also taking care to choose the correct side, often steering transverse to the flow. Control vectors are always normal to the reachability front, and thus, for time-invariant flow fields like this one, to the level sets of the arrival time function. Due to the strength of the flow field, both jump discontinuities and shocks (loss of differentiability) appear in the arrival time function.

Perhaps the most important thing to notice here is the sensitivity along optimal trajectories of x, y, and Θ to the initial heading θ. In particular, all globally optimal trajectories of length $T > 2.75$ are accounted for by 8 very small intervals of θ, roughly $[1.25, 1.28]$, $[1.40, 1.43]$, $[2.83, 2.86]$, $[2.97, 3.00]$, $[4.39, 4.42]$, $[4.54, 4.57]$, $[5.96, 5.99]$, and $[6.11, 6.14]$, in other words, by approximately $8 \times 0.03/6.28 = 4\%$ of all possible values of θ. This is most precisely shown in Figure 2.10(a). Even the 12 chosen trajectories (for which $T \in [1.75, 2.50]$) are initially concentrated in 4 distinct groups, before gradually separating to reach their separate target positions (in Figure 2.9(b)).

This is the sensitivity repeatedly alluded to. It makes the pure extremal field approach at best inefficient and at worst totally ineffective. In this approach one may use the values of x, y, and Θ observed along the computed trajectories to adaptively sample values of θ but does not insert new trajectories away from (x_0, y_0, t_0) or otherwise remesh the corresponding marker particles along the ex-

tremal front. For spatially complex flow fields and large enough $T^{\max}$, it's not uncommon for the difference in θ between neighboring trajectories to be on the order of double precision, e.g. 10^{-16}. If the target set intersects the region of the reachability surface between two such trajectories, then it can be impossible to reach it numerically. Even if one finds the precise value of θ needed, very small numerical integration errors can perturb one onto one of the neighboring trajectories. This is why remeshing like that described in Section 2.3 is generally necessary. Trajectories or marker particles must be placed adaptively, not just at time $t = t_0$ but as they evolve over time and as the front they are tracking stretches.

Figure 2.15 shows the number of marker particles needed to represent the front as it grows over time. The number grows at an approximately exponential rate in the intervals between applications of the trimming procedure. Overall, the trimming procedure reduces the number of marker particles to an approximately linear function of time. By definition, the suboptimal segments of the extremal front that we avoid computing are all contained within the interior of the reachable set $\Omega_{xy}(t_0 + T^{\max})$. Thus the steep reduction in particle numbers via trimming implies an abundance of locally optimal trajectories in addition to those indicated by the shocks visible in Figure 2.9(b).

2.4.5 Spatially and temporally variable flow field: Adriatic Sea

To test the applicability of the method in flows with spatial and temporal variability of oceanic relevance, we consider path planning in vector fields produced by a typical ocean circulation model. Specifically, we study the output from the data-assimilating NCOM model of the Adriatic Sea presented in [62], configured specifically for the DART observational programs of 2005-2006 described in [63]. The model domain is approximately 250 km by 900 km by 240 hrs with nominal horizontal resolution of 1 km and velocity fields archived hourly. The Lagrangian transport properties of the model have been studied and validated against available drifter observations in [64] and [65]. Here we consider the subset $[110, 250]$ km $\times[210, 320]$ km $\times[0, 54]$ hrs of the horizontal surface velocity field. The gridded data is interpolated cubically in space and linearly in time, with central differences used to approximate the velocity gradients u_x, etc. in the $\dot{\Theta}$ component of (2.4). Figure 2.11 shows the $t = 0$ snapshot of the full surface flow field and the approximate area of the subset in model-based grid coordinates.

The problem parameters are $(x_0, y_0, t_0) = (187, 259, 0)$, $s = 0.9$ km/hr (25 cm/s—the approximate forward speed of a Spray glider, according to [4]), and $T^{\max} = 54$ hrs. Thirteen target positions (x_f, y_f), enumerating counter-clockwise, are located at (205,240), (240,255), (240,280), (235,288), (230,310), (188,295), (185,295), (145,285), (120,275), (120,230), (150,215), (185,225), and (189,271).

The initial position and the 13th final position (189,271) were chosen to coincide with those for one of the four minimum time trajectories computed in [23] using the backward-in-time solution of the full dynamic HJB equation (1.14) (in which the value function $V = T$ is not the time-to-arrive but rather the time-to-go). The algorithm parameters are $N_0^\theta = 50$, $N^{\text{trims}} = 9$, $N^t = 54$ with the "redistribution" remesh scheme.

The presence of a coastline requires a modification of the algorithm as fully described in Appendix 7.2. Simply, a smooth, spatially dependent indicator function, $g : \mathrm{R}^2 \to [0, 1]$, is used to explicitly drive the vehicle speed to zero outside the domain. The resulting generalization of (2.4) is

$$\begin{aligned}
\dot{x} &= u(x, y, t) + g(x, y) s \cos\Theta; \\
\dot{y} &= v(x, y, t) + g(x, y) s \sin\Theta; \\
\dot{\Theta} &= -u_y \cos^2\Theta + (u_x - v_y) \sin\Theta \cos\Theta + v_x \sin^2\Theta \\
&+ (g_x s \sin\Theta - g_y s \cos\Theta).
\end{aligned} \tag{2.5}$$

Figure 2.12 shows snapshots of the flow field, extremal front, and optimal trajectories. Figure 2.13(b) shows the arrival time function, 18 instances of the extremal front (with untrimmed segments hidden by the folds in the extremal surface), and the optimal trajectories. The solution exhibits the discontinuities like those seen in Sections 2.4.1, 2.4.2, and 2.4.4, and shocks like those seen in Sections 2.4.3 and 2.4.4. The flow field is strong and time-varying, and thus the extremal surface folds over itself like the one seen in Figure 2.5(a). The reachability

front stabilizes along the coastline; however, the non-smooth model representation of the coastline leads to rapid growth in the number of self-intersections there, which cannot be seen in any of the Figures.

Figures 2.14(a)—2.14(d) show plots of the triangular mesh and the x, y, and Θ values on it, overlaid with (projections of) the 13 extracted (globally) optimal trajectories. Whereas in the gyre example of Section 2.4.4 there are 8 distinct intervals of θ that account for the entire boundary of the reachable set of (x, y) positions (and all 12 of the 12 chosen trajectories), in the present example there is just one such interval, $[5.82, 5.89]$, and it accounts for not all but about half of that boundary (and 6 of the 13 chosen trajectories). There is much less sensitivity to θ elsewhere in the boundary of the reachable set. The jet appearing right-center in Figure 2.12(a) is the cause of this sensitivity to the initial angle θ, i.e. the extreme stretching of the front. The action of this jet is similar to that of the separatrices in the gyre flow field of Section 2.4.4. Here, however, the structure is time-varying. Again, the sensitivity/stretching inherently limits the pure extremal field technique and points to the importance of remeshing in the present method.

As seen in Figure 2.16, the effect of the trimming away globally suboptimal trajectories is less dramatic in the present example than is was in the time-invariant but more spatially complex example in Section 2.4.4. While trimming still results in a significant decrease in the number of marker particles, more particles were used overall, and the effective growth of the front with the trimming appears to

be slightly faster than linear. This growth is largely due to the large number of particles needed to resolve behavior along the coastline.

Validation of forward algorithm vs. backward algorithm

Recall that instead of a 1D reachability front parameterized by θ, the backward extremal front is a 2D controllability front parameterized by θ and t_f. It's important to mention that there was no truly 2D adaptive sampling, but rather 1D adaptive sampling of each t_f slice; hence the "t_f layers" suggested schematically in Figure 2.1(b). Because of this, many t_f layers and thus many time steps were used—4 per hour, versus the 1 per hour of the present, forward result. $T^{\max}$ was 24 instead of 54 hours but trimming was not used and t_f ranged from 0 to 72, not just 0 to 24, making for $4 \times 72 = 288$ t_f levels. Also note that the backward result did include the same cubic "redistribution"-type remeshing. Overall, the backward computation took on the order of 3 hours, versus 20 minutes for the forward computation. Both were done on a basic 2009 MacBook Pro laptop. More importantly, both algorithms were implemented in Matlab (C++ for instance is expected to be much faster), and neither is fully optimized; most of the computational time is spent interpolating the time-varying flow field, rather inefficiently, from the 3D grid. Thus it's more telling to consider the numbers of marker particles. As shown in Figure 2.16, there were 1014 particles in the $t_f = 24$ hour time slice of the forward extremal field, versus 12074 in the relevant

time slice, $t_0 = 0$, of the backwards extremal field. That's roughly the equivalent of 1014 specific extremal trajectories of length 24 hrs versus 12074 trajectories of various lengths between 0 and 24 hours.

Figure 2.17 shows additional snapshots of the same flow field and reachability front shown in Figure 2.12, but zooming in on the 13th optimal trajectory. Figure 2.18 again shows the same reachability front and optimal trajectory produced by the present method (i.e. the heavy black markers), superimposed with time slices of the time-to-go function $T(x_0, y_0, t_0)$ (indicated by the colors) and optimal trajectory from [23] (the thin black curve). The control is normal to both the reachability front and the level sets of the time-to-go function. Thus the two are tangent to one another. The two versions of the trajectory match nicely. This helps validate the present, forward method.

An mentioned in Section 2.1, part of the motivation for choosing the forward approach over the backward approach is that gliders, which for maximum efficiency travel in long (several hour) dive cycles, cannot benefit much from the continuous feedback control offered, optionally, by the backward approach. They can only get GPS fixes and communicate with the controller upon surfacing. On the other hand, 48 hours, for instance, is too long to go without feedback. Besides the error in the present algorithm, there are a host of other errors, the most important of which is presumably in the forecasted flow field. And although 72 hours is a typical forecast horizon, forecasts are updated with newly assimilated data

multiple times a day. Thus ideally, the algorithm should be used in a receding horizon approach, where as frequently as every dive cycle, the forward extremal surface and minimum time trajectory are re-computed using not only the updated vehicle position, but the updated flow forecast. The computer responsible for the flow forecast, which of course does not depend on the glider position, could conceivably compute the extremal surface in a matter of 1 or 2 minutes, especially with an optimized, C++ implementation of the algorithm. Although the full-length (e.g. 48 hour long) optimal trajectories are needed no matter what, for dive cycles of, say, 4 hours, the glider would only need to download the open-loop control signals it needs in order to execute the first 4 hours worth of the updated (48 hour long) heading signal $\Theta(t)$. Even if the glider software requires holding the heading constant at some set point value for each of these 4 hour dive cycles, our intuition is that the glider would still benefit significantly from an initially 48 hour (12 dive cycle) long optimal trajectory.

Robustness to perturbations

In practice one of the main concerns is the robustness of planned paths to perturbations. The first two types of perturbations that come to mind are errors or uncertainty and the flow field and errors or uncertainty in the vehicle state. Given that perturbations of the former type are harder to generate, we focus here on perturbations of the latter type. In particular, for each of the 13 forward-

computed (open-loop) optimal trajectories presented above, we re-simulate the flow and control for two small concentric circles of initial states around the original initial state $(x_0, y_0) = (187, 259)$.

Figure 2.19 shows the results. The trajectories are shown in blue, with black markers placed every three hours. The two circles of perturbed initial states and the resulting curves of final states are shown in black. The final states are highly sensitive to the initial state when there is stretching in the flow itself.

This is most evident along the five rightmost trajectories, all of which depart very close to one another along the jet shown in Figure 2.12. As described previously, this jet is associated with local strain in the flow, just to the north. This strain is responsible for much of the stretching of the circles along the five trajectories. In addition, the bottommost trajectory also experiences a large amount of sensitivity i.e. non-robustness due to the spatial complexity of the flow along it. In contrast, the four leftmost trajectories do not experience as much stretching, and thus the circles are not as deformed.

This is a preliminary result, but demonstrates the importance of having some kind of feedback in the control loop. As described at the end of Section 2.4.5, a continuous feedback control e.g. via the backward-in-time method of [23] is desirable but probably not necessary. The time horizons of the control are on the order of days, and the forward-in-time algorithm is fast enough to allow a receding

horizon control framework. This is expected to fully correct the non-robustness visible in Figure 2.19.

2.5 Summary and Outlook

In this chapter we presented a front tracking method for identifying globally optimal minimum time trajectories in spatially complex, time-varying ocean flow fields. The "reachability front"—the boundary of the reachable set—is defined explicitly using marker particles. These are evolved not in the (x, y) state space but in (x, y, Θ) state-costate space, where the (reduced-order) costate Θ is exactly the optimal vehicle heading—the angle normal to the front. The method includes remeshing and trimming procedures, both of which have proved necessary to handle realistically complex flow fields. Finally, instead of attempting to compute the arrival time function directly on a regular spatial grid, the method defines this function indirectly as a minimum taken over the triangular faces of the multi-valued "reachability surface". Thus the method sharply captures (a) discontinuities where currents exceed the fixed vehicle speed and (b) shocks or cusps in the arrival time function due to spatial complexity in the flow. The method allows accurate extraction of optimal trajectories directly from the reachability surface without having to directly solve the underlying dynamic HJB equation.

The method was used to characterize solutions for several simple flow fields which directly highlight various aspects of the minimum time control framework.

The time-invariant flow field of Section 2.4.1 made clear the fact that finding (or determining the non-existence of) a path from one point to another is inherently a globally optimal control problem. The time-varying flow field of Section 2.4.2 demonstrated how it may be hopeless to stay on a specific path but at the same time quite easy to reach the target. In Section 2.4.3, we showed how locally optimal trajectories highlight opposing control strategies and manifest themselves as self-intersections in the "extremal front" and cusps in the reachability surface. Spatially complex flows, as shown in Section 2.4.4, result in extreme sensitivity of optimal trajectories to initial heading and point out the inherent inadequacy of pure extremal field (and shooting) methods. Optimal trajectories for disparate targets typically depart along a few initial trajectories and remain quite close for considerable times before separating. Accordingly, there are many locally optimal trajectories that quickly fall behind the global reachability front and are rendered globally sub-optimal. This fourth example demonstrates the need for (a) remeshing and (b) trimming as a part of any efficient particle-based method for optimal control.

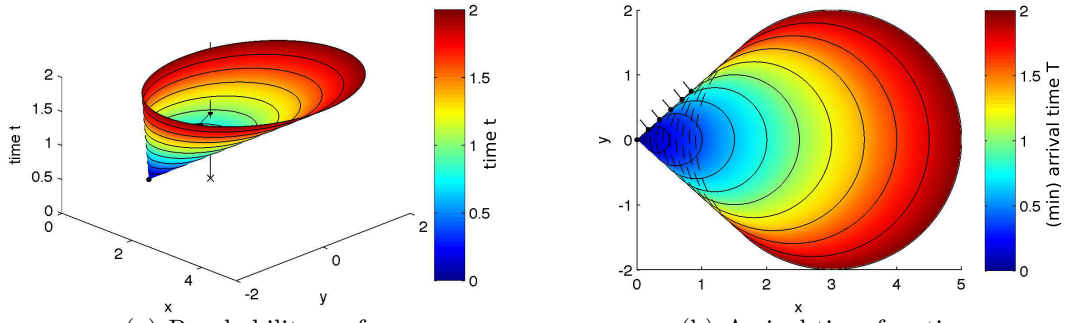
In Section 2.4.5, the proposed method was applied to the horizontal flow field produced by a typical, high resolution ocean model. The results indicated that the solution behavior produced by combining time variability like that of Section 2.4.2 with spatial variability like that of Section 2.4.4 is not much more complex than the sum of the parts. By including standard interpolation and differencing

schemes to incorporate gridded model data and a spatially dependent indicator function in the control to represent the presence of coastline, the method was readily extended to flow fields with both the magnitude and spatial/temporal variability of the ocean.

The present method allows for the consideration of depth-dependent planar flows, as shown in Equation (2.2). The assumption of a fixed vertical profile $z(t)$ is typical for gliders whose vertical velocity typically far exceeds that of the ambient ocean flow. Since gliders are often used specifically to sample frontal zones where depth dependence is the strongest, including the effects of vertical shear in the horizontal flow is a natural and straight-forward extension of the present approach. There is one minor caveat: the minimum time t_f of arrival to a target (x_f, y_f) may results in $z(t_f) \neq 0$, i.e. below the surface. In the worst case this means the glider must slightly overshoot the target. But it motivates a number of fixed final time problems, accessible via the present method of computing the reachability surface, that might also be of interest for gliders. Given a fixed saw-tooth dive profile, one idea would be to navigate in a more incremental, semi-closed-loop fashion. Choose the fixed time as some multiple of the time of a single dive. Compute the reachability surface and then choose from among the continuum of minimum time trajectories based on any number of cost functions—for instance, the final distance from a target set, the sampling quality of the trajectory, or even

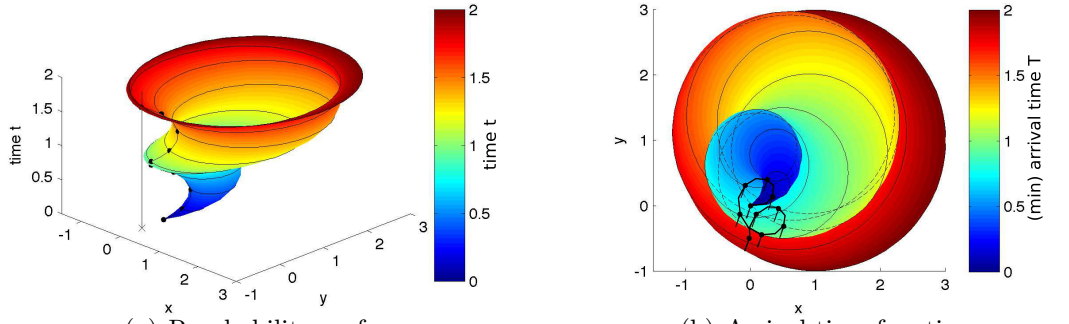
the ensuing drifter trajectory. Then drift, recharge, and/or communicate with the central controller.

Computationally, several possible extensions/modifications to the current trimming procedure could be considered. The present procedure does not allow for holes to develop in the reachable set, and it does not perform optimally in the presence of highly irregular gridded coastlines. Moreover, the trimming algorithm does not trivially extend to higher dimensions (for example, minimum energy control). The increase in dimensionality is also unavoidable if one is actually interested in the solution of the underlying dynamic HJB PDE—the time-to-go function—and the optimal feedback control as in [23], where one tracks a 2D "controllability front" in (x, y, t) space. One option is to perform 1D trimming within each time slice of the front. Another option might be to introduce numerical viscosity to prevent self-intersections from even forming. Yet another option is to pursue a Level Set Method or another Eulerian or semi-Lagrangian method for direct solution of the HJB PDE.



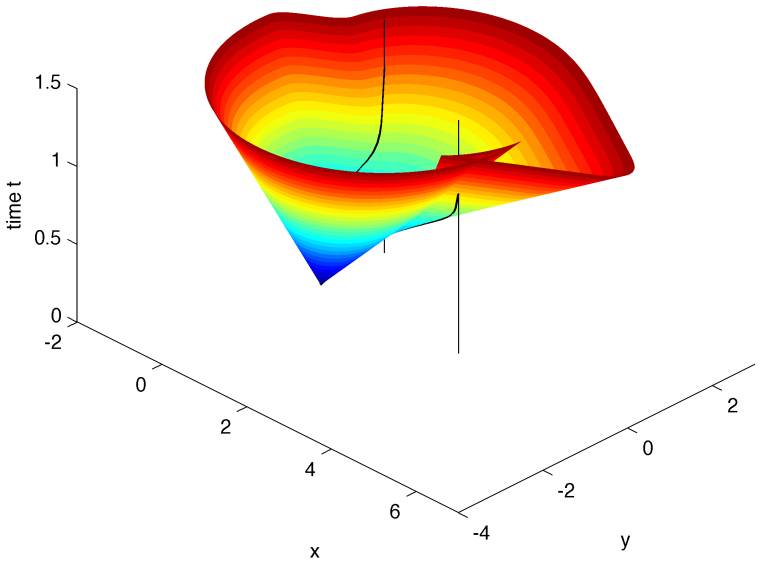
(a) Reachability surface. (b) Arrival time function.

Figure 2.4: Solution and the 1 extracted minimum time trajectory for the time-invariant, spatially uniform flow example of Section 2.4.1 (flow directed to the right, at 1.5× the speed of the vehicle). (a) shows the reachability surface, intersecting the linear target set in space-time. (b) shows the arrival time function $T(x, y)$ (the minimization of the reachability surface) and the 2D projection of the same minimum time trajectory, which begins at $(0, 0)$ and reaches the target, near $(1, 1)$, in 1 unit of time. The vehicle heading is indicated by a needle like arrow for each of the heavy black position markers. The dashed lines show the hidden segments of the reachability front. For this particular example the reachability surface is multi-valued and the arrival time function has two lines of discontinuity where it jumps to infinity—due purely to the strength of the flow, one of which contains the vehicle trajectory. The control is always normal to the contours and thus in this case normal to the path itself.

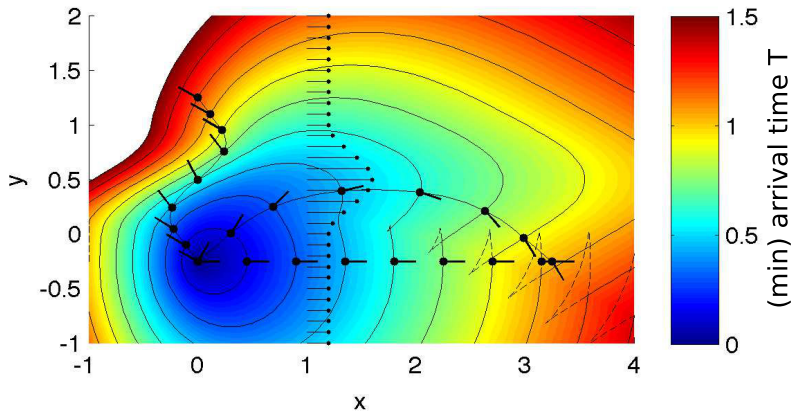


(a) Reachability surface. (b) Arrival time function.

Figure 2.5: Solution and the 1 extracted minimum time trajectory for the sinusoidally time-varying, spatially uniform flow example of Section 2.4.2. (a) and (b) show the same features as in Fig. 2.4. The arrival time function has discontinuities due to folds in the reachability surface. The vehicle is pushed back by the flow, recovers, is pushed back again, and crosses its path before reaching the target at $(0, -0.5)$ (just below the initial position). The heading is constant, as with any spatially uniform flow.



(a) Extremal surface (projected into (x,y,t) space).



(b) Arrival time function.

Figure 2.6: Solution and the 2 extracted minimum time trajectories for the "jet" example of Section 2.4.3. (a) and (b) show roughly the same features described in the Fig. 2.4 caption. But here, the extremal surface intersects itself in (x, y, t) space, and thus is not exactly equivalent to the reachability surface; the dashed lines in (b) show the portions of the self-intersecting extremal fronts that are not included in the reachability fronts. Moreover, (b) also includes 1 purely locally optimal trajectory—a straight line—to highlight the effect of the shock marked by the self-intersections; the corresponding globally optimal trajectory uses the jet to arrive slightly faster (T = 1.02 vs 1.08).

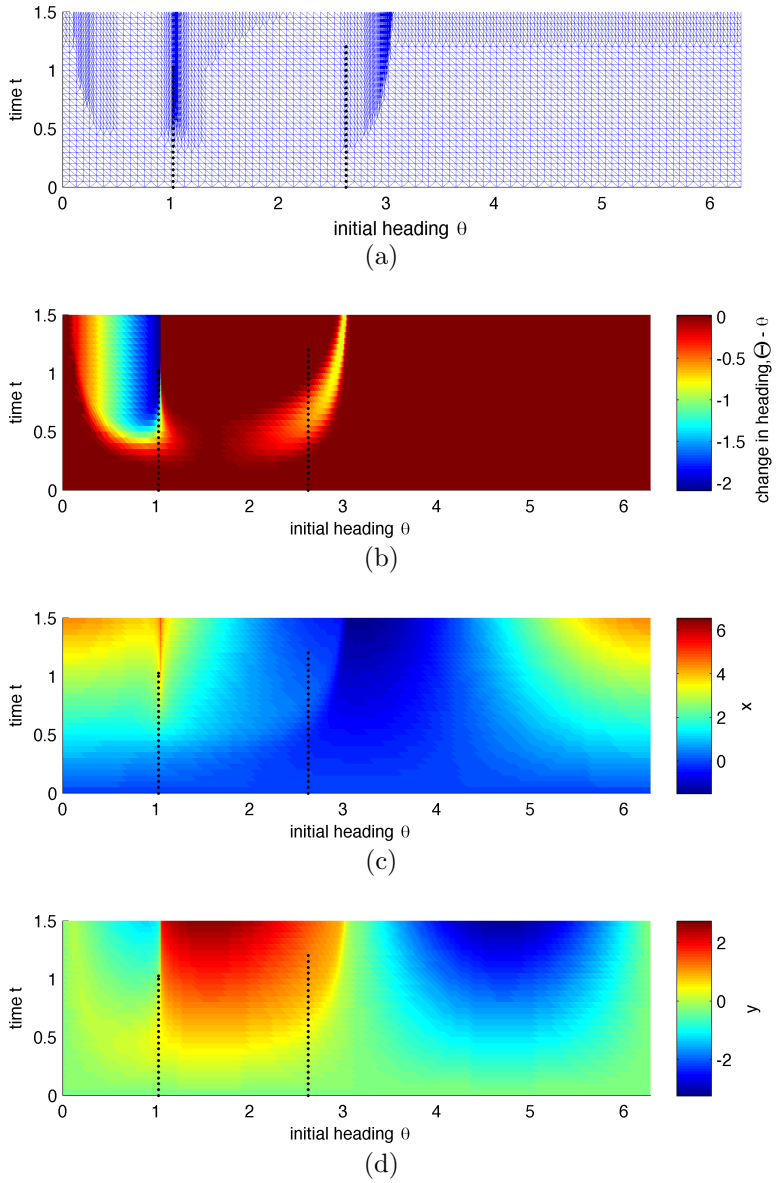


Figure 2.7: Parameterization of extremal surface (x, y, Θ) by initial heading θ and time t for the "jet" example of Section 2.4.3, and the 2 extracted minimum time trajectories. (a) shows the triangular mesh. The color values in (b), (c), and (d) show the change in heading $\Theta - \theta$, x, and y, respectively. The extreme sensitivity of Θ and y to θ and the slight peak in x near $\theta = 1$ are due to them variety of trajectories (like the first one shown) that use the jet to get downstream faster. The heading stabilizes upon exiting the region $y \in [0, 1]$. Trimming was not necessary since purely locally optimal trajectories were few.

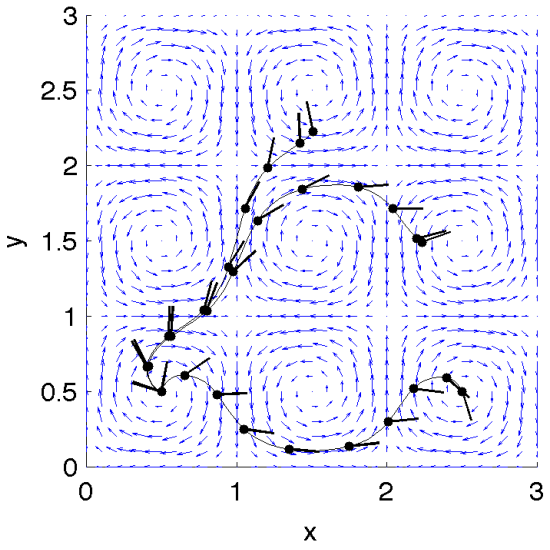
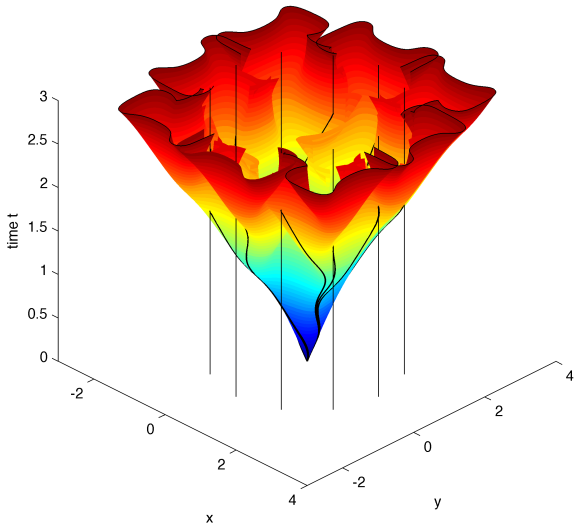
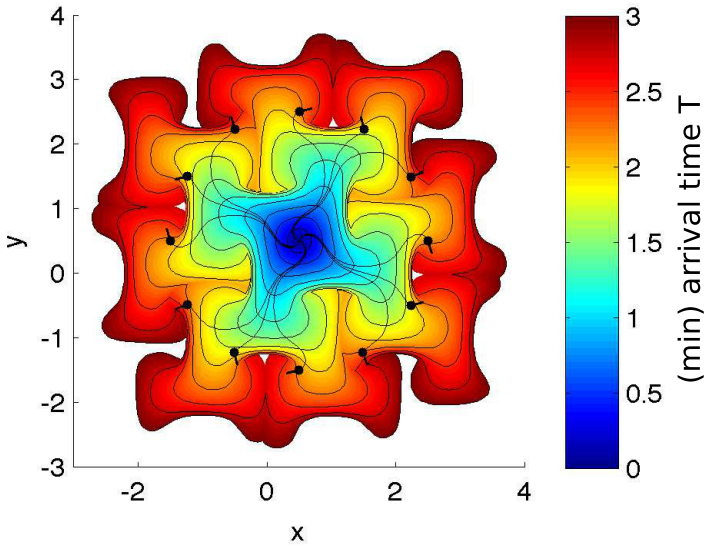


Figure 2.8: Spatially complex "gyre" flow field and 3 of the 12 optimal trajectories from Section 2.4.4, zoomed in. Control vectors are sometimes normal to the flow and sometimes tangential.



(a) (Trimmed) Extremal surface (projected into (x,y,t) space).



(b) Arrival time function.

Figure 2.9: Solution for the "gyre" example of Section 2.4.4 and the 12 extracted minimum time trajectories, zoomed out. (a) shows the (trimmed) extremal surface (projected into (x,y,t) space) and the 12 linear target sets in space-time. (b) shows the arrival time function (indicated by color) and final vehicle headings. The 12 extracted trajectories are initially concentrated along just a few initial headings. Most extremal trajectories lead to non-smooth shocks, beyond which they are suboptimal.

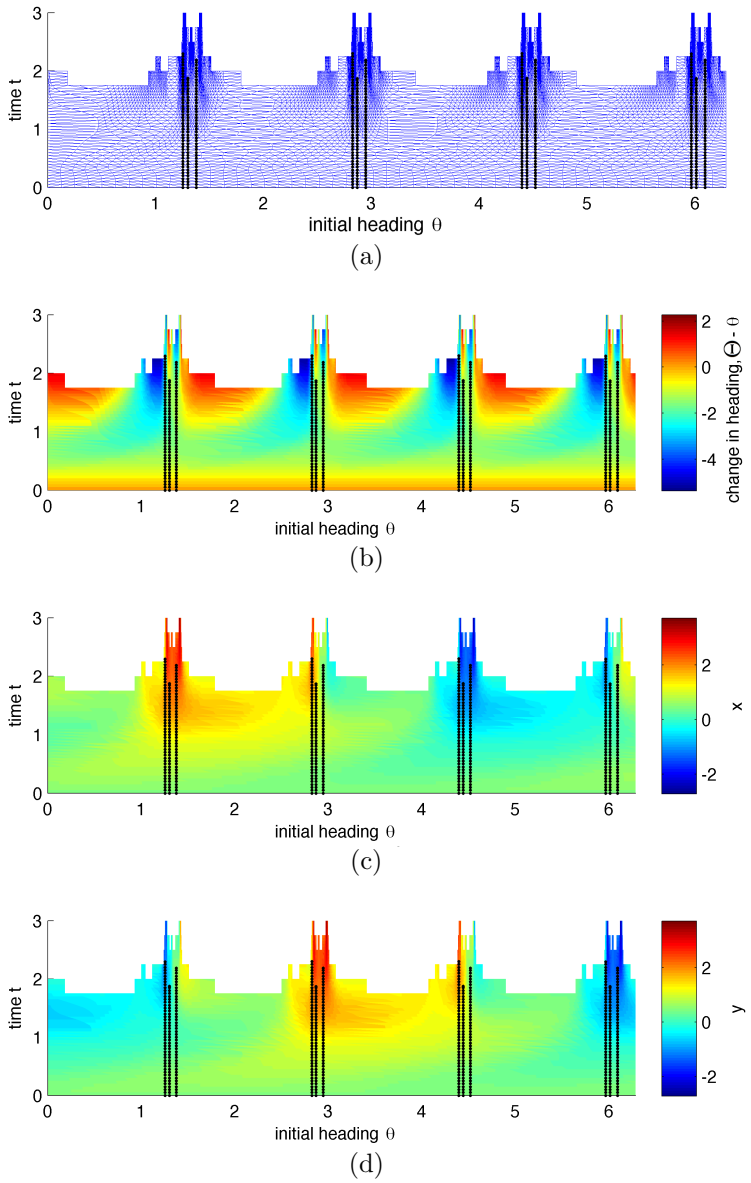


Figure 2.10: Parameterization of the (trimmed) extremal surface (x, y, Θ) by initial heading θ and time t for the "gyre" example of Section 2.4.4, and the 12 extracted minimum time trajectories. (a) shows the triangular mesh. The color values in (b), (c), and (d) show the change in heading $\Theta - \theta$, x, and y, respectively. The heavy trimming reveals the concentration of globally optimal trajectories around a very small subset of initial heading angles θ. This is due to the spatial complexity of the flow.

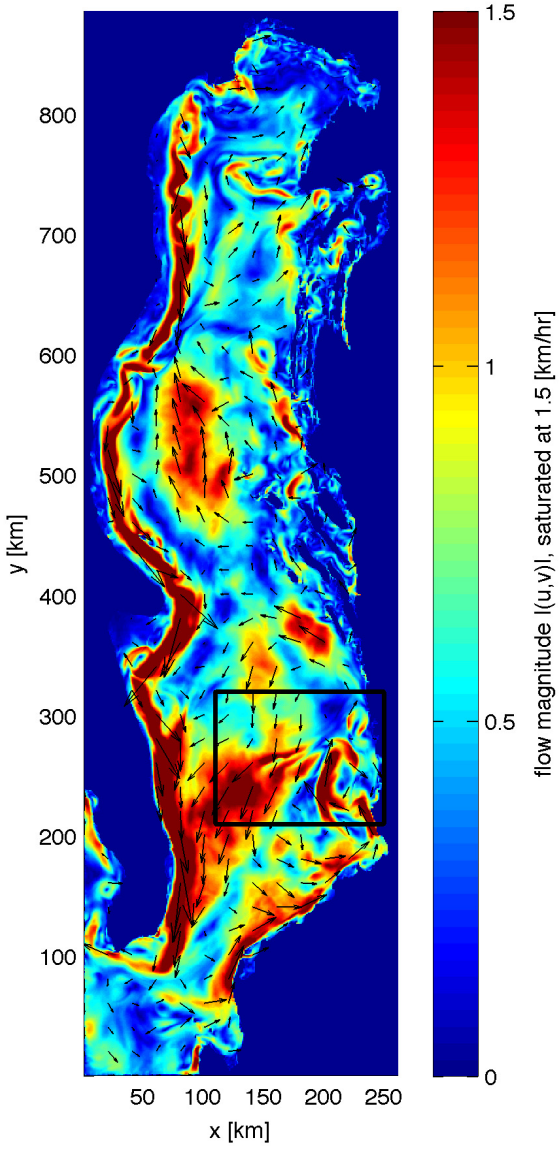
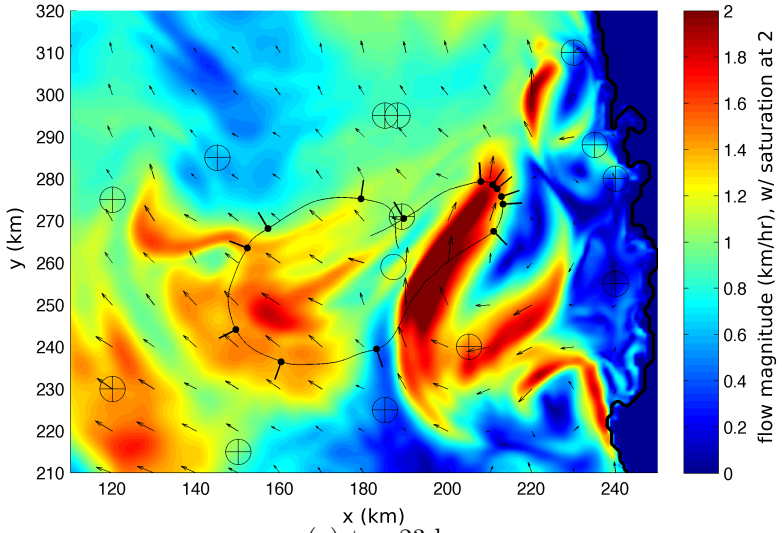
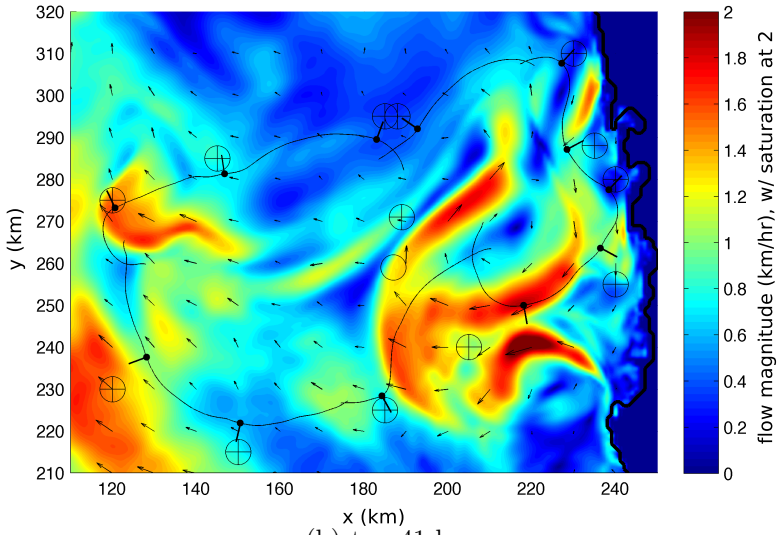


Figure 2.11: Snapshot of the entire time-varying Adriatic flow field at time $t = 0$ hrs. Approximate area of example solutions is indicated by the rectangle.

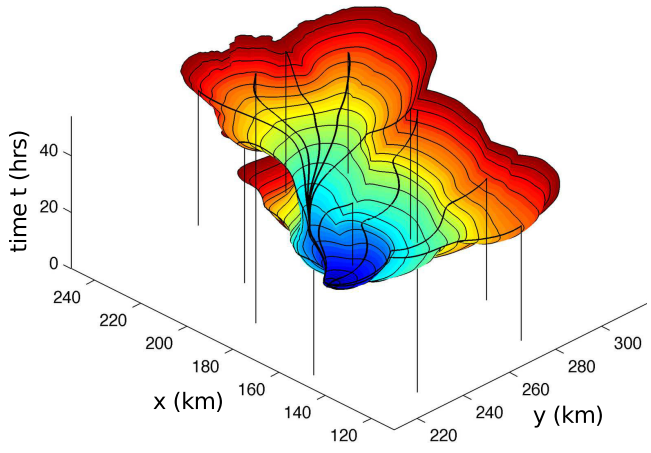


(a) t = 23 hrs.

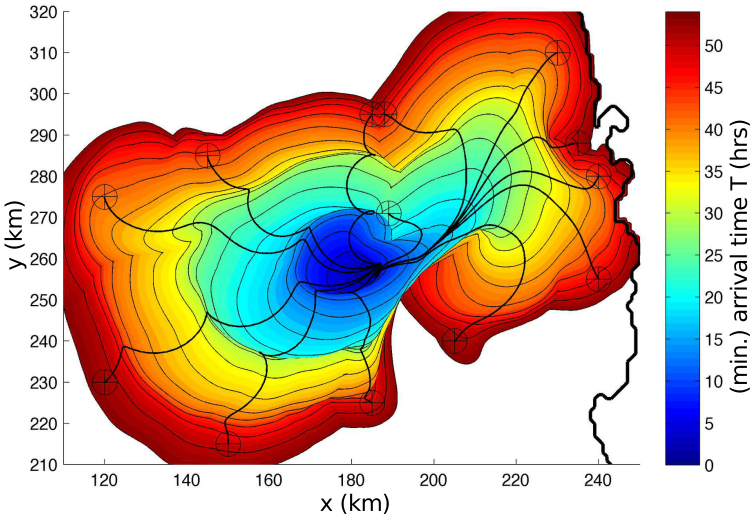


(b) t = 41 hrs.

Figure 2.12: Snapshots of the time-varying Adriatic flow field of Section 2.4.5, extremal front (trimmed 9 times total), and 13 minimum time trajectories. The jet appearing right-center in the first snapshot dramatically stretches the front, and is used by 6 of the 13 trajectories to reach their various targets. The center most trajectory is the one matching one of those from [23], which will be shown in more detail in Figures 2.17 and 2.18. The crosshairs mark the targets and the empty circle marks the initial position. A movie of all the hourly recorded snapshots is provided as supplemental material.



(a) (Trimmed) Extremal surface (projected into (x,y,t) space).



(b) Arrival time function.

Figure 2.13: Solution for Adriatic example of Section 2.4.5, and the 13 extracted minimum time trajectories. (a) shows the (trimmed) extremal surface and target sets in space-time. (b) shows the arrival time function $T(x, y)$, crosshairs at the target positions, and an open circle at the initial position. The top-center pair of trajectories highlights the presence of a shock. The arrival time function has a pair of discontinuities emanating from the initial position, like those seen in Figure 2.4(b) of Section 2.4.1, due to the strength of the flow field there initially, and a more pronounced discontinuity to the southeast, like that seen in Figure 2.5(b) of Section 2.4.2, due to a strong change in the flow field over time.

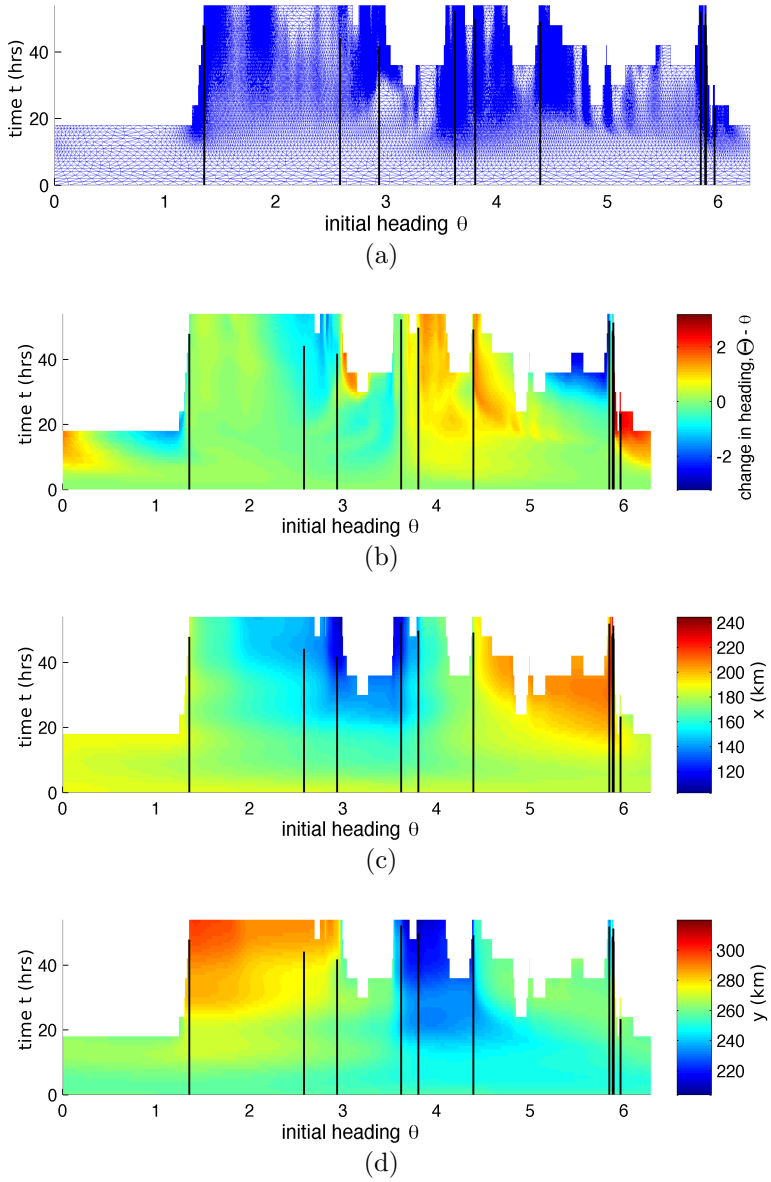


Figure 2.14: Parameterization of the extremal surface (x, y, Θ) by initial heading θ and time t for the Adriatic example of Section 2.4.5, and the 13 extracted minimum time trajectories. (a) shows the triangular mesh. The color values in (b), (c), and (d) show the change in heading $\Theta - \theta$, x, and y, respectively. The heavy trimming reveals the concentration of many of the globally optimal trajectories (including about 6 of the 13 actually extracted) around $\theta \approx 5.9$—corresponding to the jet seen in Figure 2.12(a). Θ, x, and y are extremely sensitive here, because the jet is fairly dominant, like the one in the time-invariant example of Section 2.4.3, shown in Figures 2.6(b) and 2.7.

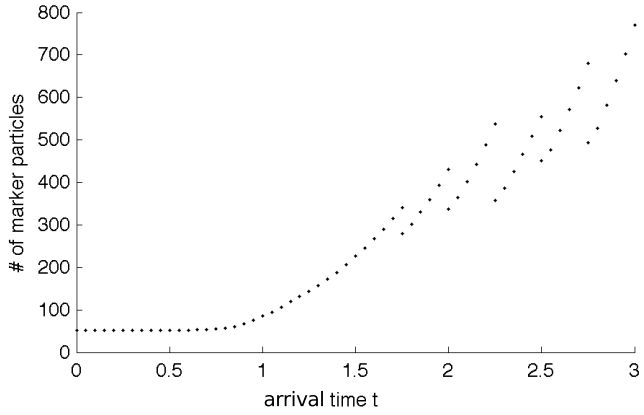


Figure 2.15: Growth with respect to time of trimmed extremal front in terms of number of marker particles, for the spatially complex gyre flow field of Section 2.4.4. Trimming makes what would be exponential growth effectively linear.

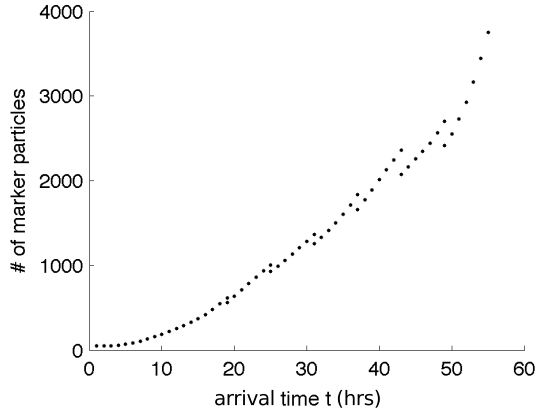
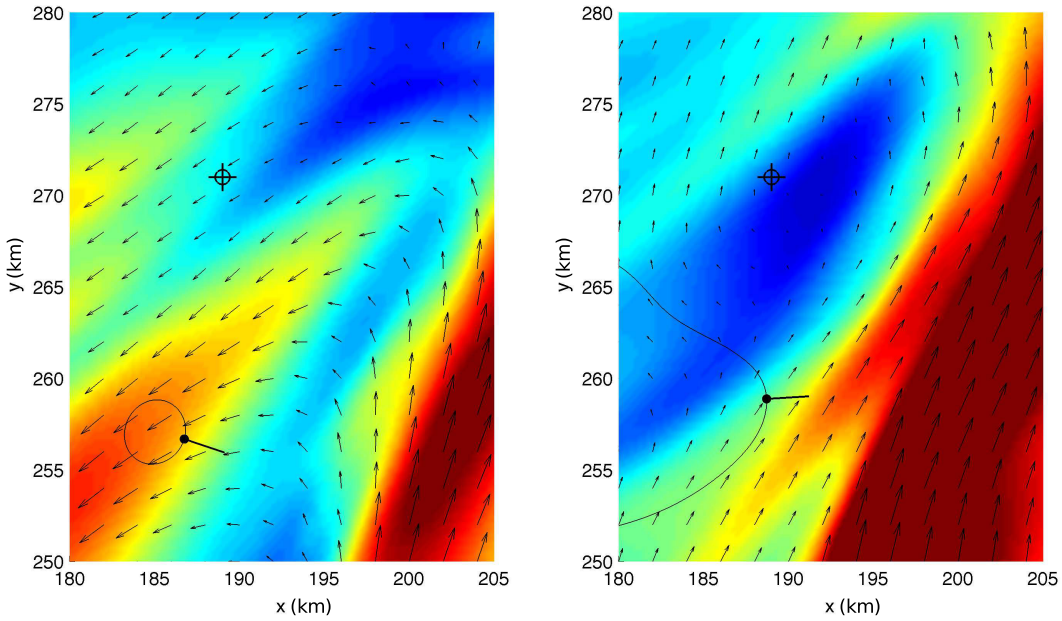
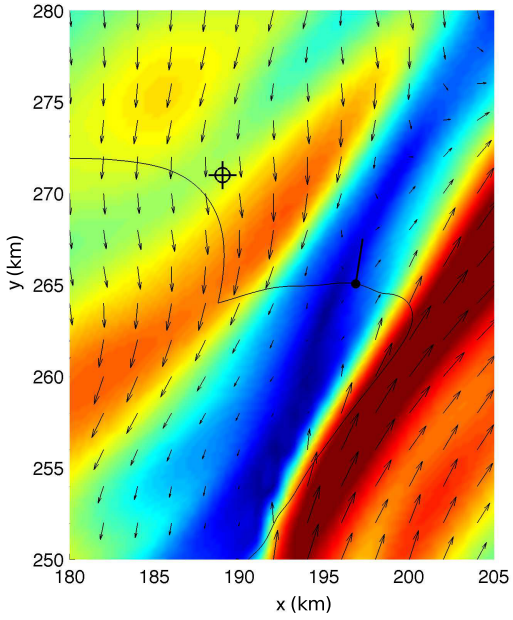


Figure 2.16: Growth with respect to time of trimmed extremal front in terms of number of marker particles, for the Adriatic example of Section 2.4.5. Trimming makes what would be exponential growth nearly linear. The effect is not quite as dramatic as for the "gyre" example of Section 2.4.4, shown in Figure 2.15, since the spatial complexity of the Adriatic flow field is not as extreme as that of the gyre flow field. But more particles are needed here, largely due to the stabilization of the front along the jagged coastline.

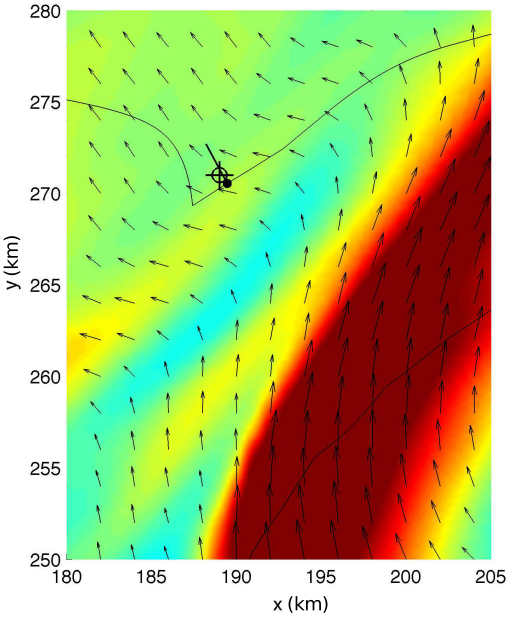


(a) t = 2 hrs.

(b) t = 9 hrs.

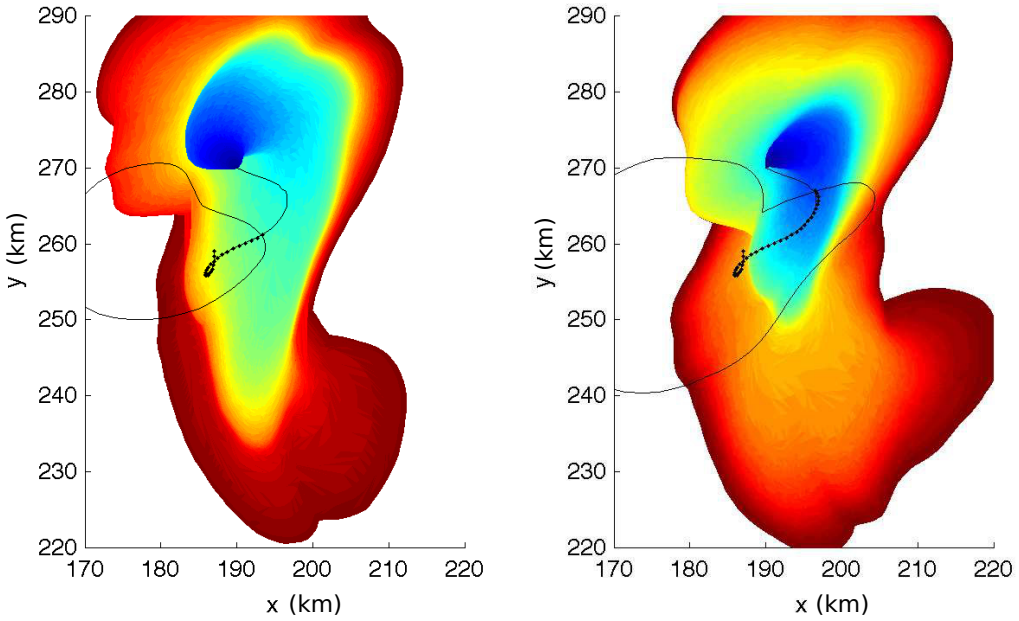


(c) t = 16 hrs.



(d) t = 23 hrs.

Figure 2.17: Additional snapshots of the flow field and the reachability front shown in Figures 2.12 and 2.18, zoomed in on the 13th optimal trajectory, which is the one reproducing the result of [23]. The color indicates the magnitude of the flow field, with saturation at 2 km/hr (red).



(a) Time t = 12 hrs.

(b) Time t = 18 hrs.

Figure 2.18: Two snapshots in time of the compared present (forward-in-time) and prior (backward-in-time) results described in Section 2.4.5. The closed curve is the reachability front of the present result, which is also shown in Figures 2.12 and 2.17. The color indicates the optimal *time-to-go* from the prior result of [23]. It goes from 0 (blue) at the target point to 24 hours (red). Points outside the colored region are not controllable to the target within 24 hours. The heavy black markers show the optimal trajectory from the present result, obtained by extraction from the reachability surface. The thin black curve shows the one obtained from the prior result, by feedback simulation. The control is known to be normal to both the reachability front and the level sets of the time-to-go function. Accordingly, the two are tangent to one another.

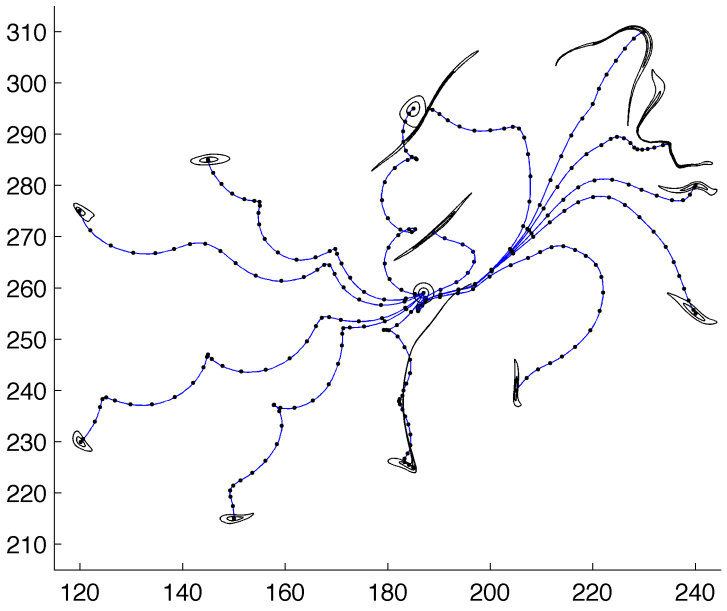


Figure 2.19: Effect of flow and control on forward-computed (open-loop) minimum time trajectories (of Figures 2.12-2.13) for perturbations in the initial state (x_0, y_0). The trajectories are shown in blue, with black markers placed every three hours. The two circles of perturbed initial states and the resulting curves of final states are shown in black. The final states are highly sensitive to the initial state when there is stretching in the flow itself, e.g. along the five rightmost trajectories.

Chapter 3

Minimum Energy Feedback Control: a Backward-in-Time semi-Lagrangian Approach

3.1 Overview

This chapter presents 1D and 2D algorithms and solution examples for the following "minimum energy" problem. The Lagrangian in (1.2) is $\mathcal{L}(\mathbf{x}, \mathbf{u}, t) = W\mathbf{u} \cdot \mathbf{u}$, where the parameter $W > 0$ is the weighting of the energy cost relative to the end cost h. The final time t_f is fixed, i.e. the terminal hypersurface in (1.3) is $\mathbf{m}(\mathbf{x}, t) = t - t_f$. The admissible inputs are still $\Omega = \{\mathbf{u} \in \mathrm{R}^n, |\mathbf{u}| \leq s\}$ (and the dynamics are $\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}, t) + \mathbf{u}$), as in the case of minimum time. The connection between this problem and the minimum time problem to a fixed position emphasized in the previous chapter lies in a simple variant of that problem that was discussed only briefly: minimum time to one or more passive particle trajectories (e.g. those terminating at minima of the present end cost function).

By using a soft constraint instead of a hard constraint, and focusing only on the final state, the minimum energy control still drives vehicles to these "target" trajectories, but (a) not all the way to them (except in the limit of zero weight on the energy term) and (b) without any hurry (even in the limit of zero weight on the energy term). This makes the transient behavior less constrained and thus more interesting than in the minimum time case or for instance in the nonlinear regulator case.

Recall that the main result of chapter 2 was an algorithm (published in [24]) for obtaining globally optimal minimum time trajectories without solving the HJB equation, but that the HJB equation was solved on an unstructured grid in [23]. The minimum energy HJB equation cannot be avoided, and thus the problem at hand is to solve this HJB equation, obtaining optimal trajectories via the optimal cost-to-go function and associated minimum energy feedback control law. Fortunately the boundary conditions are simpler in the minimum energy case; in the minimum time case, the cost-to-go is specified (as zero) at the target set (and infinite at points not controllable to the target set), whereas in the new problem it is specified at the final time slice. This immediately suggests a simple backwards time-stepping approach, to solve the HJB equation directly.

The method is a semi-Lagrangian finite difference scheme, on a structured but adaptive grid. The basic idea is to approximate the solution of the HJB equation by propagating information along its characteristics, backward in time.

This makes the scheme semi-Lagrangian. However, characteristics—in this case optimal trajectories—generally merge, both backward in time, forming shocks, and forward in time, forming rarefactions. Thus one must take care to determine the correct direction and speed of the characteristic at every grid point, capturing shocks and rarefactions properly. In this sense the goal of the present semi-Lagrangian scheme is similar to that of an Eulerian Godunov upwinding scheme [21].

The present scheme computes the proper direction of the characteristic on-line, via direct solution of the associated constrained optimization problem. In summary, the global optimal trajectory problem is transformed into a point-wise optimization problem, solvable by a variety of methods. We restrict ourselves to a first-order in space and time version of the scheme. The idea is to use the exact solution of the point-wise optimization problem and focus on reducing interpolation error via adaptive grids. This is in contrast to the method of [13], which allows for higher-order discretizations but not adaptive grids.

The remainder of this chapter is outlined as follows. Section 3.2 gives the formal problem statement and the HJB equation (in 1D and 2D). Section 3.3 presents the algorithm in 1D, where it is slightly simpler, but conceptually the same as in 2D. Section 3.4 presents a few key 1D solution examples that capture not all but most of the phenomena observed in 2D. Section 3.5 presents the extension of the algorithm to 2D. Section 3.6 presents solutions in 2D. Instead of a general

Summary and Outlook section, the computational complexity and performance of the algorithm are discussed briefly in Section 3.6.1.

3.2 2D Minimum Energy Problem

We now restate the general optimal control problem of Section 1.1 for the specific case of fixed final time minimum energy control in 2D. The same notation is used for the 1D case, which will not be restated separately. Differences include the scalar variables and the specification of domain boundaries (since we will no longer be using an unstructured Lagrangian grid like in Chapter 2).

Given an initial vehicle state $(x_0, y_0) \in \mathrm{D} := [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ and an initial time $t_0 \in [0, t_f]$, the optimal trajectory problem is to minimize the cost functional

$$J(u, v; x_0, y_0, t_0) := \int_{t_0}^{t_f} W(u^2(t) + v^2(t))dt + h(x(t_f), y(t_f)), \tag{3.1}$$

with respect to the control input signal (u, v) subject to the input constraint

$$u^2(t) + v^2(t) \leq s^2$$

(for all $t \in [t_0, t_f]$), where W is the weighting of the energy cost relative to the end cost h, s is the (maximum) vehicle speed, and the state trajectory $(x, y) = (x(\cdot), y(\cdot))$—implicitly dependent on $(u, v) = (u(\cdot), v(\cdot))$—is the unique solution

of the state equations

$$\dot{x}(t) = f(x(t), y(t), t) + u(t);$$

$$\dot{y}(t) = g(x(t), y(t), t) + v(t)$$

with the initial conditions

$$x(t_0) = x_0;$$

$$y(t_0) = y_0.$$

Again, x, etc. may be used to mean either a trajectory $x(\cdot) : [t_0, t_f] \to \mathrm{D}$, as above, or an instantaneous state $x(t) \in \mathrm{D}$, as implied by the context.

In this case the Hamiltonian is

$$\mathcal{H}(x, y, u, v, p, q, t) = (f + u) \cdot p + (g + v) \cdot q + W(u^2 + v^2)$$

(where $(p, q) = \mathbf{p}$ is the costate vector) and thus the HJB equation (1.13) is

$$V_t = - \min_{(u,v)\in\Omega} \left\{ (f + u) \cdot V_x + (g + v) \cdot V_y + W(u^2 + v^2) \right\} \tag{3.2}$$

(where $\Omega = \{(u, v) \in \mathrm{R}^2, u^2 + v^2 \leq s^2\}$).

3.3 1D Algorithm

This section presents the semi-Lagrangian Godunov finite difference scheme for the solution of the minimum energy HJB equation, in 1D, where it is simpler to derive but conceptually the same as in 2D. First we derive the finite difference scheme, then the point-wise minimization, and finally the adaptive grid.

3.3.1 Semi-Lagrangian discretization of HJB equation

In 1D, the HJB equation (3.2) is

$$V_t = -\min_{u\in\Omega}\left\{(f+u)\cdot V_x + Wu^2\right\}$$

i.e.

$$\min_{u\in\Omega}\left\{V_t + (f+u)\cdot V_x + Wu^2\right\} = 0, \tag{3.3}$$

(where now $\Omega = [-s, s]$). The optimal control here is given analytically by $u = -\frac{V_x}{2W}$ if $|V_x| \leq 2Ws$, $u = -s$ if $V_x > 2Ws$, and $u = s$ if $V_x < -2Ws$, but this value cannot be used quite yet.

Suppose x_u is an optimal trajectory i.e. characteristic passing through a given point (x, t). Then (3.3) can be written as

$$\min_{u\in\Omega}\left\{\frac{d}{dt}V(x_u(t), t) + Wu^2\right\}.$$

A simple first-order discretization in time yields

$$\min_{u\in\Omega}\left\{\frac{V(x_u(t+\Delta t), t+\Delta t) - V(x_u(t), t)}{\Delta t} + Wu^2\right\} = 0.$$

Multiplying by Δt, substituting x for $x_u(t)$ and x' for $x_u(t+\Delta t)$, and rearranging yields

$$V(x,t) = \min_{u\in\Omega}\left\{V(x', t+\Delta t) + Wu^2\Delta t\right\},$$

$\Longleftrightarrow$

$$V(x,t) = \min_{x'\in B}\left\{V(x', t+\Delta t) + Wu^2\Delta t\right\},$$

where, by first-order Taylor expansion, $x' \approx x + (f+u)\Delta t$ and thus $u \approx \frac{x'-x}{\Delta t} - f$, and where $B := [x+(f-s)\Delta t, x+(f+s)\Delta t]$ is the closed interval of states reachable in the given time step. Partitioning B based on the 1D simplexes $\sigma \in G$ defining the solution grid at time $t + \Delta$ yields

$$V(x,t) = \min_{\sigma \in G} \min_{x' \in B \cap \sigma} \left\{ V(x', t+\Delta t) + W u^2 \Delta t \right\}.$$

All that remains is how to approximate $V(x', t+\Delta t)$, from a given simplex σ of the time $t+\Delta$ slice of the grid (on which V will have already been computed). We choose to use linear interpolation. This is done purely so that the minimum can be computed exactly (in 1D and 2D), as will be explained further in Section 3.3.2 (and, for the 2D case, in Section 3.5.2). The result is $V(x', t+\Delta t) = V_1 + V_x(x' - x_1)$, where x_1 and x_2 are the vertices of the grid element σ, V_1 and V_2 are the cost-to-go values there, and $V_x = \frac{V_2 - V_1}{x_2 - x_1}$. Thus the quantity being minimized is

$$V^\sigma(x') := V_1 + V_x(x' - x_1) + W u^2 \Delta t$$

and the formula is

$$V(x,t) = \min_{\sigma \in G} \min_{x' \in B \cap \sigma} V^\sigma(x'). \tag{3.4}$$

This formula is the core of the algorithm. It is evaluated once for each discrete time t from $t_f - \Delta t$ to 0 and each point x in (that time slice of) the grid. To be clear, each such evaluation involves looping through all (relevant) grid elements σ.

3.3.2 The exact point-wise minimum in 1D: quadratic objective with linear constraints

This section takes a closer look at the minimization of $V^\sigma(x')$ with respect to x' in (3.4), for a particular grid element $\sigma \in G$ of the grid at time slice $t + \Delta t$.

Solving $\frac{d}{dx'}V^\sigma(x') = V_x + 2Wu\Delta t\frac{du}{dx'} = V_x + 2Wu\Delta t\frac{1}{\Delta t} = V_x + 2Wu = 0$ yields an unconstrained minimum of $V^\sigma(x')$ at $u = -\frac{V_x}{2W}$ i.e. $x' = x + (f - \frac{V_x}{2W})\Delta t$. Assuming $B \cap \sigma \neq \emptyset$, it's easy to see that the desired constrained minimum is given by the point in $B \cap \sigma$ closest to the unconstrained minimum point—in particular, $x' = x'_{\min}$ if $x + (f - \frac{V_x}{2W})\Delta t < x'_{\min}$, $x' = x'_{\max}$ if $x + (f - \frac{V_x}{2W})\Delta t > x'_{\max}$, or $x' = x + (f - \frac{V_x}{2W})\Delta t$ if $x + (f - \frac{V_x}{2W})\Delta t \in [x'_{\min}, x'_{\max}]$, where $x'_{\min} := \min\{B \cap \sigma\} = \max\{x_1, x + (f - s)\Delta t\}$ and $x'_{\max} := \max\{B \cap \sigma\} = \min\{x_2, x + (f + s)\Delta t\}$. (Note that in the 2D case this will not be as straightforward; for one, the constraint $x \in B$ becomes nonlinear.) We don't show the explicit version of the formula (3.4) that results from substituting these values, since it would be a rather long piecewise function.

3.3.3 Adaptive grid

This section describes the adaptive 1D grid used with the semi-Lagrangian Godunov scheme described above throughout Section 3.4. It is simple, but proved highly useful even in 1D, and helps lay the groundwork for a straightforward 2D adaptive grid, which is a crucial part of the overall algorithm. The grid consists

primarily of a growing array of vertices, a fixed array of growing binary trees of "edge" elements (referred to above by σ), which point to one another and to the vertices. The only parameters (besides the time step Δt for instance, which was already mentioned) are

- n_x, the initial grid size and
- $\Delta V_{\max}$, the maximum allowed error.

If V (or whatever function is being computed on the adaptive grid) is actually discontinuous (for instance if the end cost h is discontinuous), we also enforce a maximum allowed "level" (of refinement) $L_{\max}$, but this is not the case in this dissertation, for which h is not only continuous but smooth. The case where h and thus V are discontinuous is also of interest, and has been tested successfully, but is outside the scope of this dissertation. This case includes when $|v(x,t)| > s$ at the boundary and V jumps to ∞.

The grid is initialized with n_x "root" edges and $n_x + 1$ initial vertices. Upon creation of any vertex, the value of the cost-to-go V is computed at once, using the formula (3.4) (and the point-wise minimum described above) or, if $t = t_f$, the end cost h. Moreover, upon the creation of any edge, an additional vertex is created at its center. (So, technically, there are actually $2n_x + 1$ initial vertices.) If the edge never gets split, this center vertex is not used for the example trajectories or for the visualization of the solution V. It is only used if the edge gets split, and for the purposes of the error estimate ΔV on which the entire adaptive grid

refinement algorithm is based. The error is simply $\Delta V = \left|V - \frac{V_1+V_2}{2}\right|$, where V_1 and V_2 are the costs-to-go at the endpoints. Note that the full hierarchical tree data structure is necessary for efficient interpolation of the adaptive grid, both in the point-wise optimization and in the simulation of the optimal feedback control law.

Finally, the grid is graded. That is, the sizes of neighboring edges are allowed to differ by no more than a factor of 2. This property of the grid is maintained using recursion: before splitting an edge, one splits any neighbor larger than that edge (i.e. whose level L is smaller).

3.3.4 Simulation of optimal feedback control

The second part of the algorithm is to obtain the optimal trajectories for specific example points x_0 (and times t_0), by direct simulation of the feedback control law. The most obvious way to define the feedback control law u (the argument of the minimum (3.4)) at a given point during trajectory simulation is to have recorded it, with V, on the grid, and then simply interpolate, but this introduces significant error, especially near the shocks, where u is discontinuous. Hence we choose to perform the minimization (3.4) not only during the computation of V, but again during the feedback simulation, to obtain u more accurately. This makes the feedback simulation take significantly longer (especially in the case of adaptive grids, where the number of partitions $B \cap e$ is very large in places), but

nowhere near as long as the computation of V, since the number of simulated trajectories desired is no where near the number of grid points.

An estimate of the error e at time slice i of a given closed-loop trajectory is given by $e_i := V_i - \Sigma_{j=i}^{n_t} W u_j^2 \Delta t$, where V_i is linearly interpolated from the adaptive grid.

3.4 1D Results

This section discusses the solutions for a few key example cases, all in 1D. Section 3.4.1 considers the case of spatially invariant but time-varying flows, demonstrating the invariance of the control along optimal trajectories and the basic role of shocks. Section 3.4.2 considers the case of a spatially varying but time-invariant flows, demonstrating the optimal control's non-greedy exploitation of spatial sensitivity in the so-called flow map, and the existence of certain "attractive" optimal trajectories associated with a steady state solution of the HJB equation in backward time. Section 3.4.3 considers the case of a spatially varying and time-varying flow, demonstrating the rather simple effect of the time-dependence. All the problem parameters, algorithm parameters, and performance statistics are given in Tables 3.1, 3.2, and 3.3, respectively.

Table 3.1: Problem parameters for 1D minimum energy examples

Example	3.4.1	3.4.2a	3.4.2b	3.4.3
$f(x,t)$	0	$\sin(\pi x)$	$\sin(\pi x)$	$(\sin(2\pi t)+0.5)\sin(\pi x)$
$h(x)$	$-\sin(\pi x)$	$-\sin(\pi x)$	$-\sin(\pi x)$	$-\sin(\pi x)$
W	1	1	10	1
s	0.5	1	1	1
$x_{\min}$	0	0	0	0
$x_{\max}$	3	3	3	3
t_f	1	2	2	2
x_0	$\{0, \frac{1}{10}, ..., 3\}$	$\{0, \frac{3}{100}, ..., 3\}$	$\{0, \frac{3}{100}, ..., 3\}$	$\{0, \frac{3}{100}, ..., 3\}$
t_0	$\{0, 0, ..., 0\}$	$\{0, 0, ..., 0\}$	$\{0, 0, ..., 0\}$	$\{0, 0, ..., 0\}$

Table 3.2: Algorithm parameters for 1D minimum energy examples

Example	3.4.1	3.4.2a	3.4.2b	3.4.3
$\Delta V_{\max}$	1e-7	1e-5	1e-5	1e-5
n_x	10	10	10	10
n_t	100	100	100	100

Table 3.3: Performance statistics from $t = 0$ slice of 1D minimum energy examples

Example	3.4.1	3.4.2a	3.4.2b	3.4.3
Approx. no. vertices	2600	750	870	580
Approx. max. trajectory error e	1.8e-7	3.0e-4	3.0e-4	3.2e-4
Max. observed level of refinement L	8	15	17	14

3.4.1 Spatially invariant flows

Consider the case where the flow is spatially uniform and thus divergence-free, i.e. $f(x,t) = f(t)$ and thus $f_x(x,t) = 0$. In this case the costate equation of (1.11) is $\dot{p} = -f_x p = 0$. This implies the costate p and control $u = \min\left\{\max\left\{-s, \frac{-p}{2W}\right\}, s\right\}$ are constant along optimal state-costate trajectories. The values are simply $p = p(t_f) = h_x(x(t_f))$ and $u = \min\left\{\max\left\{-s, \frac{-h_x}{2W}\right\}, s\right\}$, where h_x is the gradient of the end cost h. The resulting trajectories evolve backwards in time from the final time slice, where $V = h$ and $V_x = h_x$. Roughly speaking, if h has a local maximum, the characteristics collide with one another in backward time, and are terminated, forming the non-smooth cusps known as shocks in the surface $V(x,t)$. Note that whereas these globally optimal trajectories are terminated in backward time, the state-costate trajectories persist, and generally intersect one another throughout large regions of space-time. The shocks organize these regions of multiple locally optimal trajectories, directing the globally optimal trajectories to one side or the other. Also note that for spatially varying flows (like those of Sections 3.4.2 and 3.4.3) not all shocks are associated with local maxima of h.

Shocks appear in almost all the examples of this dissertation, and the invariance of the control along minimum energy trajectories is true for all spatially uniform flows, but for simplicity we now focus on a special case where the flow is

zero i.e.

$$f(x,t) = f(t) = 0$$

(and thus the trajectories are line segments) and the end cost h is a smooth function with two local minima separated by a local maximum. In particular, we use

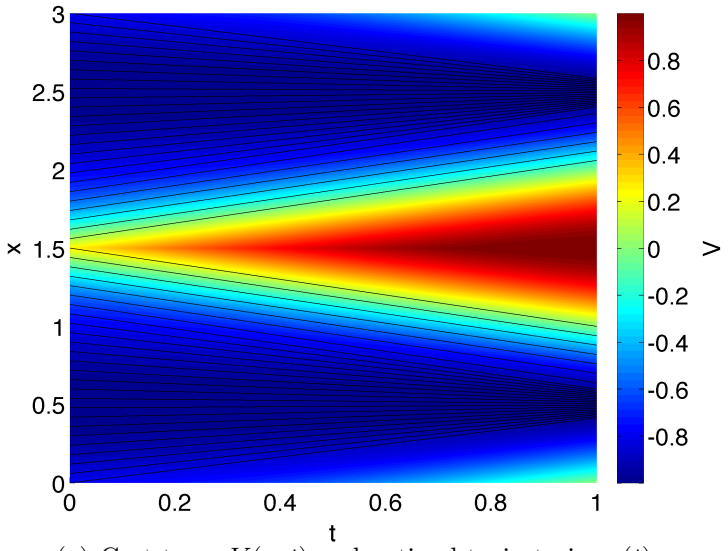
$$h(x) = -\sin(\pi x).$$

The points in the final time slice where the control transitions from $|u| < s$ to $|u| = s$ are obtained easily: $\left|\frac{-h_x}{2W}\right| = s \iff |h_x| = 2Ws \iff |-\pi\cos(\pi x)| = 2Ws \iff \cos(\pi x) = \pm\frac{2Ws}{\pi} = \pm\frac{2(1)(0.5)}{\pi} = \pm\frac{1}{\pi} \approx \pm 0.318 \iff \pi x \approx \frac{\pi}{2} \pm 0.324 + i\pi \iff x \approx 0.5 \pm 0.103 + i, i \in \mathbb{Z}$. Figure 3.1 shows the numerical results.

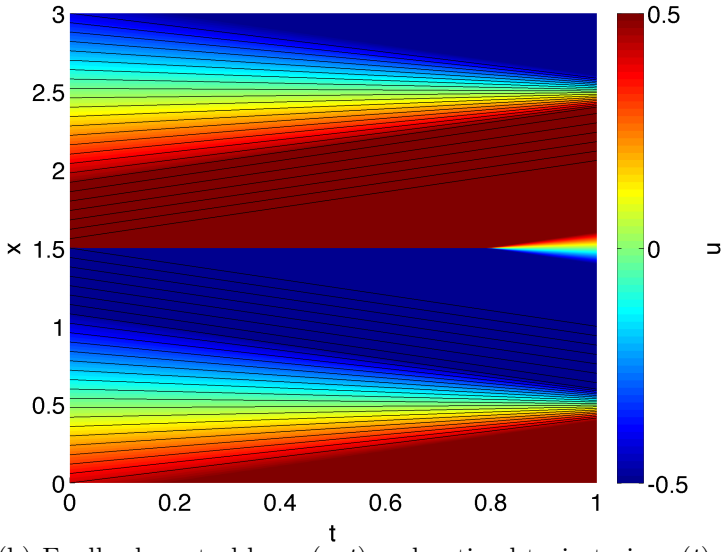
First of all, the result matches the predicted solution. The trajectories are indeed line segments, since the control is constant.The control varies continuously from $-s = -0.5$, for the several trajectories in the blue region of Figure 3.1(b), to $s = 0.5$, for those in the red, and as predicted the boundaries of the blue and red regions at $t = t_f$ appear to be ≈ 0.103 from $x = 0.5$, 1.5, and 2.5. The cost-to-go decreases steadily at a rate of $Wu^2 = u^2 \in [0, s^2] = [0, 0.25]$.

The shock lies along the line $x = 1.5$. It is most visible as a discontinuous jump from $u = -s = -0.5$ (blue) to $u = s = 0.5$ (red) in Figure 3.1(b). The shock ends (i.e. $|V_x(1.5,t)|$ falls below $2Ws$) at approximately $t = 0.8$. The required number of grid points required to get straight line trajectories was larger than expected,

and the adaptive grid, strangely, did not use a higher level of refinement along the shock, even when varying n_x. This didn't happen in any other results.



(a) Cost-to-go $V(x,t)$ and optimal trajectories $x(t)$



(b) Feedback control law $u(x,t)$ and optimal trajectories $x(t)$

Figure 3.1: Solution for the 1D zero flow case of Section 3.4.1. There is a shock (a cusp in V or a discontinuity in u) at $x = 1.5$ from $t = 0$ to ≈ 0.8. Each trajectory $x(t)$ descends V at a constant rate of $\frac{d}{dt}V(x(t),t) = -Wu^2$. This is true for any spatially-invariant flow $\mathbf{v}(\mathbf{x},t) = \mathbf{v}(t)$, since $\mathbf{u}$ depends on the gradient $V_\mathbf{x}$, and $\frac{d}{dt}V_\mathbf{x}(\mathbf{x}(t),t)$ is given by the costate equation $\dot{\mathbf{p}} = -\mathbf{v}_\mathbf{x}(\mathbf{x},t)\mathbf{p}$.

3.4.2 Spatially varying but time-invariant flows

The next two examples consider the flow

$$f(x,t) = f(x) = \sin(\pi x)$$

and the same end cost as before,

$$h(x) = -\sin(\pi x),$$

with two different values of W. Figures 3.2-3.9 show the numerical results for both cases.

The effect over time of the flow on the end cost is best observed in terms of what we will call the *pulled back end cost* function $H(x,t) := h(X(x,t,t_f))$ (where X is the "flow map" described in Chapter 1.3). This function is discussed in more detail in Chapter 4 but is essentially the cost-to-go if one were to turn off the control. It is shown for the two present examples in Figure 3.6. As confirmed by this figure, $H(x,t_f) = h(x)$. The end cost h is 0 at all four visible fixed points of the flow f but $x = 1$ and $x = 3$ are stable and $x = 0$ and $x = 2$ are unstable. The minima of h lie between fixed points and are more easily reached from below.

The question, besides which of the two minima to seek, is whether the ease of arriving from below is worth the effort of escaping the unstable fixed point initially. These questions are answered in terms of three shocks, which divide the trajectories into four distinct groups: two for each minimum of h. For example, in Figure 3.4(a), the control jumps from $-s$ (blue) to 0 (green) at $x(0) \approx 0.3$,

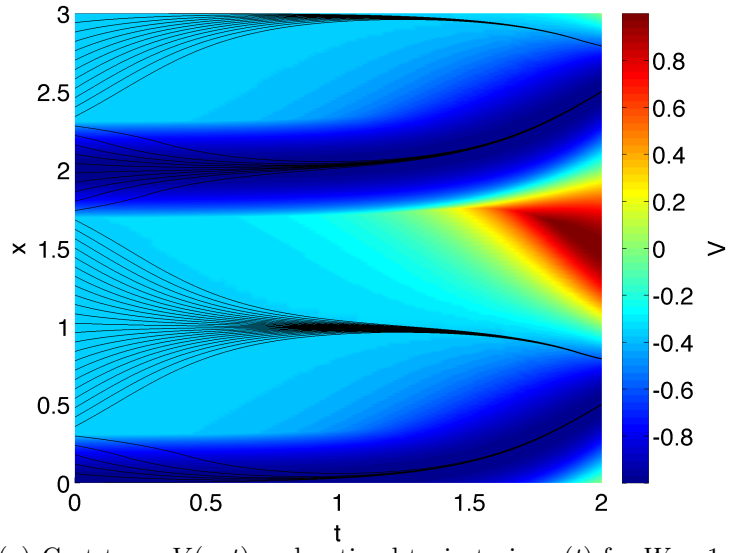
from 0 to s (red) at ≈ 1.7, and from $-s$ to 0 again at 2.3 (since obviously the flow is a sinusoid of period 2). For $x(0) \in [0.3, 1.7]$ (and $[2.3, 3.0]$) one knows it is impossible to get very close to the minimum of the end cost, and the optimal strategy is to wait and apply control $u < 0$ near the end. These trajectories (the majority) are almost indistinguishable in Figures 3.5(a), 3.3(a), and 3.7(a). For $x(0) \in [1.7, 2.3]$ (and $[0.0, 0.3]$) the optimal strategy is to use more control, concentrated near the beginning, and drift in from below to arrive precisely at the minimum of h. Basically, for the larger W, the regions where its worth it to arrive from below are smaller. Figure 3.9 shows very well how each adaptive grids focus on the three shocks at time $t = 0$.

This flow was chosen to highlight the non-greedy nature of the optimal control. This is best captured in Figure 3.7 by the evaluation along the trajectories of the pulled back end cost H. In both cases, several trajectories climb up and over the maximum of this function in order to eventually reach its minimum. In Figure 3.6(a), at time $t = 0$, the (red) maximum and (blue) minimum of H are both concentrated near the unstable fixed point at $x = 2$ and the trajectories that climb the maximum are below it and above the main shock. Naturally, this efficiency is much larger at the unstable fixed points $x = 0$ and $x = 2$, and the trajectories nearby are aware of it and indeed exploit it in order to make it all the way over said maximum. Secondly, it dies out exponentially over time, explaining the concentration of control effort at the beginning. In summary, the pulled back

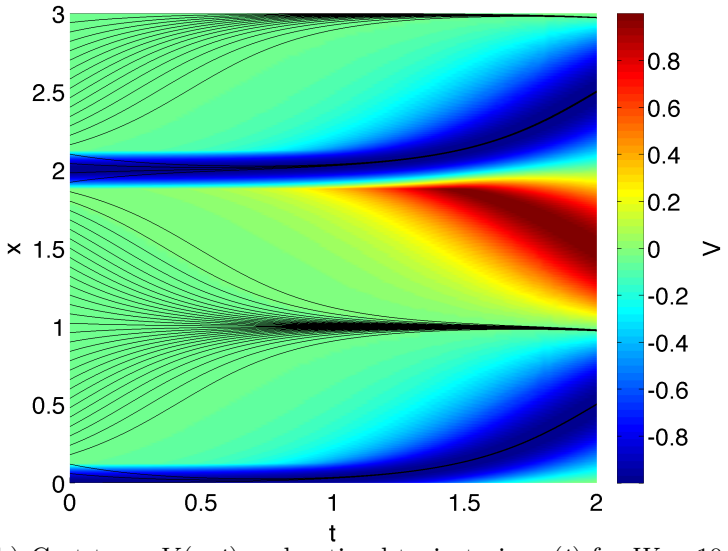
end cost and its gradient very effectively relate the optimal control not to the flow itself but to the effect over time of the flow on a particular end cost.

The main shock dies out at time $t \approx t_f - 0.2$, as in the example of Section 3.4.1. The two secondary shocks die out significantly before that. Practically speaking this means that if one were to release another fleet of vehicles at, say, time $t = 1.5$, they would organize into just two groups instead of four. Moreover, their final states would be much more spread out within the two basins of the end cost h. It's worth mentioning that the reason for simulating the feedback control beginning solely at time $t = 0$, and for choosing such a large final time t_f, is to focus on the special "attractive" trajectories around which other trajectories organize themselves given enough time, and to sufficiently capture the associated plateau-shaped steady state of the solution $V(x, t)$ in backwards time.

Finally, note that the for the larger W the hard bound s is irrelevant, since it turns out to exceed the maximum observed value the optimal control law.

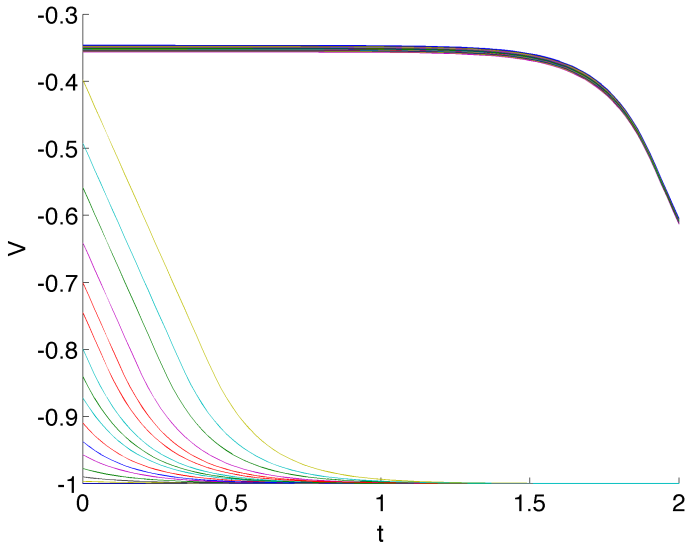


(a) Cost-to-go $V(x,t)$ and optimal trajectories $x(t)$ for $W = 1$

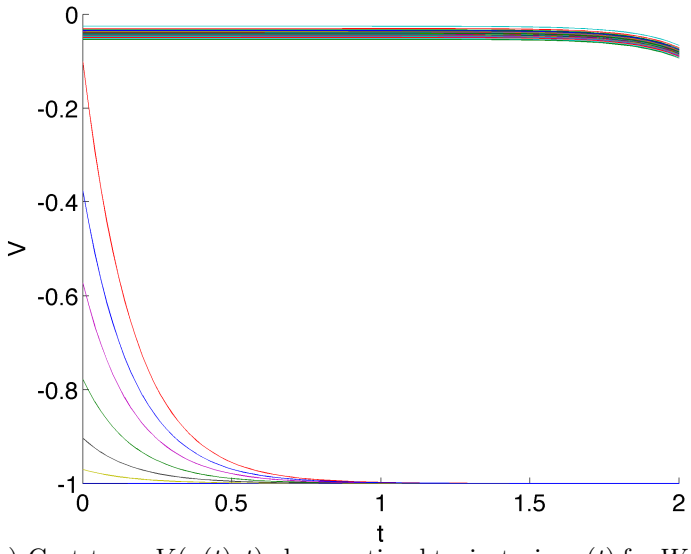


(b) Cost-to-go $V(x,t)$ and optimal trajectories $x(t)$ for $W = 10$

Figure 3.2: Cost-to-go $V(x,t)$ and optimal trajectories $x(t)$ for (a) $W = 1$ and (b) $W = 10$ in the 1D time-invariant case of Section 3.4.2. Three shocks (cusps in V) divide the trajectories into four groups. For $W = 1$ (a) the main shock is at $x \approx 0.3$ and terminates at $t \approx 1.8$, arising from the minima at $x = 0.5$ and 2.5 of the end cost $h(x) = V(x, t_f)$. The secondary shocks at $x \approx 1.7$ and 2.3 terminate at $t \approx 1.1$ and arise from the two opposing strategies for each of the two minima of h: expend more energy (earlier) and terminate right at the minimum, or expend less energy (later) and pay a larger end cost. For $W = 10$ (b) the shocks are closer together and terminate earlier, and of course $V(x,t)$ is greater for all (x,t).

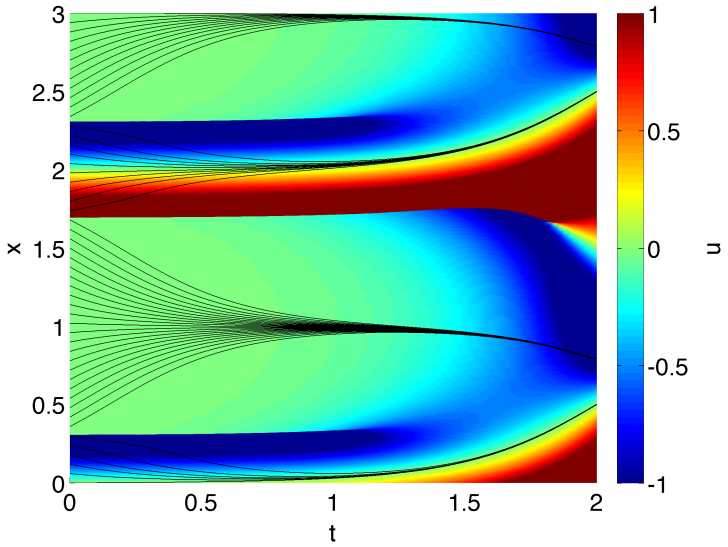


(a) Cost-to-go $V(x(t), t)$ along optimal trajectories $x(t)$ for $W = 1$

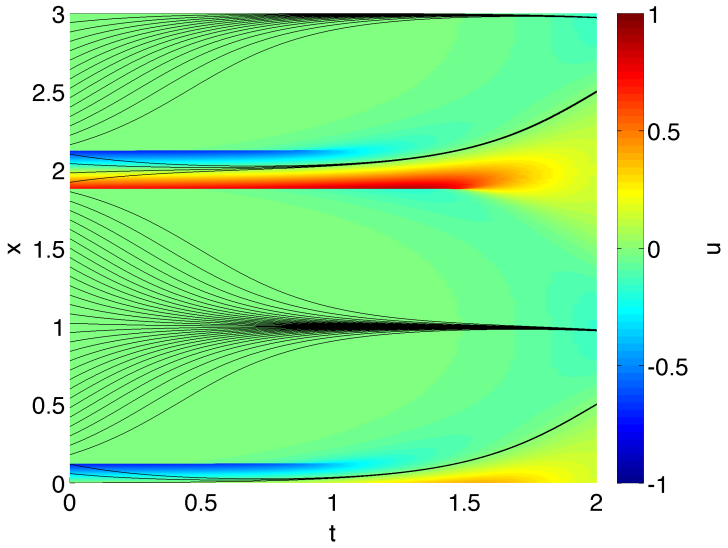


(b) Cost-to-go $V(x(t), t)$ along optimal trajectories $x(t)$ for $W = 10$

Figure 3.3: Cost-to-go $V(x(t), t)$ along optimal trajectories $x(t)$ for (a) $W = 1$ and (b) $W = 10$ in the 1D time-invariant case of Section 3.4.2. The topmost curves here correspond to the trajectories $x(t)$ initialized on the plateaus of V (light blue or green) in Figure 3.2 (the majority). For $W = 10$ more of the trajectories are in this group and V is greater everywhere.

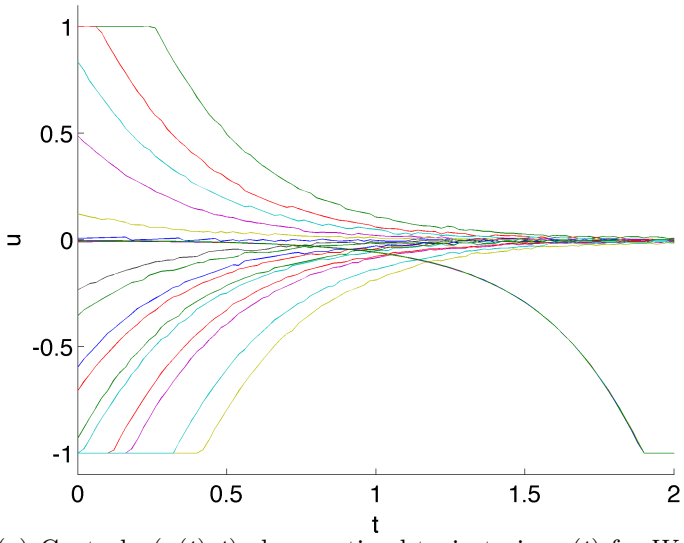


(a) Feedback control law $u(x,t)$ and optimal trajectories $x(t)$ for $W = 1$

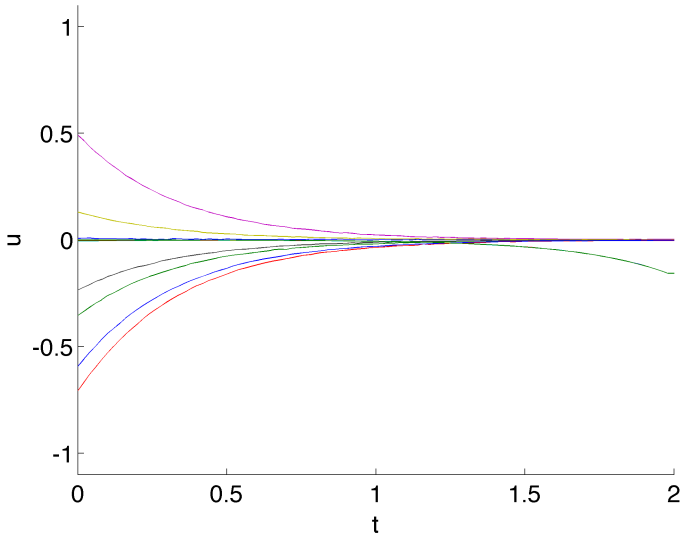


(b) Feedback control law $u(x,t)$ and optimal trajectories $x(t)$ for $W = 10$

Figure 3.4: Feedback control law $u(x,t)$ and optimal trajectories $x(t)$ for (a) $W = 1$ and (b) $W = 10$ in the 1D time-invariant case of Section 3.4.2. The three shocks—cusps in the cost-to-go $V(x,t)$—are more visible here as discontinuities in u. The control u is saturated in much of the domain for $W = 1$ (a) but not for $W = 10$ (b).

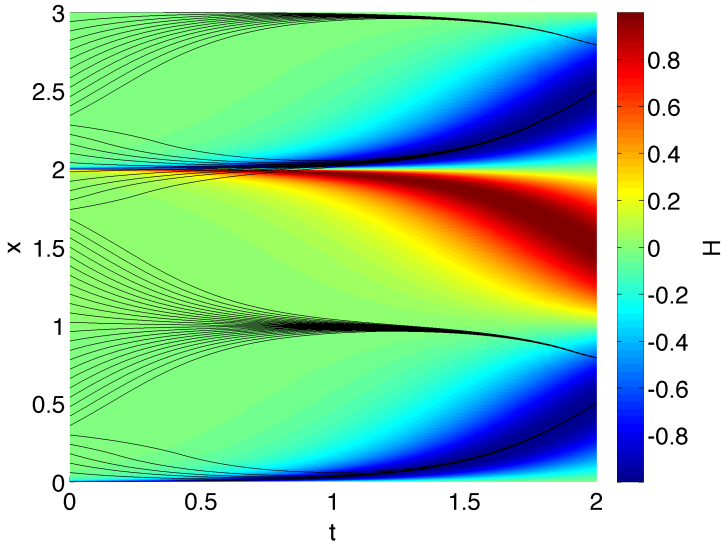


(a) Control $u(x(t), t)$ along optimal trajectories $x(t)$ for $W = 1$

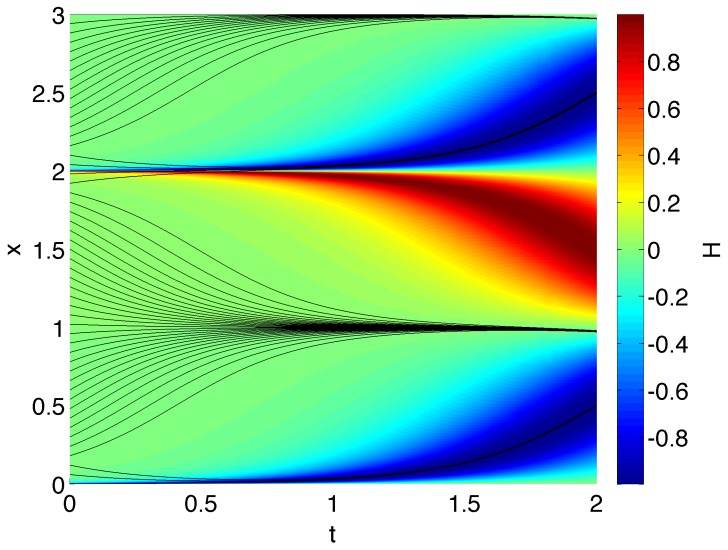


(b) Control $u(x(t), t)$ along optimal trajectories $x(t)$ for $W = 10$

Figure 3.5: Control $u(x(t), t)$ along optimal trajectories $x(t)$ for (a) $W = 1$ and (b) $W = 10$ in the 1D time-invariant case of Section 3.4.2. For $W = 1$ (a) the trajectories terminating at $x \approx 0.7$ and 2.7 in Figure 3.4(a) (the majority) have a burst of energy and $u = -s = -1$ at $t = t_f = 2$. Conversely u decreases over time for the handful of trajectories within reach of $x(t_f) = 0.5$ or 2.5. For $W = 10$ (b) the trend is the same, but the higher cost of energy means $|u|$ is "softly" constrained well within the hard bound of $s = 1$. This reveals the true benefit of an early transition from a stable fixed point to an unstable fixed point in that the initial value of u is as much as four times as large as the final value of u.

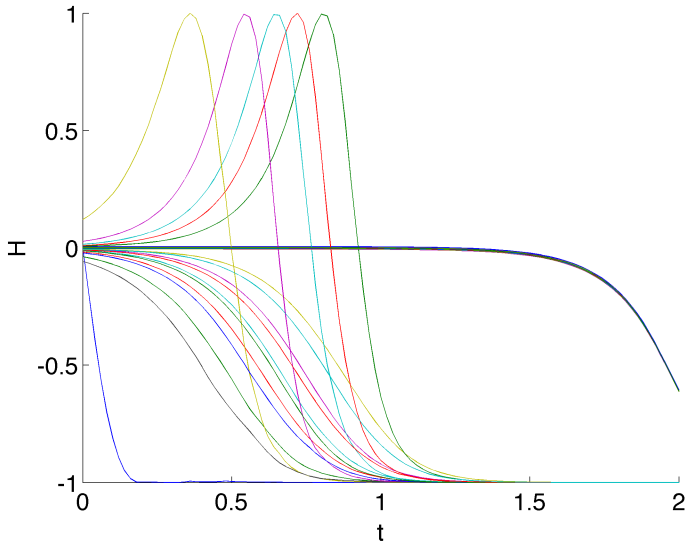


(a) Pulled back end cost $H(x,t)$ and optimal trajectories $x(t)$ for $W = 1$

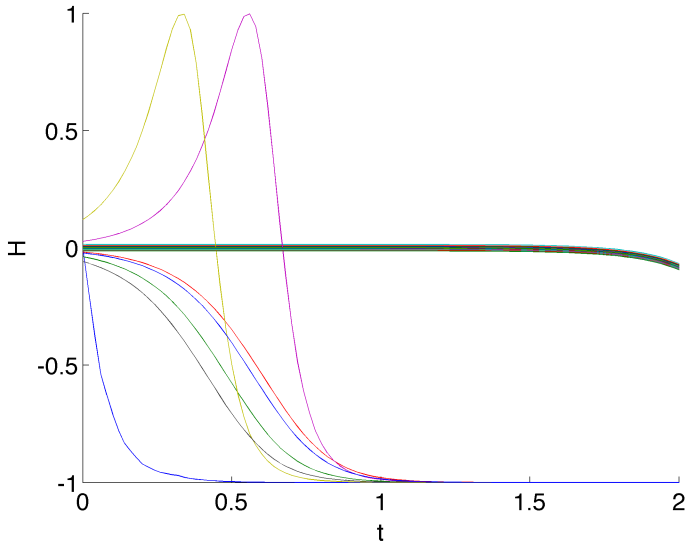


(b) Pulled back end cost $H(x,t)$ and optimal trajectories $x(t)$ for $W = 10$

Figure 3.6: Pulled back end cost $H(x,t)$ and optimal trajectories for (a) $W = 1$ and (b) $W = 10$ in the 1D time-invariant case of Section 3.4.2. The optimal control exploits the sensitivity of H near the unstable fixed point at $x = 2$, and ascends its slope locally to ultimately descend it globally. For $W = 10$ (b) fewer trajectories can afford to do this.

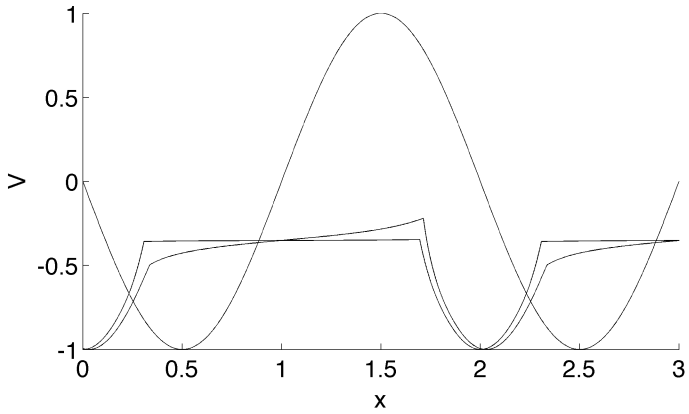


(a) Pulled back end cost $H(x(t), t)$ along optimal trajectories $x(t)$ for $W = 1$

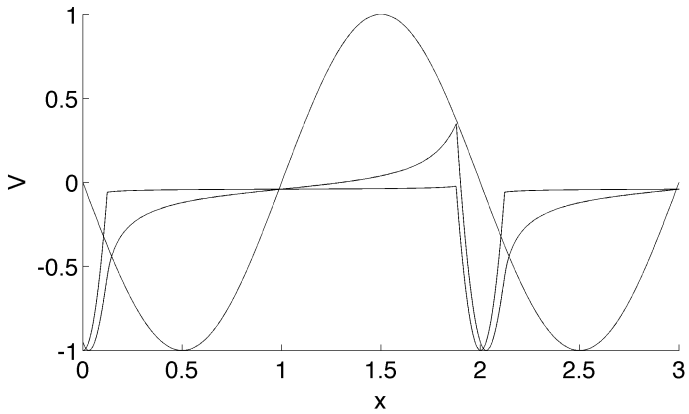


(b) Pulled back end cost $H(x(t), t)$ along optimal trajectories $x(t)$ for $W = 10$

Figure 3.7: Pulled back end cost $H(x(t), t)$ along optimal trajectories $x(t)$ for (a) $W = 1$ and (b) $W = 10$ in the 1D time-invariant case of Section 3.4.2. For $W = 1$ (a) the five trajectories initialized just below $x = 2$ in Figure 3.6(a) ascend from H near 0 to $H = 1$ in order to ultimately reach $H \approx -1$. This non-monotonic gradient descent (enabled by the sensitivity of H) highlights the non-greedy nature of optimal control. For $W = 10$ just two trajectories exhibit this behavior.

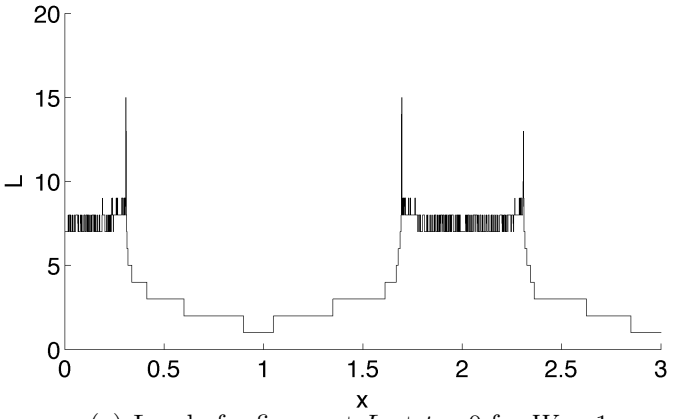


(a) Cost-to-go $V(x,t)$ at $t = 2$ (top), 1 (middle) and 0 (bottom) for $W = 1$

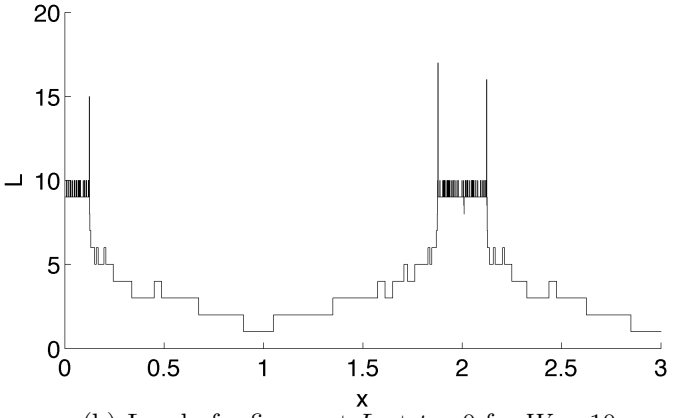


(b) Cost-to-go $V(x,t)$ at $t = 2$ (top), 1 (middle) and 0 (bottom) for $W = 10$

Figure 3.8: Time slices of cost-to-go $V(x,t)$ for (a) $W = 1$ and (b) $W = 10$ in 1D time-invariant case of Section 3.4.2. Computation proceeds backward in time, from $t = t_f = 2$, where $V(x,t) = h(x)$ (the sinusoidal end cost), to $t = 0$, where $V(x,t)$ appears close to a steady state—two plateaus and two valleys. The shocks (cusps) are more visible here than in the color plot of Figure 3.2. For $W = 10$ (b) the shocks are closer together and V is greater.



(a) Level of refinement L at $t = 0$ for $W = 1$



(b) Level of refinement L at $t = 0$ for $W = 10$

Figure 3.9: Level of refinement L of the adaptive grid at $t = 0$ for (a) $W = 1$ and (b) $W = 10$ in the 1D time-invariant case of Section 3.4.2. The refinement is based on linearly interpolated error estimates for the cost-to-go slice $V(x, 0)$ of Figure 3.8. The resulting grid resolution ranges from $\Delta x = 0.15$ ($L = 1$) to $\Delta x \approx 2.3 \times 10^{-6}$ ($L = 17$) and peaks at the shocks (the cusps of V). No explicit bound on L is needed, as long as V is continuous.

3.4.3 Spatially varying and time-varying flows

The final example considers the flow

$$f(x,t) = \sin(\pi x)(\sin(\pi t) + 0.5),$$

i.e. a time-varying version of the flow from the previous example, and the same end cost

$$h(x) = -\sin(\pi x).$$

Figures 3.10-3.17 show the numerical results.

Here, the flow of the previous examples is multiplied by a number that varies between -0.5 and 1.5. As such, the net effect is similar to that of the time-invariant case, but not quite as strong, since it is temporarily counteracted while the multiplier is negative. In particular, the region where the pulled back end cost is positive (red) in Figure 3.14 gets narrower in backward time but not without temporarily widening at times.

In addition, $V(x,0)$ is generally less here (in Figure 3.16) than in the time-invariant case (i.e. in Figure 3.8(a)), largely because the flow is zero at the final time, i.e. $f(x,t_f) = 0$; the trajectories drifting in to the minima of h from below need not get as close to the unstable fixed point initially, and those above, who saved their energy for the end, are fighting much less current. In fact, it is not clear whether the secondary shocks from the time-invariant case are actually shocks here—whether the jumps in control values are truly discontinuities; the difference

between the two strategies is fairly small. On the other hand, the main shock is basically the same as in the time-invariant case, but with oscillations.

Overall, the time-varying case is not fundamentally different from the time-invariant case. The key solution features are still the shocks and the non-greedy exploitation of sensitivity in the flow. Note however the lack of a (non-periodic) steady state for $V(x,t)$ in backward time.

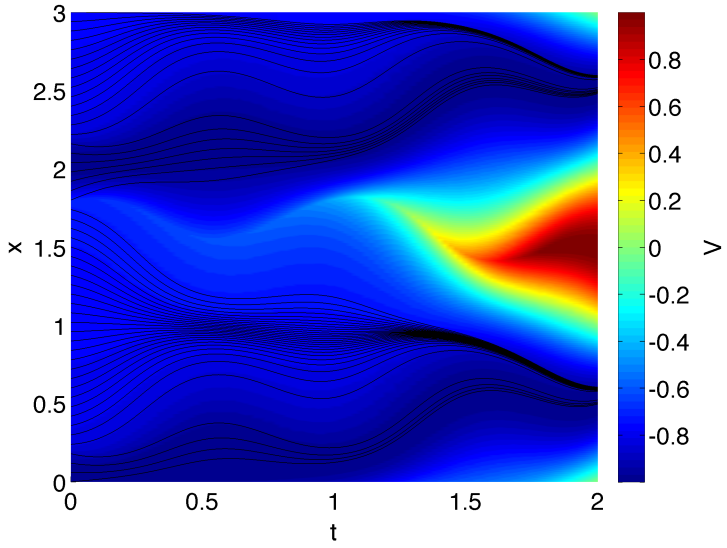


Figure 3.10: Cost-to-go $V(x,t)$ and optimal trajectories for example of Section 3.4.3. Compared to Figure 3.2(a), the main shock oscillates in time and the two secondary shocks is are less distinct; in other words there is a smoother transition between the "$h=-1$" strategy and the lower-energy strategy.

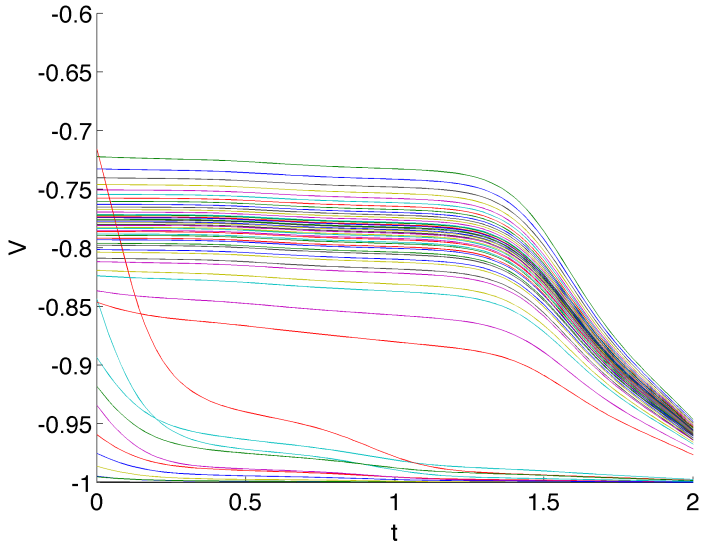


Figure 3.11: Cost-to-go $V(x(t),t)$ along optimal trajectories $x(t)$ for the example of Section 3.4.3. The two main strategies are to expend more energy (earlier) and arrive precisely at a minimum of $h=-1$ of expend less energy (later) and pay a higher end cost h, as in Figure 3.3(a), but less extreme, and time-varying.

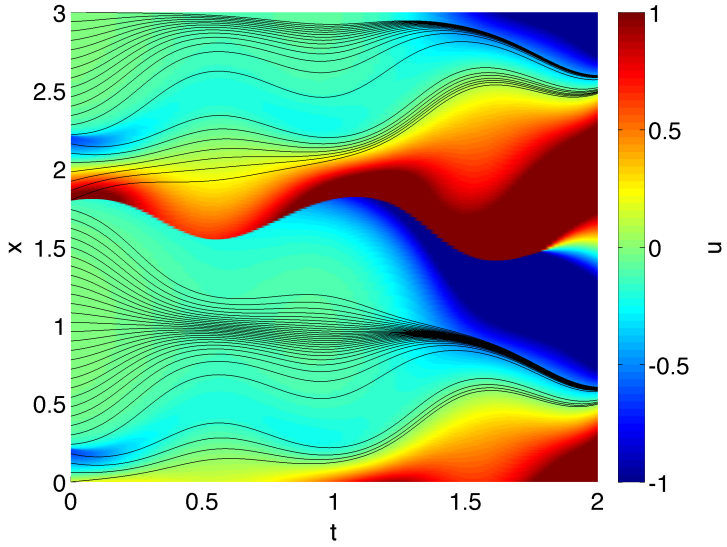


Figure 3.12: Optimal feedback control law $u(x,t)$ and optimal trajectories for example of Section 3.4.3. The main shock is where u jumps discontinuously from green or blue to red. It terminates at $t \approx 1.8$, just like in Figure 3.4(a).

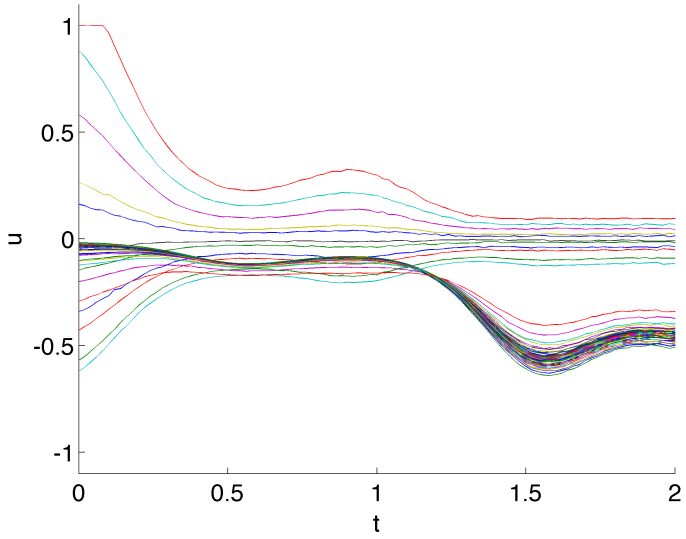


Figure 3.13: Optimal control signals u for the example of Section 3.4.3. The "top" group of signals expend more energy (earlier) and the "bottom" group of signals expend less energy (later), as already described in several instances. The hard input constraint of $s = 1$ only affects the "topmost" signal here.

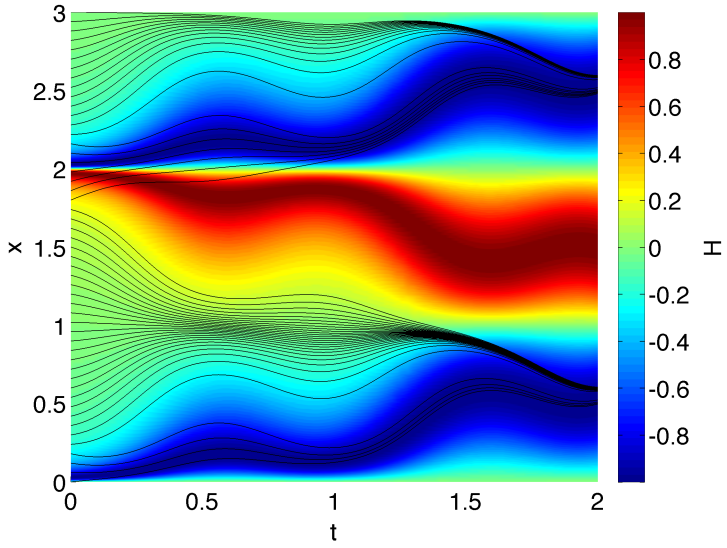


Figure 3.14: Pulled back end cost $H(x,t)$ and optimal trajectories for example of Section 3.4.3. The attraction of the red peak and blue valley to $x = 2$ in reverse time is much less extreme than in Figure 3.6(a) (due to the time-varying reversal of the flow), but several trajectories still cross over the red peak.

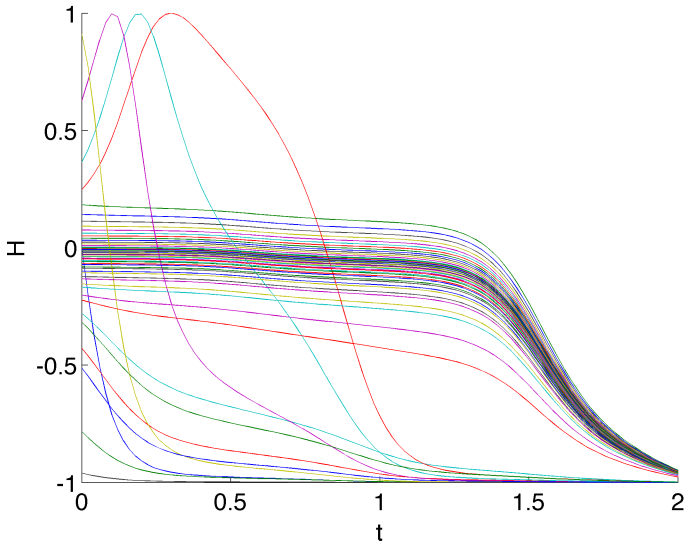


Figure 3.15: Pulled back end cost $H(x(t),t)$ evaluated along optimal trajectories $x(t)$ (shown in Figure 3.14) for the example of Section 3.4.3. The crossing of the peak of h by three trajectories again highlights the non-greedy nature of optimal control, as in the time-invariant results of Figure 3.7.

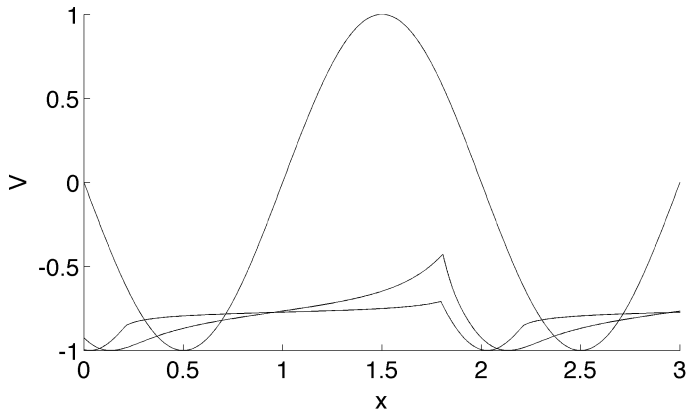


Figure 3.16: Backward computed cost-to-go $V(x,t)$ for the example of Section 3.4.3 at times $t = t_f = 2$ (where $V = h$), $t = \frac{t_f}{2} = 1$ (moderately flattened), and $t = 0$ (most flattened). Compare to the time-invariant result of Figure 3.8(a).

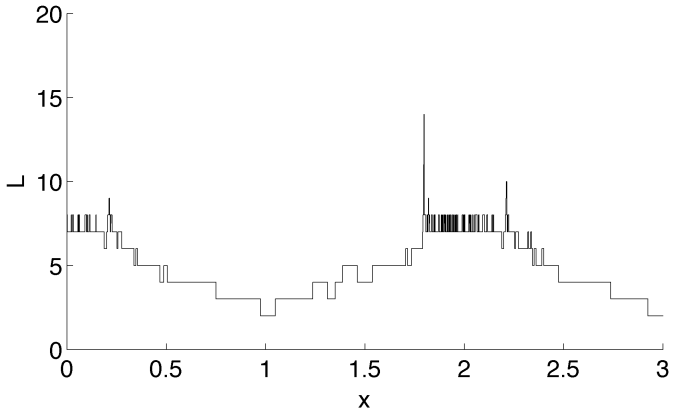


Figure 3.17: Level of refinement L for adaptive grid defining the cost-to-go slice $V(x,0)$ (shown in Figure 3.16) for the example of Section 3.4.3. Compare to the time-invariant result of Figure 3.9(a).

3.5 2D Algorithm

This section extends the semi-Lagrangian algorithm of Section 3.3 from 1D to 2D. First is the finite difference scheme, then the point-wise minimization, and finally the adaptive grid. The main differences are due to the fact that in 2D the primary grid elements are triangles instead of edges.

3.5.1 Semi-Lagrangian discretization of HJB equation

The 2D version of (3.4) is

$$V(x,y,t) = \min_{\sigma \in G} \min_{(x',y') \in B \cap \sigma} V^\sigma(x',y') \cdot \Delta t, \tag{3.5}$$

where now the grid elements $\sigma \in G$ are triangles, $(x',y') = (x,y)+((f,g)+(u,v))\cdot \Delta t$, B is the closed ball of radius $s\Delta t$ centered at $(x,y)+(f,g)\Delta t$, and

$$V^\sigma(x',y') := V_1^\sigma + V_x^\sigma(x'-x_1^\sigma) + V_y^\sigma(y'-y_1^\sigma) + W(u^2+v^2)\Delta t.$$

Here the gradient $(V_x^\sigma, V_y^\sigma) = \nabla V^\sigma$ is defined from the triangle vertex values x_1^σ, x_2^σ, x_3^σ, V_1^σ, V_2^σ, and V_3^σ by the solution of the linear system

$$\begin{bmatrix} x_2^\sigma - x_1^\sigma & y_2^\sigma - y_1^\sigma \\ x_3^\sigma - x_1^\sigma & y_3^\sigma - y_1^\sigma \end{bmatrix} \begin{bmatrix} V_x^\sigma \\ V_y^\sigma \end{bmatrix} = \begin{bmatrix} V_2^\sigma - V_1^\sigma \\ V_3^\sigma - V_1^\sigma \end{bmatrix}$$

3.5.2 The exact point-wise minimum in 2D: quadratic objective with *nonlinear* constraints

Substituting $u = \frac{x'-x}{\Delta t} - f$ and $v = \frac{y'-y}{\Delta t} - g$ into (3.5) ultimately yields

$$V^\sigma(x', y') = \frac{W}{\Delta t} \left|(x', y') - (x^*, y^*)\right|^2 + C^\sigma, \tag{3.6}$$

where

$$(x^*, y^*) := (x, y) + \left((f, g) - \frac{\nabla V^\sigma}{2W}\right) \Delta t$$

gives the minimum

$$C^\sigma := V_1^\sigma + V_x^\sigma (x^* - x_1^\sigma) + V_y^\sigma (y^* - y_1^\sigma) + W \left| -\frac{\nabla V^\sigma}{2W} \right|^2 \Delta t$$

in the absence of the constraint $(x', y') \in B \cap \sigma$. Note that the unconstrained optimal control here matches the form $-\frac{\nabla V}{2W}$ mentioned previously.

Equation (3.6) shows that the quadratic objective function $V^\sigma(x', y')$ has circular level sets. As such, the constrained minimum

$$\min_{(x', y') \in B \cap \sigma} V^\sigma(x', y')$$

is given simply the point $(x', y') \in B \cap \sigma$ nearest to (x^*, y^*) in the Euclidean norm, regardless of the value of the constant C^σ. Recall that in 1D the "projection" of x^* onto the boundary of $B \cap \sigma = [x'_{\min}, x'_{\max}]$ is trivial, since there are just two critical points ($x'_{\min}$ and $x'_{\max}$) and three cases for x^*. In 2D things are more complicated; the closed ball B and triangle σ yield one nonlinear inequality constraint and three linear inequality constraints respectively.

Our algorithm for constraining/projecting (x^*, y^*) "into" $B \cap \sigma$ works as follows:

1. Consider whether $(x^*, y^*) \in B \cap \sigma$ already.

2. Consider the individual projections of (x^*, y^*) onto the circular boundary ∂B and onto each of the three lines defining the triangle σ.

3. Consider the three vertices of the triangle σ and the up to six intersections between the lines defining σ and the circle.

4. Consider whether $B \cap \sigma$ is empty.

Examples of cases 1-3 are shown in Figure 3.18. The order here was chosen to work as efficiently as possible. In a preliminary implementation, we attempted to determine the specific case of (x^*, y^*) more explicitly, using the well-known Karush-Kuhn-Tucker (KKT) necessary conditions. The KKT conditions consider many special sub-cases of the above cases. We found it algorithmically simpler if not computationally more efficient to naively compare the value of the quadratic objective for the above cases as needed. Care was taken to ensure that the obtained solution is correct and thus consistent with the KKT conditions.

3.5.3 Adaptive grid

The extension of the adaptive grid of Section 3.3.3 from 1D to 2D is not trivial, but not difficult either. The primary grid elements are right (45-45-90-degree)

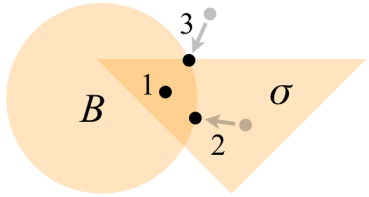


Figure 3.18: Schematics of the unconstrained optimum (x^*, y^*) being constrained to $B \cap \sigma$ for examples of the cases 1-3 described in Section 3.5.2. In case 1 none of the four inequality constraints is binding. In case 2 the one nonlinear constraint is binding. In case 3 both the nonlinear constraint and one of the linear constraints are binding. Each triangular element σ of the adaptive grid at $t + \Delta t$ reachable from the given grid point (x, y, t) yields a constrained optimization problem with a different point (x^*, y^*).

triangles, stored—like the edges—in a fixed array of binary tree data structures. Triangles point to vertices and edges, which are specified as either hypotenuses or legs.

The only new parameter is n_y. Instead of just n_x root edges, the initial grid now involves n_x by n_y rectangular root "cells"; these cells aren't defined explicitly, but are used in searching for triangles. In addition there are $2n_xn_y$ root triangles (2 per cell), $3n_xn_y + n_x + n_y$ root edges, and $(n_x + 1)(n_y + 1)$ initial vertices (not including those at edge centers).

Refinement proceeds as in 1D, with a queue of refinable leaf edges. The main complication is that after splitting an edge one must split the triangles to either side, and connect child edges to child triangles, paying close attention to their orientations relative to one another. Also, we maintain the convention the *length* of a grid element is inversely proportional to 2^L, where L is the level of refinement. Specifically, the *area* of a triangular grid element is given by

$\frac{1}{2}\left(\frac{x_{\max}-x_{\min}}{n_x}\right)\left(\frac{y_{\max}-y_{\min}}{n_y}\right)\cdot 2^{-2L}$. Hence each triangle bisection increments L by 0.5.

Maintaining the gradedness of the grid works differently than in 1D, but is still remarkably simple. Before splitting a given edge, one simply splits the hypotenuses of any neighboring triangles (not identical to the given edge). In other words, one never splits a triangle leg, and thus only splits the right triangles into more right triangles.

3.6 2D Results

This section presents and discusses the solutions for two examples in 2D. Section 3.6.1 considers the case of spatially varying but time-invariant "gyre" flow, for which the end cost function h is also a stream function. Section 3.6.2 considers a periodically time-varying version of the same flow, with the same end cost. All the problem parameters, algorithm parameters, and performance statistics are given in Tables 3.4, 3.5, and 3.6, respectively.

3.6.1 Time-invariant three gyre flow

This example and that of Section 3.6.2 both consider the end cost

$$h(x,y) = -\sin(\pi x)\sin(\pi y),$$

Table 3.4: Problem parameters for 2D minimum energy examples

Example	3.6.1	3.6.2
$f(x,y,t)$	$0.1\pi \sin(\pi x)\cos(\pi y)$	$0.1\pi \sin(\pi c(x,t))\cos(\pi y)$
$g(x,y,t)$	$-0.1\pi \cos(\pi x)\sin(\pi y)$	$-0.1\pi c_x(x,t)\cos(\pi c(x,t))\sin(\pi y)$
$h(x,y)$	$-\sin(\pi x)\sin(\pi y)$	$-\sin(\pi x)\sin(\pi y)$
W	10	10
s	1	1
$[x_{\min}, x_{\max}]$	$[0,3]$	$[0,3]$
$[y_{\min}, y_{\max}]$	$[0,1]$	$[0,1]$
t_f	10	10
(x_0, y_0, t_0)	30 by 10 grid w/ t_0=0	30 by 10 grid w/ t_0=0

Table 3.5: Algorithm parameters for 2D minimum energy examples

Example	3.6.1	3.6.2
$\Delta V_{\max}$	5e-4	5e-4
n_x	30	30
n_y	10	10
n_t	100	100

Table 3.6: Performance statistics from $t = 0$ slice of 1D minimum energy examples

Example	3.6.1	3.6.2
Max. observed level of refinement L	8.0	9.0
Approx. no. vertices	18,800	47,900
Approx. max. trajectory error e	0.020	0.021

visible in Figure 3.20(c) and this example considers a time-invariant three-gyre flow having h as a stream function, namely

$$f(x, y, t) = f(x, y) = -h_y(x, y) = 0.1\pi \sin(\pi x) \cos(\pi y);$$

$$g(x, y, t) = g(x, y) = h_x(x, y) = -0.1\pi \cos(\pi x) \sin(\pi y),$$

which is shown in terms of a snapshot of the time-varying three gyre flow of Section 3.6.2 in Figure 3.28(b). Figures 3.19-3.27 show the results.

The implicit "target" states—the minima of h—coincide with two of the three elliptic fixed points of the flow. This is in contrast to the 1D example of Section 3.4.2, where, recall, the targets lie in the strongest regions of the flow. Moreover, the end cost function h being a stream function means that the so-called pulled back end cost is simply the end cost h itself, that is $H(x, y, t) = h(x, y)$. As a result, the direction of the optimal control near these implicit target states is quite obvious: toward the target, and thus transverse to the streamlines of the flow. The availability of this mechanism in a Hamiltonian flow is used in the [66] to obtain conditions for controllability. As in Section 3.4.2, the main question addressed by the optimal control is which of the two targets to pursue and is answered in terms of a shock.

For our purposes recall a shock is a cusp in the cost-to-go V, or a discontinuity in the gradient of V. Whereas for 1D flows (time-varying or not) it forms a 1D curve in space-time and thus a 0d point in each time slice, for 2D flows it forms a 2D surface in space-time and thus a 1D curve in each time slice. In this particular

example the main shock (and the only one that is present at time $t = 0$) forms an S-shaped curve in the center gyre that is quite obviously related to the clockwise rotation there. The cusp in V itself, e.g. in Figure 3.19(a), though captured quite sharply by the adaptive grid, is hard to see. The shock is easier to see as a discontinuity in the control—essentially the negative gradient of V (e.g. in Figure 3.21(a))—or as a very sharp ridge in the level of grid refinement (e.g. in Figure 3.23(a)). Note that the level of refinement ranges from $L = 1.5$ to 8.0.

Figures 3.19-3.24 also show, in white, the two resulting groups of simulated optimal trajectories (with control vectors). Note from Figures 3.21-3.22 that $W = 10$ is large enough that the control magnitude is nowhere near the hard bound $s = 1$.

Also visible in the time slices $t = 6$ and $t = 8$, though it doesn't affect the simulated trajectories, is a short-lived secondary S-shaped shock intersecting the main shock at the very center of the flow. Though not investigated very thoroughly, the two smaller areas between the main shock and the secondary shock are thought to be where the optimal controller simply knows it's impossible to escape the center gyre within the time limit, and thus doesn't try.

The performance of the present algorithm was rather disappointing. It resulted in as many as 20,000 grid points per time slice, but took around a day of computation on a standard laptop and still yielded significant inaccuracies. In particular, the same error estimate e that was driven to 3e-4 along the 1D trajec-

tories of Section 3.4.2 was as large as 2e-2 in the present 2D case and the simulated optimal control signals (shown in Figure 3.26) were not as smooth as desired, as in the 1D examples. More largely, doubling the computation time yields only marginal improvement; in other words, there are still issues with computational complexity that are not well-understood.

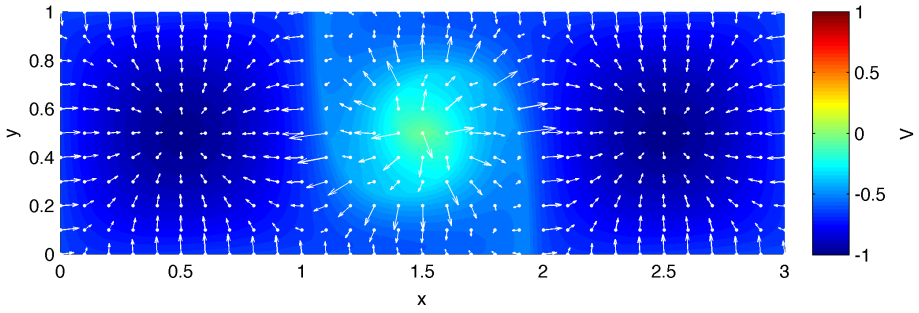
If the grid were uniform, the theoretical computational complexity in 2D could be summarized as follows. Let N be the number of grid points in each of the n_t time slices. The number of simplexes (triangles) considered for each of the $n_t N$ grid points is roughly proportional to the number of grid points in the subsequent time slice that are reachable within one time step, which in turn is roughly proportional to N and to the area of the reachable circle. The radius of the reachable circle is $s\Delta t \propto \frac{s}{n_t}$. Thus the area of the reachable circle is of the order $\left(\frac{s}{n_t}\right)^2$, the complexity of computing V at a given grid point is $O\left(\frac{s^2}{n_t^2}N\right)$, and the overall complexity is $O\left(\frac{s^2}{n_t^2}Nn_tN\right) = O\left(\frac{s^2}{n_t}N^2\right)$. The N^2 term is very bad. This is in contrast to the $O(n_tN)$ complexity of simpler schemes for simpler PDEs.

It would appear that one could alleviate this by simply increasing n_t, for instance choosing $n_t \propto N$. Then the overall complexity would appear to be $O(s^2N)$. In practice increasing n_t did not seem to work. The issue was not thoroughly investigated, but in addition to the $O(n_tN)$ memory complexity, is thought to be due to the fact that if N is the average number of grid points per time slice in an adaptive grid, then the grid points and triangles are concentrated

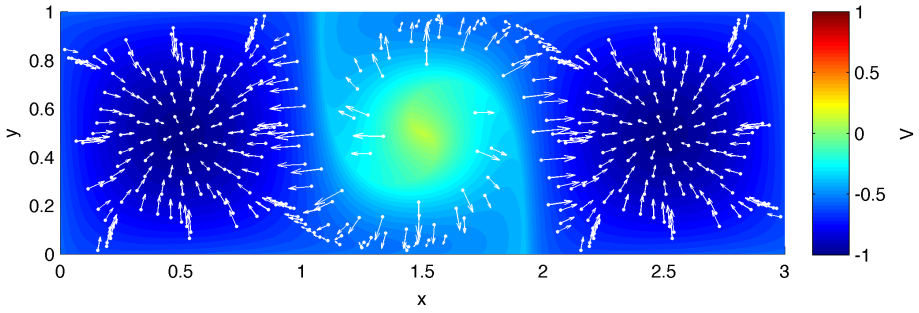
in a very small area, and thus the number of reachable simplexes for the average grid point is much larger than $O\left(\frac{s^2}{n_t^2}N\right)$. Though some improvement may be possible, this is thought to be a fundamental limitation of the algorithm that was somewhat overlooked. It has prompted us to look into more Lagrangian methods that are expected to be closer $O(N \log N)$ instead of $O(N^2)$; this is the topic of Chapter 5.

Despite all this, the adaptive grid resolved the shocks very well, if not too well. Moreover, the overall feedback control framework appears to be quite robust. The control trajectories, though not as smooth as desired, do generate state trajectories that are basically satisfactory, that behave correctly near the shock, and that capture the basic characteristics of minimum energy control. If the error were large enough, V might actually increase along trajectories, but that is not the case here, as shown in Figure 3.25. Finally, recall that one of the signs of an effective optimal control (if indeed there is much to gain from the optimal control for this particular example problem) is when the value of the so-called pulled back end cost function evaluated along trajectories increases temporarily in order to decrease further in the end. This phenomenon is observed in the present example, but only just barely (see Figure 3.27); especially compared to the analogous 1D example. This is largely due to fact, mentioned above, that the pulled back end cost function is equivalent to the end cost function, which is a stream function of

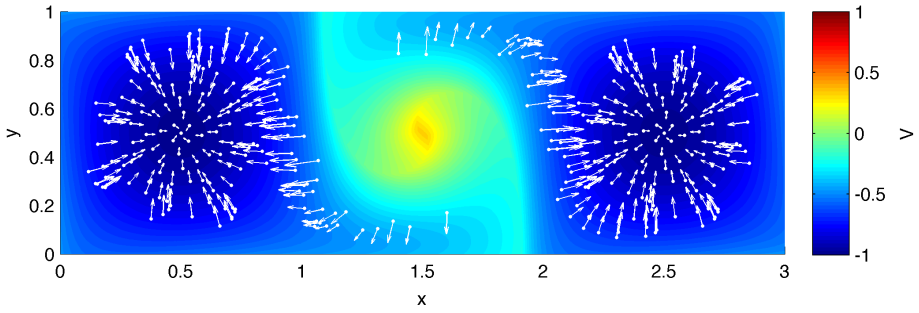
the flow. The next example, which considers a time-varying version of the same flow field, yields the more interesting behavior.



(a) $t = 0$.

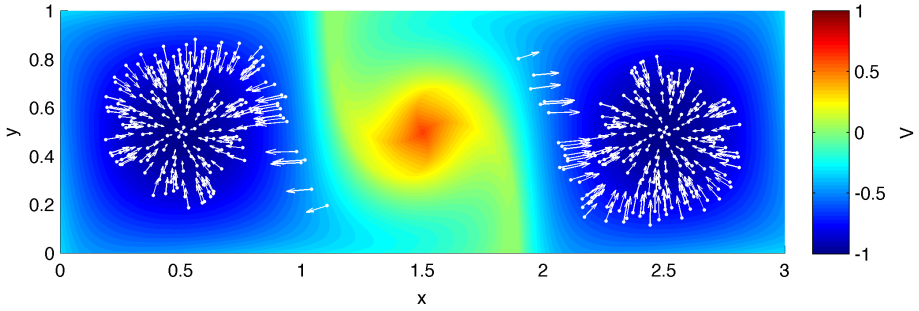


(b) $t = 2$.

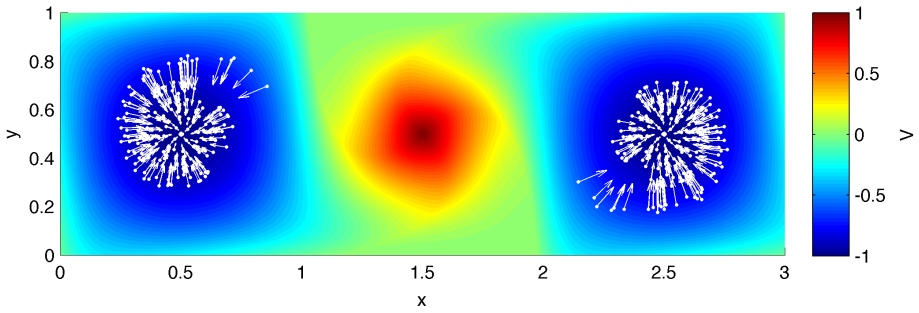


(c) $t = 4$.

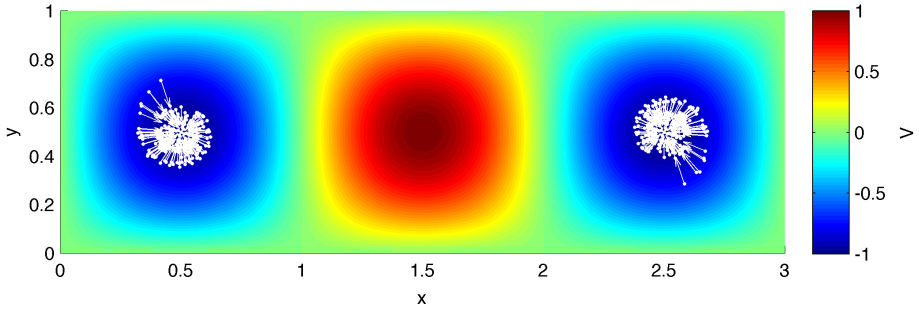
Figure 3.19: Initial time slices of the cost-to-go $V(x, y, t)$ (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.1. Though hard to see, a shock (a cusp in V) divides the trajectories as they exit the clockwise rotating center gyre. The main purpose of the adaptive triangular grid is to capture shocks like this.



(a) $t = 6$.

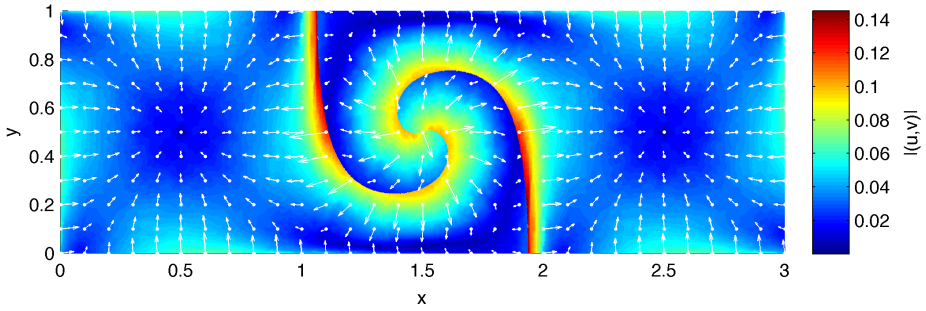


(b) $t = 8$.

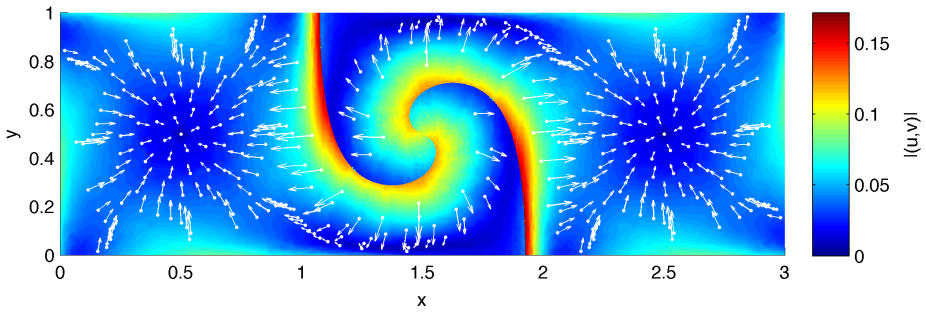


(c) $t = 10$.

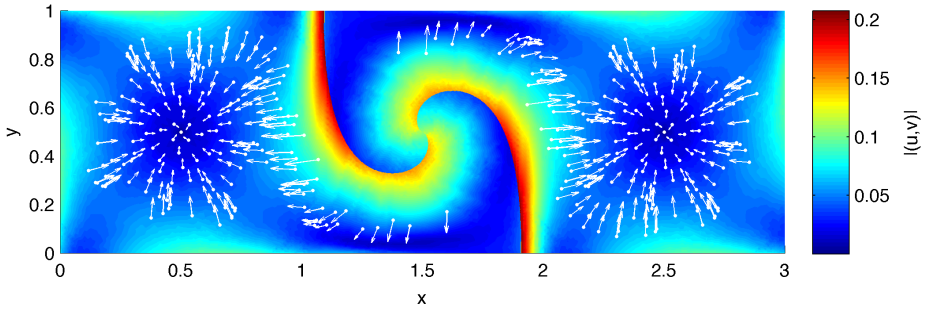
Figure 3.20: Final time slices of the cost-to-go $V(x, y, t)$ (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.1. The shock dies out and the trajectories essentially cross streamlines to spiral into the two counter-clockwise rotating gyres containing the minima of the end cost $V(x, y, 10) = h(x, y)$ (a stream function of the time-invariant flow (f, g)).



(a) $t = 0$.

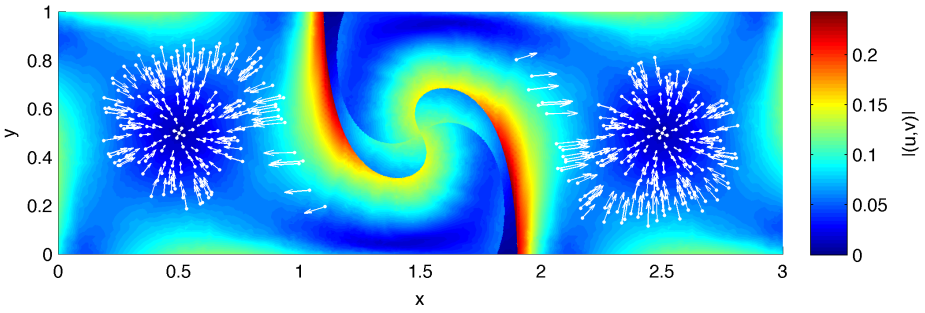


(b) $t = 2$.

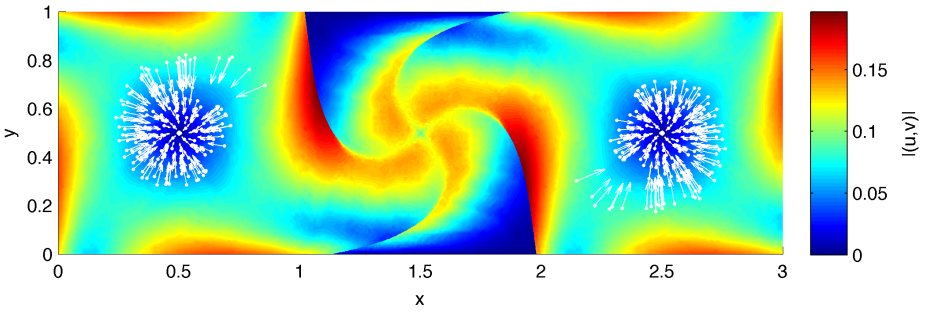


(c) $t = 4$.

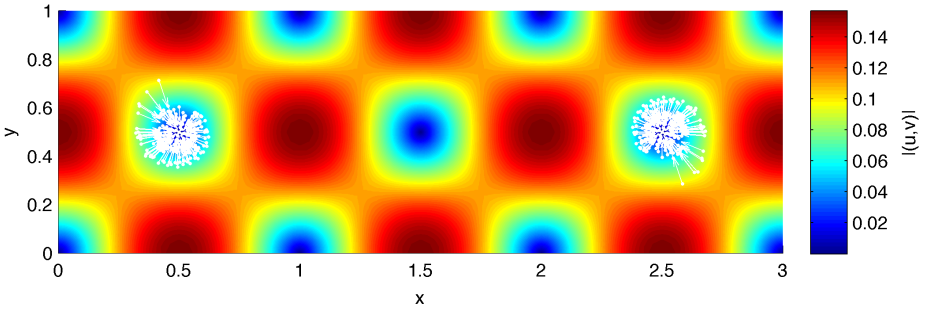
Figure 3.21: Initial time slices of the size of the optimal feedback control law $(u, v)(x, y, t)$ (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.1. The shock (a cusp in the cost-to-go V seen in Figure 3.19) is more visible here, as a discontinuity in (u, v)). Note that (u, v) is nowhere saturated and thus its size is proportional to the size of the gradient of V.



(a) $t = 6$.

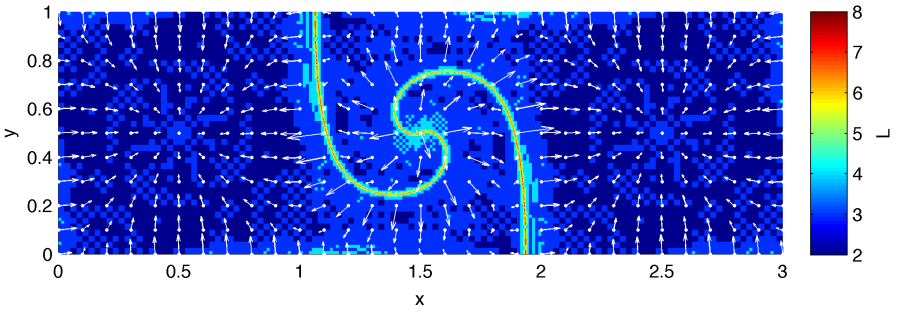


(b) $t = 8$.

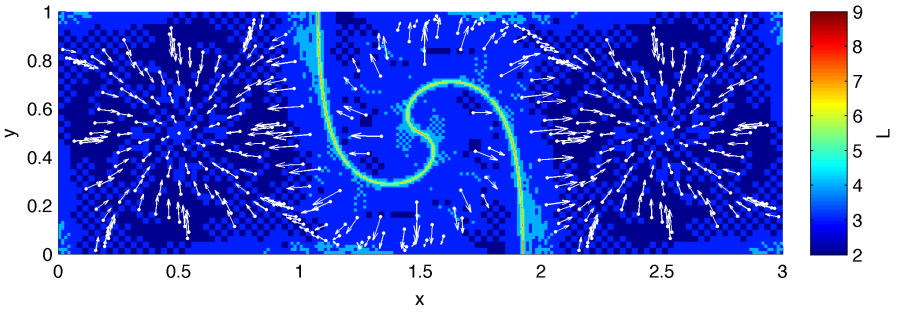


(c) $t = 10$.

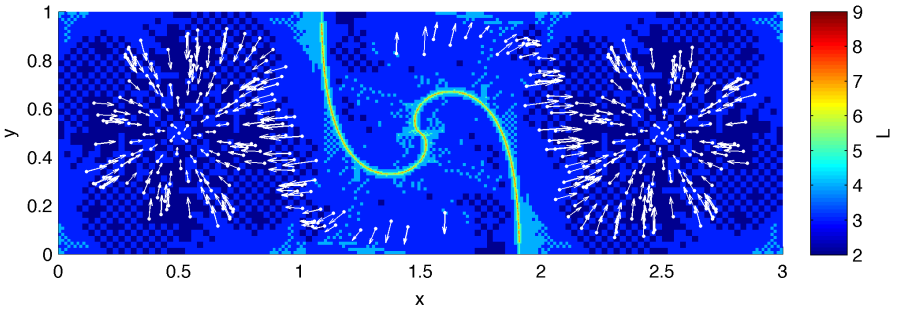
Figure 3.22: Final time slices of the size of the optimal feedback control law $(u, v)(x, y, t)$ (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.1. A secondary shock appears and intersects the main shock before they both die out. Trajectories initialized in the two small regions between the shocks would exhibit a different behavior because there would not be time to exit the center gyre.



(a) $t = 0$.

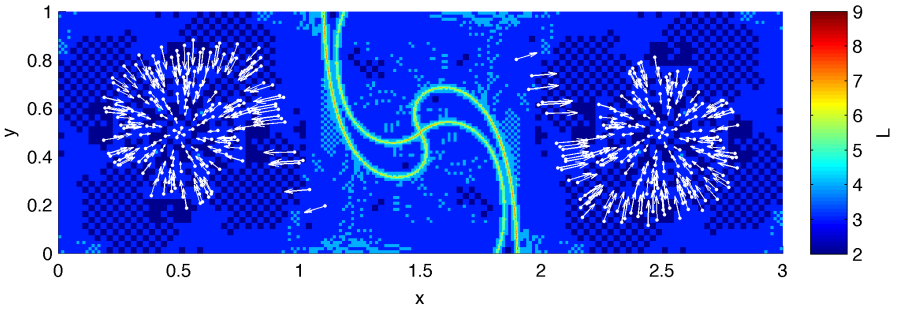


(b) $t = 2$.

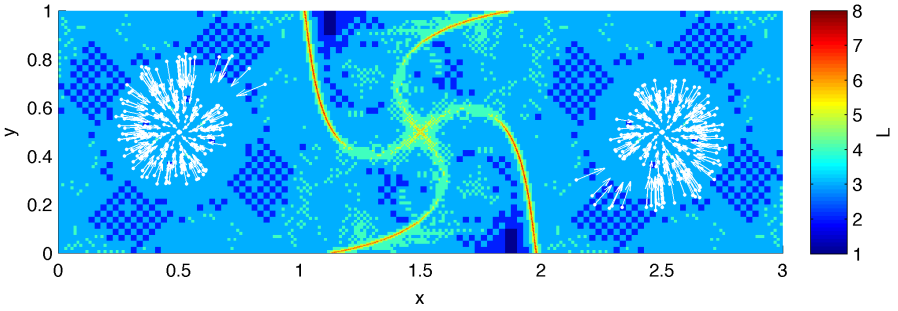


(c) $t = 4$.

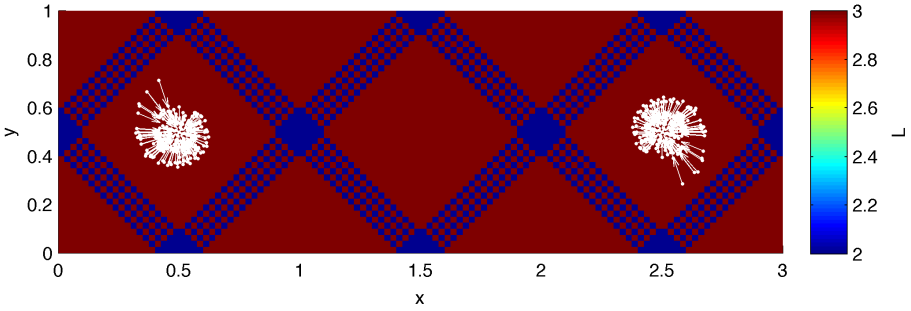
Figure 3.23: Initial time slices of the level of refinement L of the triangular grid elements (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.1. The shocks (cusps in the cost-to-go V in Figure 3.19 or discontinuities in the feedback control law (u, v) in Figure 3.21) are even more visible here as sharp ridges in L). An increase of one in L means a decrease in the area of a triangular grid element by a factor of four.



(a) $t = 6$.



(b) $t = 8$.



(c) $t = 10$.

Figure 3.24: Final time slices of the level of refinement L of the triangular grid elements (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.1. See also Figures 3.20 and 3.22.

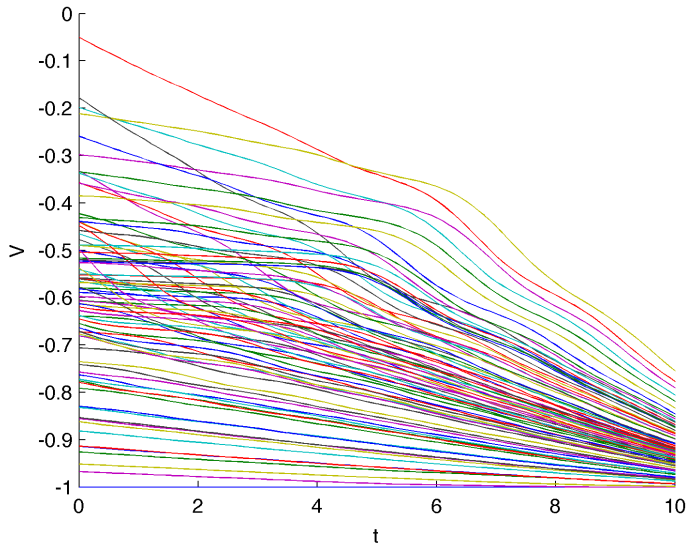


Figure 3.25: Cost-to-go V along optimal trajectories for the example of Section 3.6.1. V is monotonically decreasing, which indicates at least a basic level of accuracy.

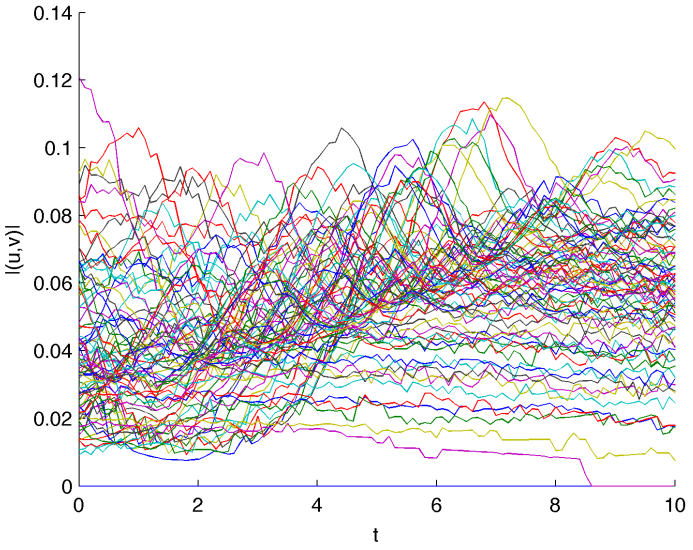


Figure 3.26: Size of control (u, v) along optimal trajectories for the example of Section 3.6.1. Signals are not as smooth as desired. Note that the hard constraint $|u^2 + v^2| \leq s = 1$ turns out to have no effect (other than to scale the complexity of the point-wise optimization procedure).

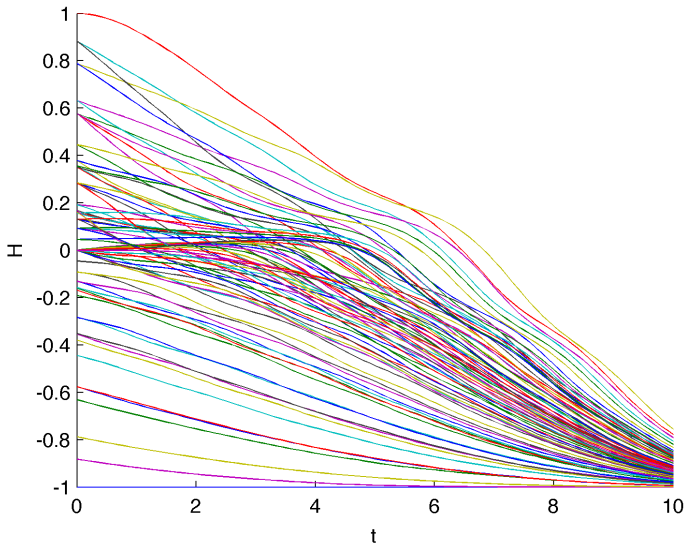


Figure 3.27: Pulled back end cost $H(x, y, t)$ along optimal trajectories for the example of Section 3.6.1. Transient growth of this value (evidence of the benefit of the optimal control over a simpler, more greedy gradient descent algorithm) is very limited. This is because the pulled back end cost function is actually a stream function of the time-invariant flow (f, g) and the optimal control is often just to cross streamlines.

3.6.2 Time-varying three gyre flow

This example considers the exact same problem as that of Section 3.6.1 except that it considers the following time-varying version of the previous flow field:

$$f(x, y, t) = 0.1\pi \sin(\pi c) \cos(\pi y);$$

$$g(x, y, t) = -0.1\pi \cos(\pi c) \sin(\pi y) c_x,$$

where $c = c(x, t) := \frac{2}{3}ax^2 + bx$ and thus $c_x = c_x(x, t) = \frac{4}{3}ax + b$, $a = a(t) = 0.25 \sin(\frac{\pi}{5}t)$, and $b = b(t) = 1 - 2a$. The results are shown in Figures 3.29-3.39. The analogous time-invariant results are shown in Figures 3.19-3.27.

The effect of the time-varying component of the flow is a sloshing back and forth of the center gyre and its two vertical separatrices, and thus chaotic mixing of passively advected particles. As such the optimal trajectories and the structure of the shocks in the solution of the HJB equation are much less intuitive and much more complicated. Snapshots of this flow are shown in Figure 3.28, but it is best visualized in terms of the pulled back end cost function H, shown along with the optimal trajectories in Figures 3.35-3.35. The shocks are easiest to see in Figures 3.33-3.34 as ridges in the grid refinement level.

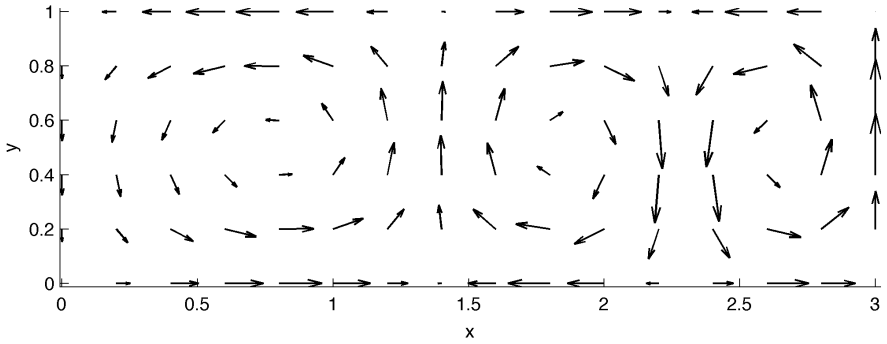
The basins of attraction of the two minima of h are only barely distinguishable just before time $t = 2$, when a vertical shock appears to intersect the x-axis at ≈ 2.2 (just left of another vertical shock at $x \approx 2.3$), and separates four of the simulated trajectories from several others nearby. Those several others go on to

travel all the way from the right gyre, along the bottom of the middle gyre, and to minimum of h at the center $(x, y) = (0.5, 0.5)$ of the left gyre. Over time, many shocks appear, disappear, merge, and otherwise change topology. But, as in the time-invariant three gyre flow of Section 3.6.1, the left and right basins are basically alone until, some time between $t = 4$ and $t = 6$, when, in the present example, two new areas form, which again correspond to where escaping the center gyre is impossible—one along the top, and one just below it in the shape of a very small triangle, which by time $t = 8$ stretches all the way to the bottom of the domain.

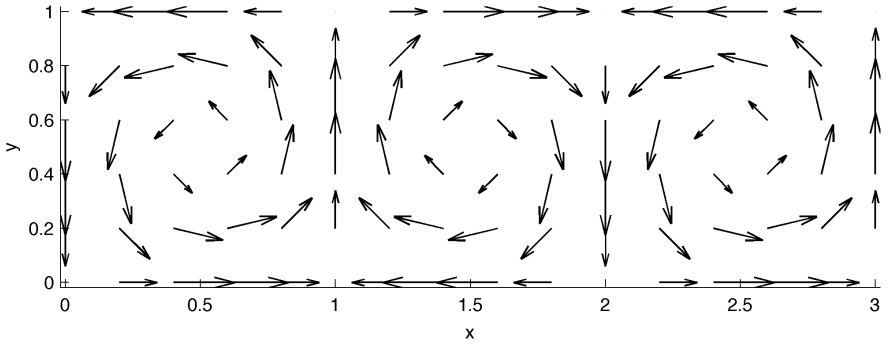
The most notable difference between this example and the time-invariant case is that one does witness significant transient growth in the pulled back end cost along several of the trajectories; from Figure 3.39, it appears that value grows from as low as ≈ 0.2 to as high as ≈ 0.9, in order to ultimately decrease to < -0.8. This is almost as extreme as the 1D results shown in Figures 3.7(a) and 3.7(b), for which the problem was specifically designed to highlight this transient growth (or "non-monotonic descent") phenomenon.

The pulled back end cost function $H(x, y, t)$ is shown globally in Figure 3.7(a). As will be described in more detail in Chapter 4.3 this function is computed on a different adaptive grid than V. The only difference was that the refinement criterion used was $2^{L/2}\Delta H > \Delta_{\max} = 0.001$ instead of $\Delta V > \Delta V_{\max} = 0.0005$. The refinement criteria for V could very well be computed this way as well, scaling

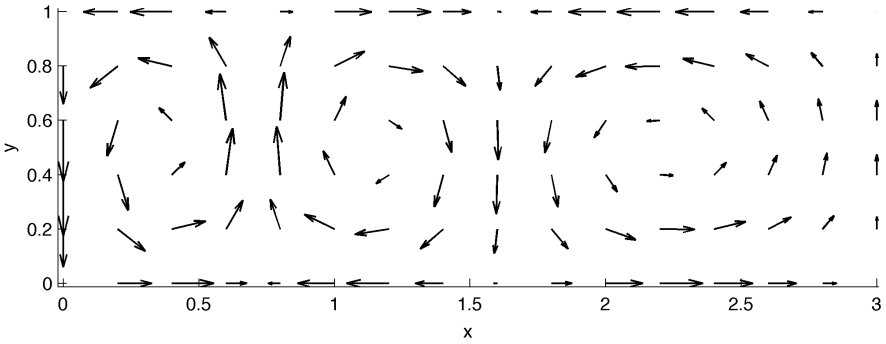
by length, or even area, but this would bias the computation away from the shocks. Computing the flow map (X, Y) on which H is based took on the order of half an hour, compared to about a day for V, both using C++ on a standard laptop. This yielded ≈ 66000 grid points at time 0 compared to $48,000$; however, the number of grid points required for a given tolerance grows exponentially in backward time, rather than peaking and decaying, as shown in Figure 3.43.



(a) $t = 2.5$.

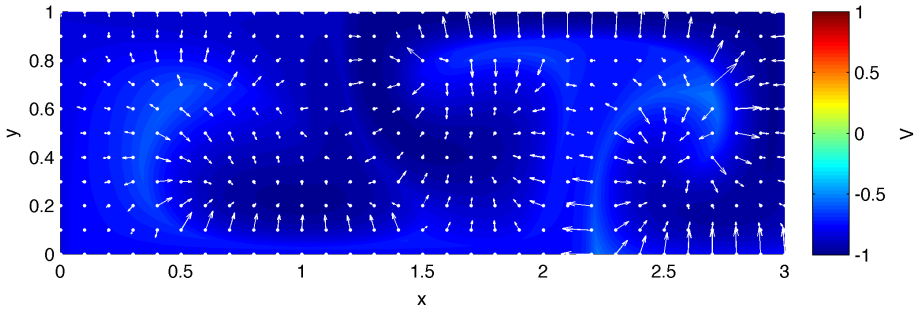


(b) $t = 5.0$.

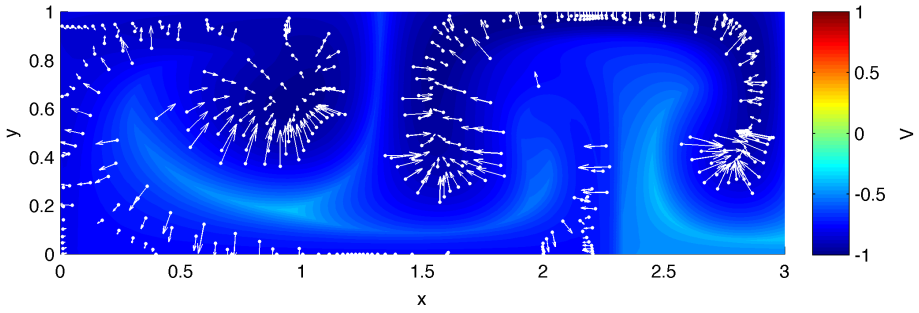


(c) $t = 7.5$.

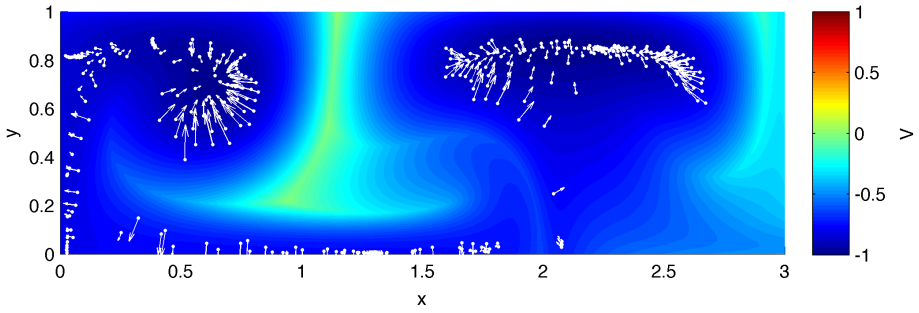
Figure 3.28: Snapshots at times (a) t = 2.5, (b) t = 5.0, and (c) t = 7.5 of the time-varying three gyre flow for the example of Section 3.6.2. The period of the flow is 10. The time-invariant three gyre flow for the example of Section 3.6.1 corresponds to Figure 3.28(b).



(a) $t = 0$.

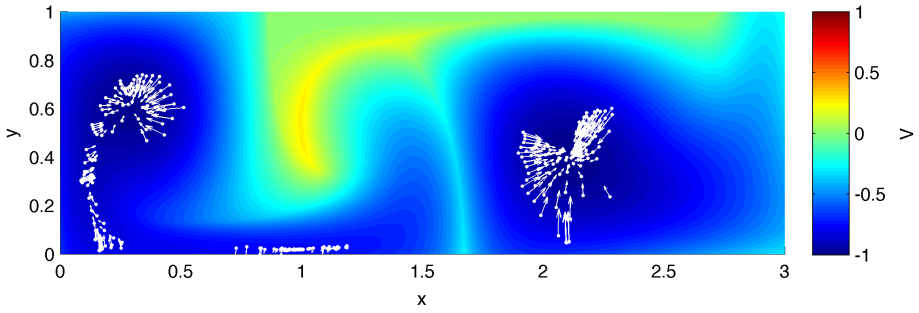


(b) $t = 2$.

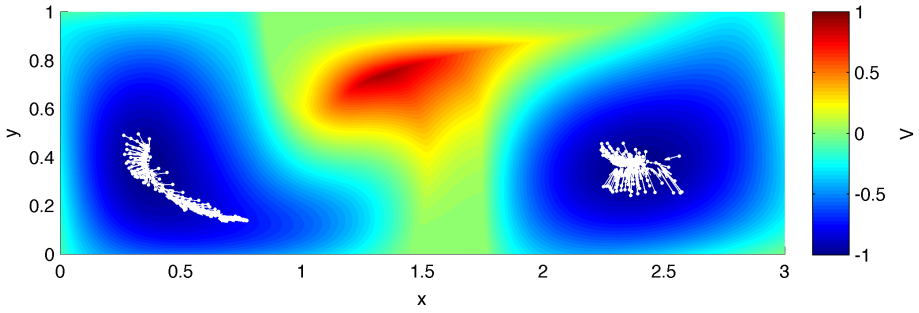


(c) $t = 4$.

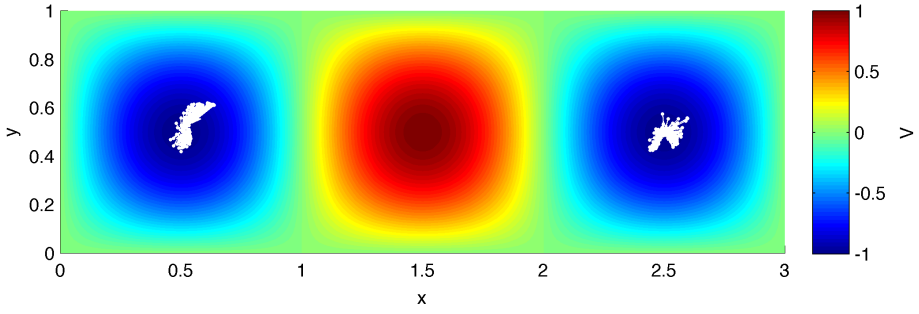
Figure 3.29: Initial time slices of the cost-to-go $V(x, y, t)$ (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.2. $V(x, y, 0)$ is significantly closer to zero than in the time-invariant result of Figure 3.19, thanks to the transport of the flow itself. Nevertheless there is a complex pattern of shocks (cusps in V). Between $t = 2$ and $t = 4$ four trajectories near $(x, y) \approx (2.2, 0.2)$ break away to the north, and the two main groups of trajectories become clear.



(a) $t = 6$.

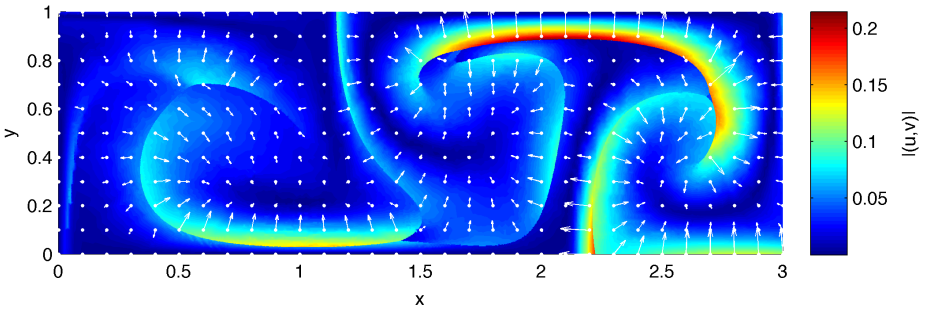


(b) $t = 8$.

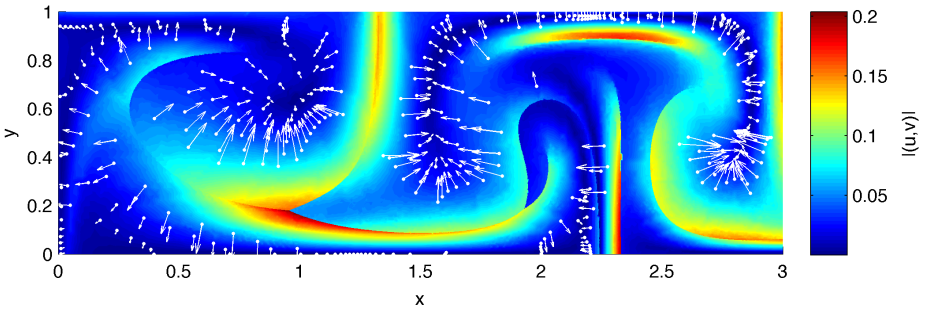


(c) $t = 10$.

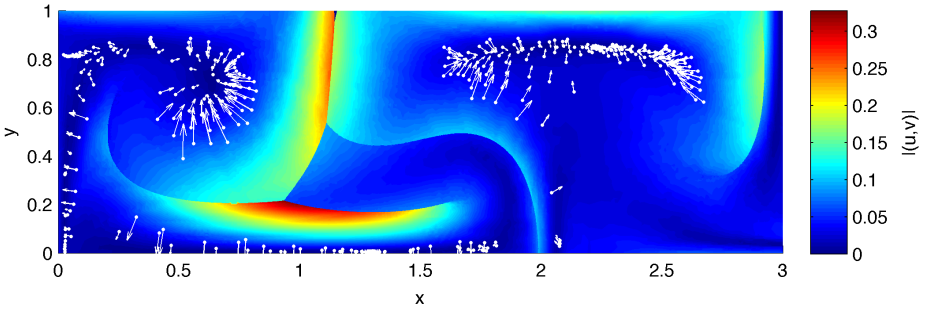
Figure 3.30: Final time slices of the cost-to-go $V(x, y, t)$ (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.2. Between $t = 4$ and $t = 8$ a small group of trajectories gets left behind and loiters near the moving stagnation point at $(x, y) \approx (1, 0)$ before rejoining the lefthand group.



(a) $t = 0$.

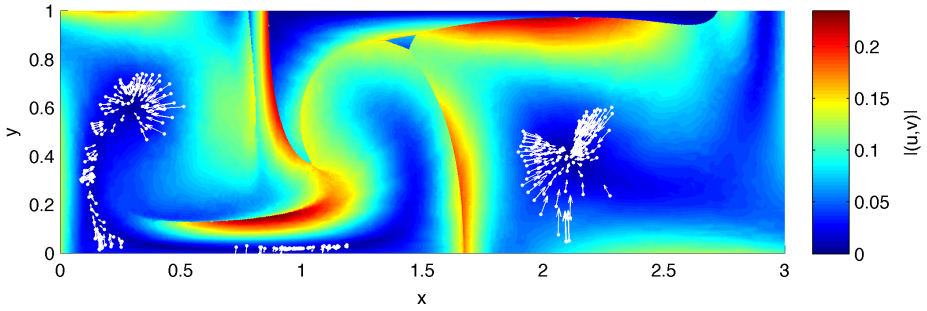


(b) $t = 2$.

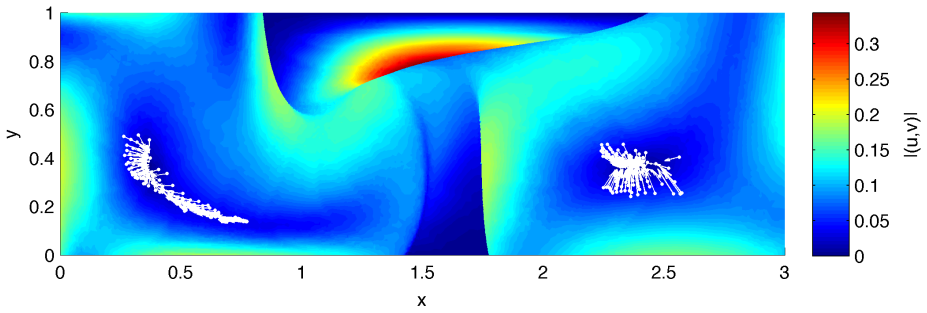


(c) $t = 4$.

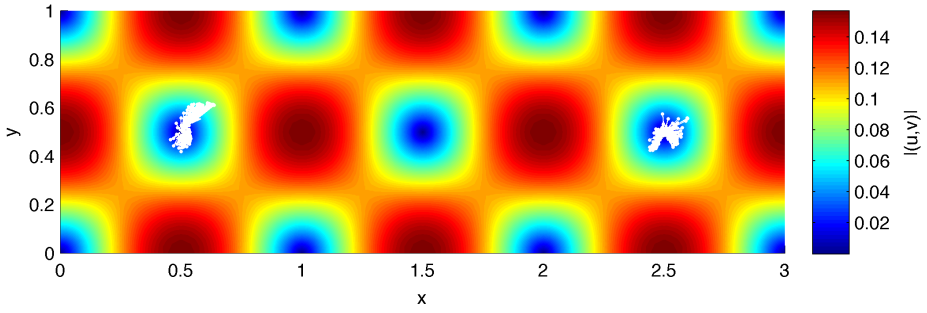
Figure 3.31: Initial time slices of the size of the optimal feedback control law $(u, v)(x, y, t)$ (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.2. The shocks appear here as discontinuities in (u, v). The red color at just to the outside of the two prominent horizontal shocks at $t = 2$ essentially repel nearby trajectories toward the moving stagnation points at $(x, y) \approx (1, 0)$ and $(2, 1)$, so that they don't get caught in the nearby jets.



(a) $t = 6$.

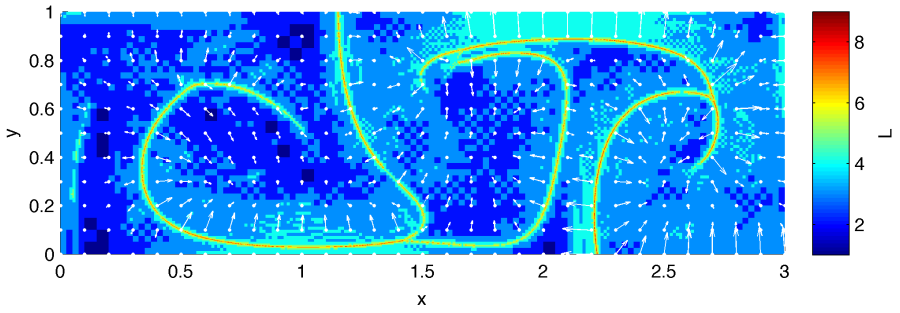


(b) $t = 8$.

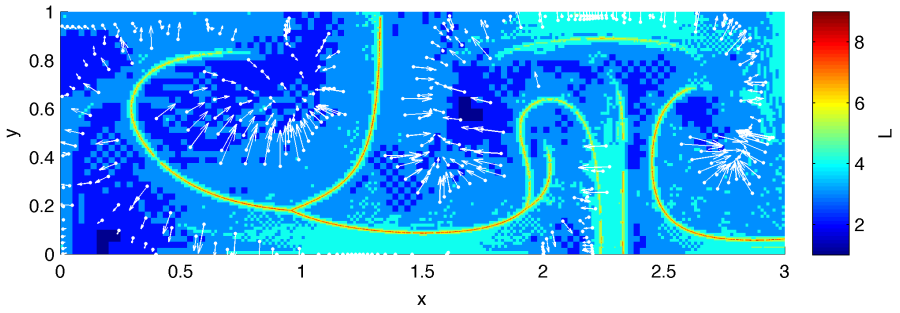


(c) $t = 10$.

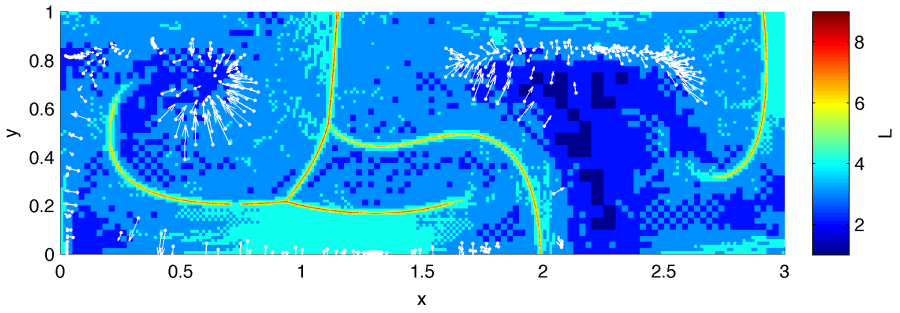
Figure 3.32: Final time slices of the size of the optimal feedback control law $(u, v)(x, y, t)$ (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.2. By $t = 8$ there are just one mostly horizontal and two vertical shocks. Again, note that (u, v) is nowhere saturated and thus its size is proportional to the size of the gradient of V.



(a) $t = 0$.

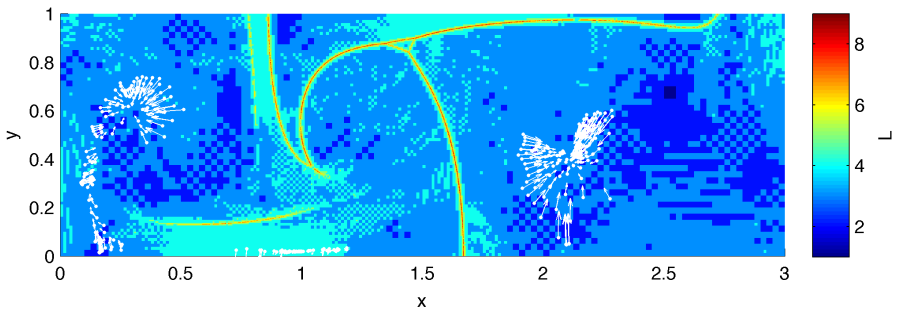


(b) $t = 2$.

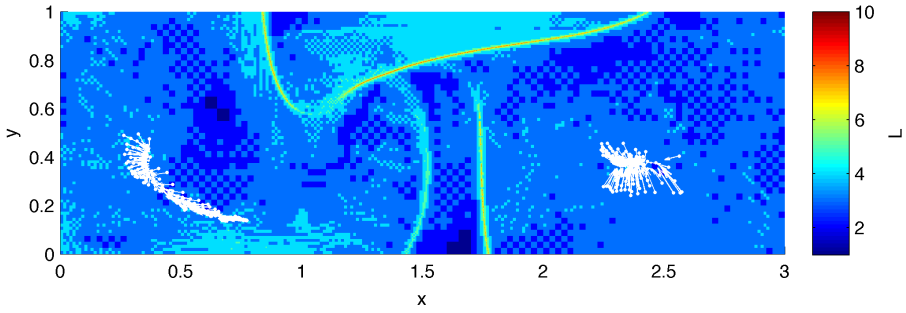


(c) $t = 4$.

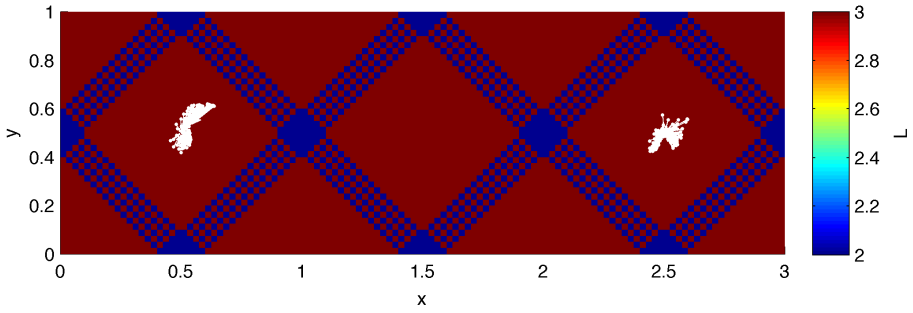
Figure 3.33: Initial time slices of the level of refinement L of the triangular grid elements (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.2. Again, an increase of one in L means a decrease in the area of a triangular grid element by a factor of four.



(a) $t = 6$.

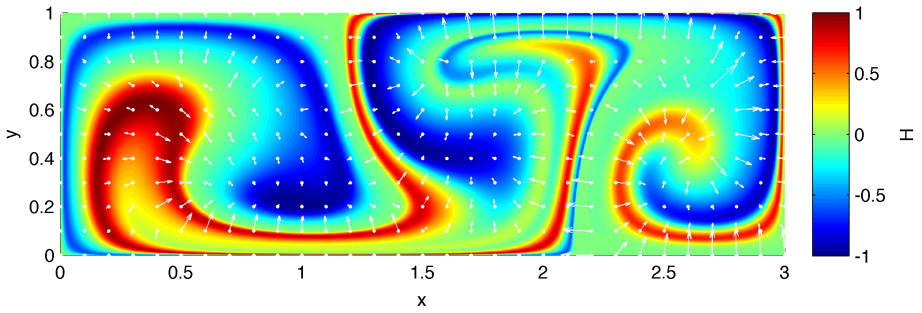


(b) $t = 8$.

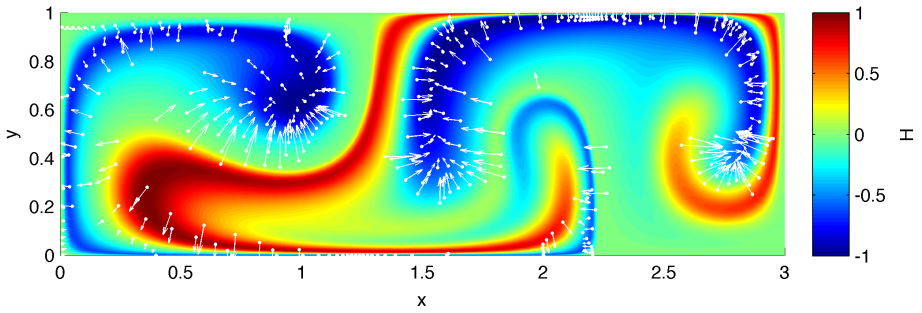


(c) $t = 10$.

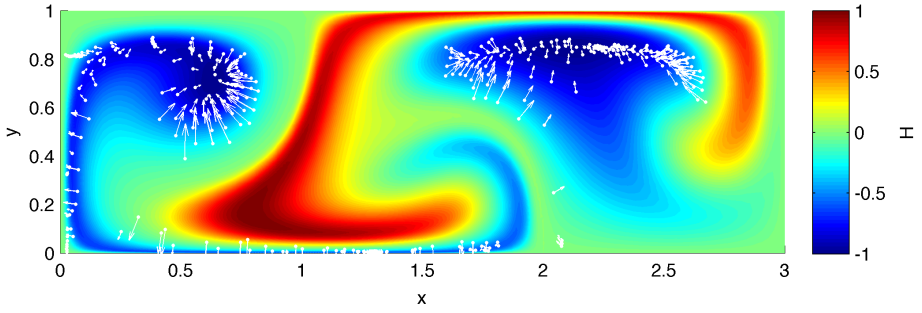
Figure 3.34: Final time slices of the level of refinement L of the triangular grid elements (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.2. See also Figures 3.30 and 3.32. Shocks appear, merge, separate, and eventually die out.



(a) $t = 0$.

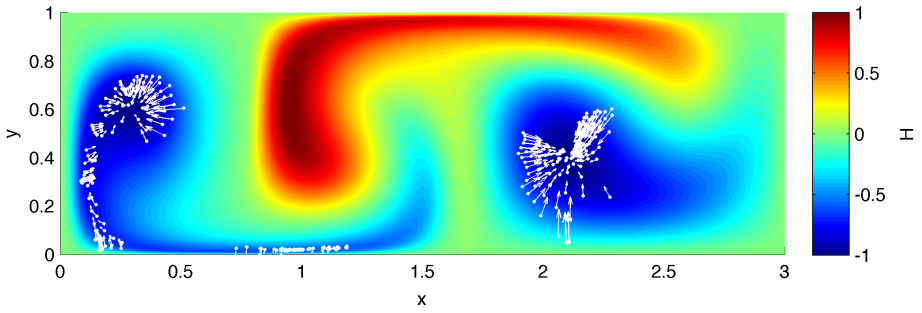


(b) $t = 2$.

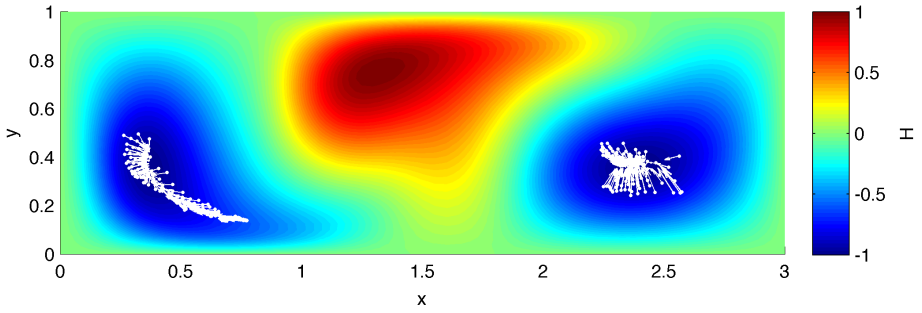


(c) $t = 4$.

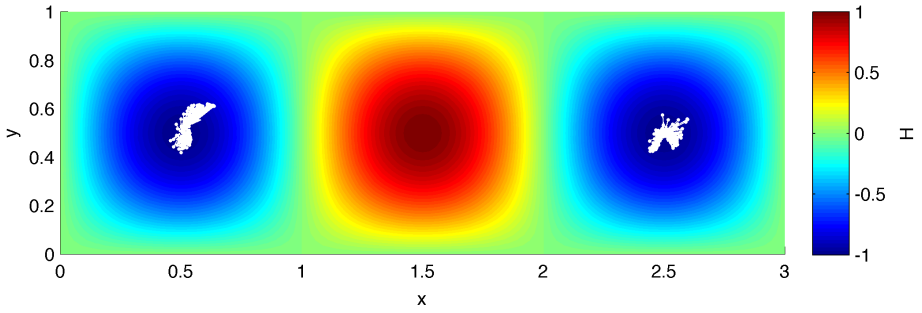
Figure 3.35: Initial time slices of the pulled back end cost function $H(x, y, t)$ (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.2. The non-monotonic descent of H highlights the non-greedy nature of the optimal control. The $t = 0$ adaptive grid for H is larger than that for the cost-to-go V, since H is smooth but sensitive.



(a) $t = 6$.



(b) $t = 8$.



(c) $t = 10$.

Figure 3.36: Final time slices of the pulled back end cost function $H(x, y, t)$ (color) and optimal trajectories (white markers and arrows) for the example of Section 3.6.2.

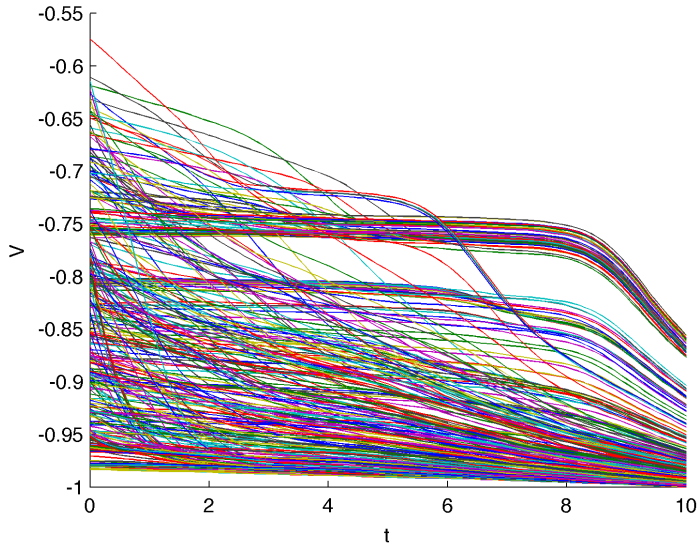


Figure 3.37: Cost-to-go V along optimal trajectories for the example of Section 3.6.2. Again, V is less overall than in the time-invariant case, and appears to decrease monotonically, as it should.

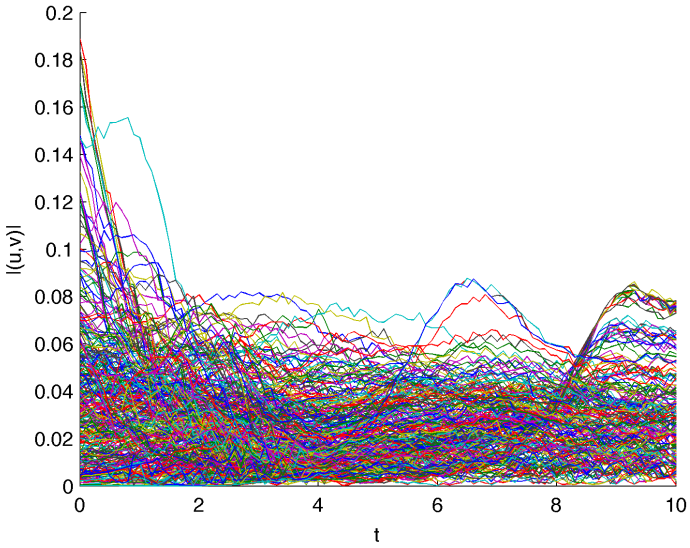


Figure 3.38: Size of control (u, v) along optimal trajectories for the example of Section 3.6.2. The size appears generally smaller than in the time-invariant result of Figure 3.26, which complements the observation that V is generally less. Again, the signals are not as smooth as desired. And again, note that the hard constraint $|u^2 + v^2| \leq s = 1$ turns out to have no effect (other than to scale the complexity of the point-wise optimization procedure).

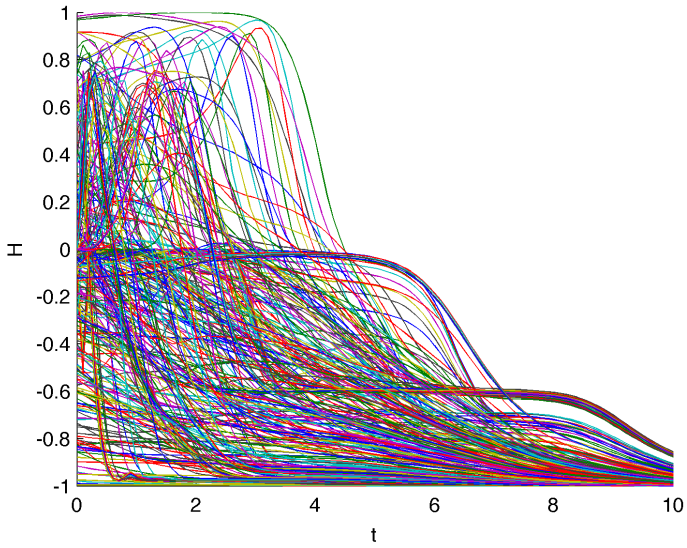


Figure 3.39: Pulled back end cost $H(x, y, t)$ along optimal trajectories for the example of Section 3.6.2. Unlike in the time-invariant case, this value appears to grow from as low as ≈ 0.2 to as high as ≈ 0.9, in order to ultimately decrease to < -0.8. As in the 1D examples of Section 3.4, this non-monotonic descent of the pulled back end cost function highlights the benefit of the optimal control over a more simpler, more greedy gradient descent algorithm.

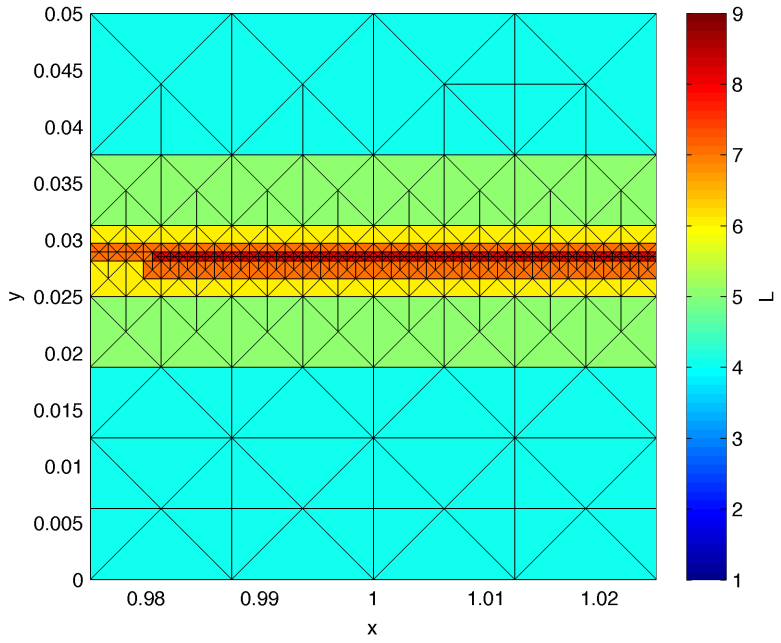


Figure 3.40: Close up of triangular adaptive grid elements and level of refinement L along one of the shocks shown in Figure 3.33(a)—a cusp in $V(x, y, 0)$ almost invisible in Figure 3.29(a). An increase of one in L means bisecting a triangle and its two children. No explicit bound on L was needed. The refinement criterion is simply $\Delta V > \Delta V_{\max}$, where ΔV is a linear error estimate, not an edge difference.

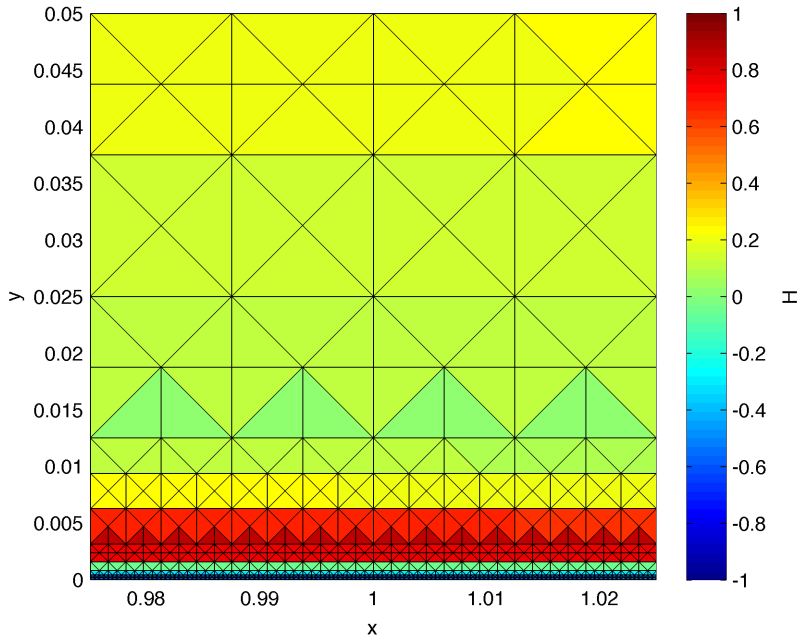


Figure 3.41: Close up of triangular adaptive grid elements and pulled back end cost $H(x, y, 0)$ along a ridge barely visible in Figure 4.8(a). Unlike the cost-to-go V, H is smooth but still very sensitive.

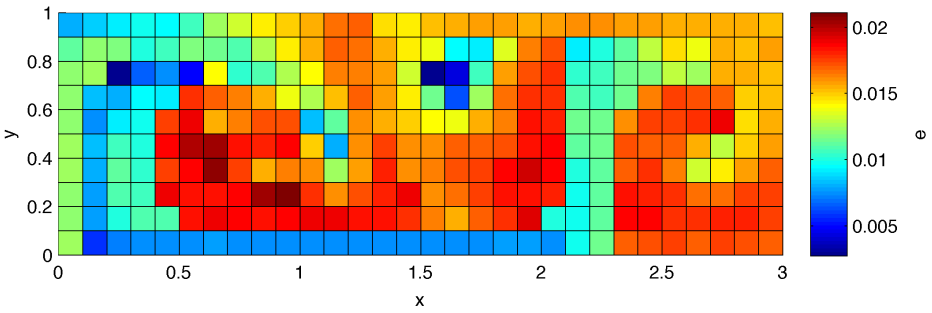


Figure 3.42: Estimated error $e(x, y, 0)$ (defined in Section 3.3.4) of the 300 simulated optimal trajectories for the example of Section 3.6.2. The maximum value of ≈ 0.02 is unsatisfactorily large, due to the limitation of the present semi-Lagrangian HJB PDE scheme to first order accurate spatio-temporal discretizations and the computational complexity of the exact point-wise optimization procedure with respect to the grid size.

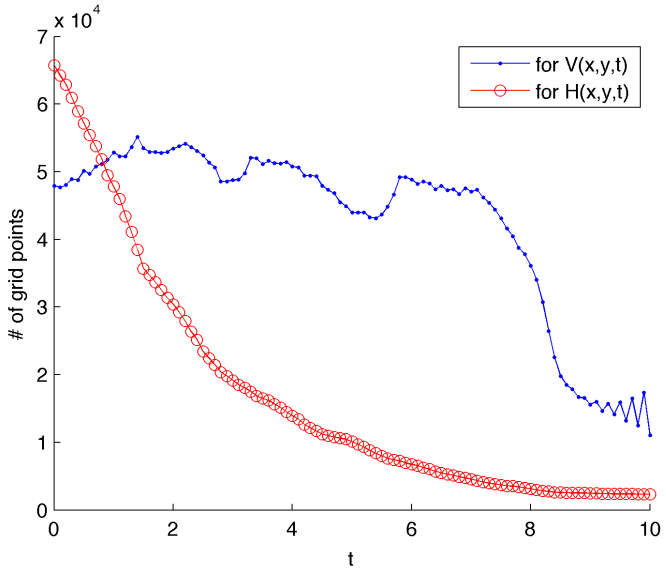


Figure 3.43: Number of grid points in the adaptive triangular grids for the cost-to-go V and pulled back end cost function H at each time step in the example of Section 3.6.2. The grid size typically peaks for V but grows exponentially for H

Chapter 4

Feedback Control and the Flow Map

4.1 Overview

In this chapter we study the relationship between the optimal control and the structure of the flow itself in terms of the flow map $\mathbf{X}$ and what we call the pulled back end cost function H, defined briefly in Chapters 1 and 3, respectively. H leads to a simple suboptimal control law which is tested against the optimal control in 1D and 2D. Some of the 1D results have been published in [60]. The problem is the same fixed final time minimum energy problem as in Chapter 3.

The more popular application of the flow map has been to compute a scalar field known as the finite time Lyapunov exponent (FTLE), the ridges of which are used to identify Lagrangian coherent structures (LCS) [35]. These structures act as transport barriers and indicate where the effect of small actuations on the ultimate destination of a vehicle is the greatest. The FTLE field plays an

important role but it is not the sole lens through which the flow map should be exploited for control, as seems to be the case in the more heuristic approaches of [37] and [38].

The chapter is organized as follows. In Section 4.2 we provide the definition of the flow map $\mathbf{X}$ and its Jacobian $\mathbf{X}_{\mathbf{x}}$, proving their key properties. Then we show how the fixed final time flow map yields a useful coordinate transformation for the fixed final time optimal control problem. In the transformed coordinates the flow is zero, but the control input is multiplied by the Jacobian of the flow map—a time-varying matrix. In Section 4.3 we define the pulled-back end cost function and prove a bound between it and the optimal cost-to-go function for the special case where control effort is not penalized i.e. $W = 0$ (which is not actually considered in Chapter 3 or elsewhere but relevant conceptually). In Section 4.4 we propose a general class of feedback control laws which we call local Lagrangian control (LLC), and three specific LLC laws. One of these reduces to the optimal control for linear time-invariant 1D flows and quadratic end costs, and another reduces to the optimal control for spatially invariant 1D flows and quadratic end costs. In Sections 4.5 and 4.6 we test the latter LLC law against the optimal control for the time-varying 1D example of Section 3.4.3 and for the time-varying 2D three-gyre example of Section 3.6.2, respectively. Finally in Section 4.7 we conclude the chapter.

4.2 Transformed problem: zero flow

We begin by restating the fixed final time minimum energy problem using general (nD) vector notation. Given an initial state $\mathbf{x}_0 \in \mathrm{D} \subset \mathrm{R}^n$ and an initial time $t_0 \in [0, t_f]$, minimize the cost functional

$$J(\mathbf{u}(\cdot); \mathbf{x}_0, t_0) := \int_{t_0}^{t_f} W \cdot \mathbf{u}(t) \cdot \mathbf{u}(t) \cdot dt + h(\mathbf{x}(t_f), t_f) \tag{4.1}$$

with respect to $\mathbf{u}(\cdot)$, subject to the constraint

$$\mathbf{u}(t) \in \Omega := \{\mathbf{u}' \in \mathrm{R}^n : |\mathbf{u}'| \leq s\} \tag{4.2}$$

(for all $t \in [t_0, t_f]$) where the state trajectory $\mathbf{x}(\cdot)$—implicitly dependent on the control input signal $\mathbf{u}(\cdot)$—is the unique solution of the state equation

$$\dot{\mathbf{x}}(t) = \mathbf{v}(\mathbf{x}(t), t) + \mathbf{u}(t) \tag{4.3}$$

with the initial conditions

$$\mathbf{x}(t_0) = \mathbf{x}_0. \tag{4.4}$$

In this chapter pairs $(\mathbf{x}, \mathbf{u}) = (\mathbf{x}(\cdot), \mathbf{u}(\cdot))$ satisfying (4.2)-(4.3) will be referred to as *admissible*.

In this case the Hamiltonian is

$$\mathcal{H}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t) = (\mathbf{v}(\mathbf{x}, t) + \mathbf{u}) \cdot \mathbf{p} + W \cdot \mathbf{u} \cdot \mathbf{u} \tag{4.5}$$

and thus the HJB equation (1.13) is

$$V_t = -\min_{\mathbf{u} \in \Omega} \{(\mathbf{v} + \mathbf{u}) \cdot V_{\mathbf{x}} + W \cdot \mathbf{u} \cdot \mathbf{u}\} \tag{4.6}$$

and the boundary condition is $V(\mathbf{x}, t_f) = h(\mathbf{x})$ as before.

4.2.1 The flow map

As stated in Chapter 1 let the *flow map* $\mathbf{X} : D \times [0, t_f]^2 \to D$ be defined as the mapping of points $(\mathbf{x}, t)$ in space-time along the (passive particle) trajectories of the flow field $\mathbf{v}$ forward or backward to a given time T, that is, the solution of

$$\frac{d}{dT}\mathbf{X}(\mathbf{x}, t, T) = \mathbf{v}\left(\mathbf{X}(\mathbf{x}, t, T), T\right)$$

for which $\mathbf{X}(\mathbf{x}, t, t) = \mathbf{x}$. In addition, assuming the Jacobian of the flow $\mathbf{v}_\mathbf{x}$ is defined, the Jacobian of the flow map $\mathbf{X}_\mathbf{x} : D \times [0, t_f]^2 \to \mathrm{R}^{n \times n}$ is also defined. Later, we will also use "flow map" and $\mathbf{X}$ more loosely to refer to the fixed final time flow map $\mathbf{X}(\mathbf{x}, t) := \mathbf{X}(\mathbf{x}, t, t_f)$ and to the evaluation of the fixed final time flow map along a controlled trajectory $\mathbf{X}(t) := \mathbf{X}(\mathbf{x}(t), t)$, as determined by the context. Similarly, let $\mathbf{X}_\mathbf{x}(\mathbf{x}, t) := \mathbf{X}_\mathbf{x}(\mathbf{x}, t, t_f)$ and $\mathbf{X}_\mathbf{x}(t) := \mathbf{X}_\mathbf{x}(\mathbf{x}(t), t)$.

Lemma: transitive property of the flow map

For all $(\mathbf{x}, t, T, \tau) \in \mathrm{R}^n \times [0, t_f]^3$,

$$\mathbf{X}(\mathbf{X}(\mathbf{x}, t, \tau), \tau, T) = \mathbf{X}(\mathbf{x}, t, T). \tag{4.7}$$

The proof is as follows. Let $(\mathbf{x}, t, T, \tau) \in \mathrm{R}^n \times [0, t_f]^2$ and, for all $T \in [0, t_f]$,

$\mathbf{y}(T) := \mathbf{X}(\mathbf{x}, t, T)$ and

$\mathbf{z}(T) := \mathbf{X}(\mathbf{X}(\mathbf{x}, t, \tau), \tau, T)$.

Then, by the definition of $\mathbf{X}$,

$\mathbf{y}_T(T) = \mathbf{X}_T(\mathbf{x}, t, T) = \mathbf{v}(\mathbf{X}(\mathbf{x}, t, T), T) = \mathbf{v}(\mathbf{y}(T))$ and

$\mathbf{z}_T(T) = \mathbf{X}_T(\mathbf{X}(\mathbf{x}, t, \tau), \tau, T) = \mathbf{v}(\mathbf{X}(\mathbf{X}(\mathbf{x}, t, \tau)\tau, T), T) = \mathbf{v}(\mathbf{z}(T), T).$

and

$\mathbf{z}(\tau) = \mathbf{X}(\mathbf{x}, t, \tau) =: \mathbf{y}(\tau).$

That is, for all $T \in [0, t_f]$,

$\mathbf{y}_T(T) = \mathbf{v}(\mathbf{y}(T), T)$ and

$\mathbf{z}_T(T) = \mathbf{v}(\mathbf{z}(T), T),$

and $\mathbf{y}(\tau) = \mathbf{z}(\tau),$

i.e. $\mathbf{y}$ and $\mathbf{z}$ satisfy the same ODE and coincide at time τ. Hence, by the uniqueness of solutions of continuous ODEs, they are the same trajectory, i.e. for all $T \in [0, t_f]$,

$\mathbf{z}(T) = \mathbf{y}(T).$

That is,

$\mathbf{X}(\mathbf{X}(\mathbf{x}, t, \tau), \tau, T) = \mathbf{X}(\mathbf{x}, t, T).$

QED.

Lemma: invariance of the (fixed final time) flow map along passive particle trajectories

Suppose $(\mathbf{x}, t) \in \mathrm{D} \times [0, t_f]$ and $\mathbf{y}(T) := \mathbf{X}(\mathbf{x}, t, T)$ for all $T \in [0, t_f]$. Then, for all T_1 and $T_2 \in [0, t_f]$,

$$\mathbf{X}(\mathbf{y}(T_1), T_1, t_f) = \mathbf{X}(\mathbf{y}(T_2), T_2, t_f).$$

The proof is as follows.

$$\begin{aligned}
\mathbf{X}(\mathbf{y}(T_1), T_1, t_f) &:= \mathbf{X}(\mathbf{X}(\mathbf{x}, t, T_1), T_1, t_f), \\
&= \mathbf{X}(\mathbf{x}, t, t_f), && \text{by (4.7)}, \\
&= \mathbf{X}(\mathbf{X}(\mathbf{x}, t, T_2), T_2, t_f), && \text{by (4.7)}, \\
&=: \mathbf{X}(\mathbf{y}(T_2), T_2, t_f).
\end{aligned}$$

Note, the conclusion of the Lemma (that the flow map is constant along the nominal trajectory $\mathbf{y}$) is true if and only if, for all $T \in [0, t_f]$,

$$\frac{d}{dT}\mathbf{X}(\mathbf{y}(T), T, t_f) = 0$$

$\iff$ for all $T \in [0, t_f]$, $\quad \mathbf{X}_{\mathbf{x}}(\mathbf{y}(T), T, t_f) \cdot \mathbf{y}_T(T) + \mathbf{X}_t(\mathbf{y}(T), T, t_f) = 0$

$\iff$ for all $T \in [0, t_f]$,

$$\mathbf{X}_{\mathbf{x}}(\mathbf{y}(T), T, t_f) \cdot \mathbf{v}(\mathbf{y}(T), T) + \mathbf{X}_t(\mathbf{y}(T), T, t_f) = 0. \tag{4.8}$$

In particular,

$\mathbf{X}_{\mathbf{x}}(\mathbf{y}(t), t, t_f) \cdot \mathbf{v}(\mathbf{y}(t), t) + \mathbf{X}_t(\mathbf{y}(t), t, t_f) = 0.$

i.e. $\mathbf{X}_{\mathbf{x}}(\mathbf{x}, t, t_f) \cdot \mathbf{v}(\mathbf{x}, t) + \mathbf{X}_t(\mathbf{x}, t, t_f) = 0.$

In summary, the Lemma also states that the flow map $\mathbf{X}_{\mathbf{x}}$ solves the advection PDE $\mathbf{X}_{\mathbf{x}} \cdot \mathbf{v} + \mathbf{X}_t = 0$. The boundary condition is of course $\mathbf{X}_{\mathbf{x}}(\mathbf{x}, t_f, t_f) = \mathbf{x}$.

4.2.2 Theorem: change of the (fixed final time) flow map along controlled trajectories

Now consider the evaluation of the flow map and its Jacobian along an admissible trajectory terminating at the fixed final time $T = t_f$. In other words consider $\mathbf{X}(t) = \mathbf{X}(\mathbf{x}(t), t) = \mathbf{X}(\mathbf{x}(t), t, t_f)$ and $\mathbf{X}_\mathbf{x}(t) = \mathbf{X}_\mathbf{x}(\mathbf{x}(t), t, t_f)$ for $(\mathbf{x}, \mathbf{u})$ satisfying (4.2)-(4.4).

As such, we have the following theorem: $\frac{d}{dt}\mathbf{X}(t) = \mathbf{X}_\mathbf{x}(t) \cdot \mathbf{u}(t)$, or simply

$$\dot{\mathbf{X}} = \mathbf{X}_\mathbf{x} \cdot \mathbf{u}. \tag{4.9}$$

This transformed state equation is analogous to (4.3). The proof of (4.9) is as follows.

$\frac{d}{dt}\mathbf{X}(\mathbf{x}(t), t, t_f)$

$= \mathbf{X}_\mathbf{x}(\mathbf{x}(t), t, t_f) \cdot \dot{\mathbf{x}}(t) + \mathbf{X}_t(\mathbf{x}(t), t, t_f)$,

by the chain rule,

$= \mathbf{X}_\mathbf{x}(\mathbf{x}(t), t, t_f) \cdot [\mathbf{v}(\mathbf{x}(t), t) + \mathbf{u}(t)] + \mathbf{X}_t(\mathbf{x}(t), t, t_f)$,

by (4.3)

$= \mathbf{X}_\mathbf{x}(\mathbf{x}(t), t, t_f) \cdot \mathbf{u}(t) +$

$\mathbf{X}_\mathbf{x}(\mathbf{x}(t), t, t_f) \cdot \mathbf{v}(\mathbf{x}(t), t) + \mathbf{X}_t(\mathbf{x}(t), t, t_f)$,

by the distributive property,

$= \mathbf{X}_\mathbf{x}(\mathbf{x}(t), t, t_f) \cdot \mathbf{u}(t)$,

by Lemma 4.2.1 (4.8).

Thus the optimal control problem is transformed from $(\mathbf{x}, t)$ space to $(\mathbf{X}, t)$ space. If, for instance, $h(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_f|$, then instead of seeking a trajectory from $(\mathbf{x}_0, t_0)$ to (the vicinity of) $(\mathbf{x}_f, t_f)$, one seeks a trajectory from $(\mathbf{X}_0, t_0) := (\mathbf{X}(t_0), t_0)$ to $(\mathbf{X}_f, t_f) := (\mathbf{x}_f, t_f)$—since $\mathbf{X}(t_f) = \mathbf{X}(\mathbf{x}(t_f), t_f, t_f) = \mathbf{x}(t_f)$. Note that $\mathbf{x}(t)$ is easily recovered by $\mathbf{x}(t) = \mathbf{X}(\mathbf{X}(t), t_f, t)$, that is, $\mathbf{X}(\mathbf{X}(\mathbf{x}(t), t, t_f), t_f, t)$ (by (4.7)). Given this one-to-one mapping between $\mathbf{X}$ and $\mathbf{x}$, the HJB equation is well defined in the new space. Instead of being simultaneously advected by $\mathbf{v}$ and driven by $\mathbf{u}$ per (2.1), backwards in time, the optimal state trajectories and the surface of the value function V are driven solely by $\mathbf{X}_\mathbf{x} \cdot \mathbf{u}$; the maximum size of the effective control depends on the direction of $\mathbf{u}$ but it is zero if (and only if) $\mathbf{u} = 0$. It's worth mentioning that our transformed trajectories $\mathbf{X}(t)$ and their dynamics appear to be analogous to the concept of "streak lines" of a flow and the associated vector field derived in [67], which includes a potentially useful numerical method for computing them, but this is beyond the scope of this dissertation. Similarly, there may be some advantages to solving the HJB equation numerically in these new coordinates. However, this has not been attempted; for the purposes of this dissertation, the advantage of the transformed coordinates is primarily conceptual.

4.3 Approximation of V (for the $W = 0$ case) by the pulled-back end cost function H

We now temporarily depart from the assumption that $W > 0$ (maintained throughout Chapter 3 and throughout the rest of this Chapter) to present a theorem for an explicit bound on the difference between the value function V and what we will call the *pulled-back end cost function*, defined in terms of the flow map $\mathbf{X}$ as

$$H(\mathbf{x}, t) := h(\mathbf{X}(\mathbf{x}, t)),$$

where again $\mathbf{X}(\mathbf{x}, t) := \mathbf{X}(\mathbf{x}, t, t_f)$. Moreover, we assume the gradient is well-defined and given by $H_{\mathbf{x}}(\mathbf{x}, t) = h_{\mathbf{x}}(\mathbf{X}(\mathbf{x}, t)) \cdot \mathbf{X}_{\mathbf{x}}(\mathbf{x}, t)$. As with $\mathbf{X}$, we will also use $H(t) := H(\mathbf{x}(t), t)$ and $H_{\mathbf{x}}(t) := H_{\mathbf{x}}(\mathbf{x}(t), t)$ as needed (but not $H(\mathbf{x}, t, T)$ or $H_{\mathbf{x}}(\mathbf{x}, t, T)$).

The bound is relatively conservative. Nevertheless, it highlights the connection between the optimal control and the hyperbolicity of the flow field, and it provides theoretical grounds for suboptimal control schema based on the pulled-back end cost function, at least for small vehicle speed s and small time horizons $t_f - t$. Note that for $W = 0$ the optimal control is non-unique but V is still well-defined.

Although the methods of this chapter are not operator theoretic, it's important to point out the role here of the *Koopman operator*, also known as the composition operator. Specifically, if we temporarily re-define the end cost function $h(\mathbf{x})$ by

$h(\mathbf{x}, t)$, then the above definition of the pulled back end cost function H may be written as

$$H(\mathbf{x}, t) := C_{\mathbf{X}}(h)(\mathbf{x}, t),$$

where $C_{\mathbf{X}}$ is the Koopman operator for the fixed final time flow map $\mathbf{X}(\mathbf{x}, t)$, i.e.

$$C_{\mathbf{X}}(h) := (h \circ \mathbf{X}). \tag{4.10}$$

Assumptions:

1. $W = 0$ in (4.1).

2. There exists an upper bound $c_{\mathbf{v}}$ on the magnitude of the Jacobian $\mathbf{v}_{\mathbf{x}}$ of the flow.

3. There exists an upper bound c_h on the magnitude of the gradient $h_{\mathbf{x}}$ of the end cost.

Lemma: bound on flow map Jacobian

Suppose $(\mathbf{x}, t, T) \in \mathrm{D} \times [0, t_f]^2$. Then

$$|\mathbf{X}_{\mathbf{x}}(\mathbf{x}, t, T)| \leq \exp c_{\mathbf{v}}(T - t).$$

This follows intuitively from the fact that

$$\mathbf{X}_{\mathbf{x}}(\mathbf{x}, t, T) = I + \int_t^T \mathbf{v}_{\mathbf{x}}(\mathbf{X}(\mathbf{x}, t, \tau), \tau) \cdot \mathbf{X}_{\mathbf{x}}(\mathbf{x}, t, \tau) d\tau.$$

We do not yet have a more rigorous proof.

Lemma: bound on difference between (fixed final time) flow map and the final state

Suppose $(\mathbf{x}, t) \in \mathrm{D} \times [0, t_f]$ and $(\mathbf{y}(\cdot), \mathbf{u}(\cdot))$ is an admissible trajectory with $\mathbf{y}(t) = \mathbf{x}$. Then

$$|\mathbf{X}(\mathbf{x}, t, t_f) - \mathbf{y}(t_f)| \leq s(t_f - t) \exp\{c_{\mathbf{v}}(t_f - t)\}.$$

The proof is as follows. Consider

$$\mathbf{X}^{\mathbf{y}}(T) := \mathbf{X}(\mathbf{y}(T), T, t_f),$$

that is, the flow map evaluated along the controlled trajectory, for times $T \in [0, t_f]$. This "trajectory" can be written in integral form as

$$\mathbf{X}^{\mathbf{y}}(T) = \mathbf{X}^{\mathbf{y}}(t) + \int_t^T \frac{d}{dT}\mathbf{X}^{\mathbf{y}}(\tau)d\tau.$$

But also, by Theorem 4.2.2, we have $\frac{d}{dT}\mathbf{X}^{\mathbf{y}}(T) = \frac{d}{dT}\mathbf{X}(\mathbf{y}(T), T, t_f) = \mathbf{X}_{\mathbf{x}}(\mathbf{y}(T), T, t_f) \cdot$ $\mathbf{u}(T)$, and thus $\mathbf{X}^{\mathbf{y}}(T) = \mathbf{X}^{\mathbf{y}}(t) + \int_t^T \mathbf{X}_{\mathbf{x}}(\mathbf{y}(\tau), \tau, t_f) \cdot \mathbf{u}(\tau)d\tau$,

$\iff \mathbf{X}(\mathbf{y}(T), T, t_f) = \mathbf{X}(\mathbf{y}(t), t, t_f)$

$+ \int_t^T \mathbf{X}_{\mathbf{x}}(\mathbf{y}(\tau), \tau, t_f) \cdot \mathbf{u}(\tau)d\tau$,

$\iff \mathbf{X}(\mathbf{y}(T), T, t_f) = \mathbf{X}(\mathbf{x}, t, t^{\max})$

$+ \int_t^T \mathbf{X}_{\mathbf{x}}(\mathbf{y}(\tau), \tau, t_f) \cdot \mathbf{u}(\tau)d\tau$,

and in particular

$\mathbf{X}(\mathbf{y}(t_f), t_f, t_f) = \mathbf{X}(\mathbf{x}, t, t_f)$

$+ \int_t^{t_f} \mathbf{X}_{\mathbf{x}}(\mathbf{y}(\tau), \tau, t_f) \cdot \mathbf{u}(\tau)d\tau$,

$\iff \mathbf{y}(t_f) = \mathbf{X}(\mathbf{x}, t, t_f)$

$+ \int_t^{t_f} \mathbf{X}_\mathbf{x}(\mathbf{y}(\tau), \tau, t_f) \cdot \mathbf{u}(\tau) d\tau,$

and thus

$\mathbf{X}(\mathbf{x}, t, 0) - \mathbf{y}(0) = -\int_t^{t_f} \mathbf{X}_\mathbf{x}(\mathbf{y}(\tau), \tau, t_f) \cdot \mathbf{u}(\tau) d\tau,$

which yields the bound

$$
\begin{aligned}
&|\mathbf{X}(\mathbf{x}, t, t_f) - \mathbf{y}(t_f)| \\
&\quad = \left| -\int_t^{t_f} \mathbf{X}_\mathbf{x}(\mathbf{y}(\tau), \tau, t_f) \cdot \mathbf{u}(\tau) \cdot d\tau \right|, \\
&\quad \leq \int_t^{t_f} |\mathbf{X}_\mathbf{x}(\mathbf{y}(\tau), \tau, t_f)| \cdot |\mathbf{u}(\tau)| \cdot d\tau, \\
&\quad \leq \int_t^{t_f} \exp\{c_\mathbf{v}(t_f - \tau)\} \cdot s \cdot d\tau, \\
&\qquad \text{by Lemma 4.3 and (4.2)}, \\
&\quad \leq \int_t^{t_f} \exp\{c_\mathbf{v}(t_f - t)\} \cdot s \cdot d\tau, \\
&\qquad \text{since } (t_f - \tau) \leq (t_f - t), \\
&\quad \leq \exp\{c_\mathbf{v}(t_f - t)\} \cdot s \cdot \int_t^{t_f} d\tau, \\
&\quad = \exp\{c_\mathbf{v}(t_f - t)\} \cdot s \cdot (t_f - t).
\end{aligned}
$$

4.3.1 Theorem: bound on the difference between value function and pulled-back end cost function

If the assumptions of Section 4.3 are satisfied, then

$$H(\mathbf{x}, t) - V(\mathbf{x}, t) \leq c_h s (t_f - t) \exp\{c_\mathbf{v}(t_f - t)\}.$$

This implies $H \to V$ exponentially as $t \to t_f$ and linearly as $s \to 0$.

The proof is as follows. By the definition of V, $\exists$ admissible $(\mathbf{y}, \mathbf{u})$ with $\mathbf{y}(t) = \mathbf{x}$ for which $V(\mathbf{x}, t) = h(\mathbf{y}(t_f))$. And

$$
\begin{aligned}
&H(\mathbf{x}, t) - V(\mathbf{x}, t) \\
&\quad = h(\mathbf{X}(\mathbf{x}, t, t_f)) - V(\mathbf{x}, t) \\
&\quad = |h(\mathbf{X}(\mathbf{x}, t, t_f)) - V(\mathbf{x}, t)| \\
&\quad = |h(\mathbf{X}(\mathbf{x}, t, t_f)) - h(\mathbf{y}(t_{max}))|, \\
&\quad \leq c_h |\mathbf{X}(\mathbf{x}, t, t_f) - \mathbf{y}(t_f)|, \\
&\quad \leq c_h s(t_f - t) \exp\{c_{\mathbf{v}}(t_f - t)\}, \quad \text{by Lemma 4.3.}
\end{aligned}
$$

4.4 Local Lagrangian Control (LLC) Laws

Unlike $\mathbf{x}$, the transformed state $\mathbf{X}$ can be driven in any direction by the control $\mathbf{u}$, regardless of the strength of the flow. The caveat is that there are generally fast and slow directions, especially if the flow is approximately area-preserving. Specifically, the effect of $\mathbf{X}_\mathbf{x}$ on $\mathbf{u}$ is a combination of rotation and scalar multiplication determined by the average ellipticity and/or hyperbolicity of the flow along the associated passive particle trajectory. Roughly speaking, this effect grows exponentially backward in time. Some insight can be gained by observing the maximum of the speed $|\dot{\mathbf{X}}| = |\mathbf{X}_\mathbf{x}\mathbf{u}|$ with respect to $\mathbf{u}$—essentially the FTLE field; however, this does not take into account the given control objective.

A better approach is to consider the effect on the time derivative of the pulled back end cost function $\frac{d}{dt}H(t) = \frac{d}{dt}h(\mathbf{X}(t)) = h_\mathbf{x}(\mathbf{X}(t)) \cdot \dot{\mathbf{X}}(t) = h_\mathbf{x}(\mathbf{X}(t)) \cdot \mathbf{X}_\mathbf{x}(t) \cdot$

$\mathbf{u} = H_{\mathbf{x}}(t) \cdot \mathbf{u}(t) = H_{\mathbf{x}}(\mathbf{x}(t), t) \cdot \mathbf{u}(t)$ by a control of the form

$$\mathbf{u} = -\tilde{s}\frac{H_{\mathbf{x}}}{|H_{\mathbf{x}}|},$$

where $s \geq \tilde{s} = \tilde{s}(\mathbf{x}, t) \geq 0$. This general class of control laws is what we call *local Lagrangian control* (LLC). LLC is greedy in the sense that it can only descend $H(\mathbf{x}, t) = h(\mathbf{X}, t)$ monotonically; it heeds only what's happening in the immediate vicinity of the passive particle trajectory that it's on. But it does look ahead in time.

Next we propose and test three specific LLC laws in 1D. Each law is ultimately modified to satisfy the constraint $u \in [-s, s]$, but this is ignored in much of what follows.

- *LLC law 1* is defined by

$$u(x, t) = -\frac{1}{2W}H_x, \tag{4.11}$$

  and follows from the approximation of V by H (discussed in Section 4.3) and thus of V_x by H_x in the analytical optimum $u = -\frac{1}{2W}V_x$ of the HJB equation (4.6).

- *LLC law 2* is defined by

$$u(x, t) = -\frac{1}{2\left(W + \frac{X_x^2 - 1}{2v_x}\right)}H_x, \tag{4.12}$$

  where recall $v_x = v_x(x, t)$ is the Jacobian of the flow v and $X_x = X_x(x, t)$ is the Jacobian of the flow map. This law is equivalent to the optimal control

for the special case where the flow is linear and time-invariant and the end cost is quadratic, i.e. $v(x,t) = ax$ and $h(x) = x^2$, as shown below. For $v_x = 0$ law 2 is defined by law 3.

- *LLC law 3* is defined by

$$u(x,t) = -\frac{1}{2(W + t_f - t)} H_x. \tag{4.13}$$

This law is equivalent to the optimal control for the special case where the flow is divergence-free and the end cost is quadratic, i.e. $v(x,t) = v(t)$ and $h(x) = x^2$, as shown below.

We do not consider the more obvious case of $\tilde{s}(\mathbf{x},t) = s$, relevant for the $W = 0$ "pure end cost" problem discussed briefly in Section 4.3 and closely related to the minimum time control of Chapter 2, because the optimal control for $W = 0$ is generally non-unique and because the optimal control for $W > 0$ is able to exploit the structure of the flow in more extreme and interesting ways.

Equivalence of LLC law 2 to the optimal control for 1D LTI flow and quadratic end cost

Suppose the flow v is linear and time-invariant i.e. $v(x,t) = ax$ for some scalar constant $a = v_x(x,t)$ and the end cost h is quadratic i.e. (without loss of generality) $h(x) = x^2$. From Equation (4.5), the Hamiltonian is $\mathcal{H}(x,u,p,t) = (ax + u)p + Wu^2$ and is minimized by $u = -\frac{p}{2W}$. Thus the HJB equation (4.6)

reduces to the Hamilton Jacobi (HJ) equation

$$V_t = \frac{1}{4W} V_x^2 - axV_x,$$

and the boundary condition is $V(x, t_f) = h(x) = x^2$. Rather than solve this equation directly, we look to the transformed problem.

The transformed coordinate i.e. the flow map is given by $X = xe^{a(t_f - t)}$ and its Jacobian is given by $X_x = e^{a(t_f - t)}$. Thus (4.9) reduces to

$$\dot{X} = e^{a(t_f - t)} u.$$

In addition, we define the transformed cost-to-go $V'(X, t) := V(x, t)$ and the transformed Hamiltonian $\mathcal{H}'(X, u, V'_X, t) := e^{a(t_f - t)} u V'_X + Wu^2$. The optimal control is $u = -\frac{1}{2W} e^{a(t_f - t)} V'_X$, and the transformed HJB equation is

$$V'_t = \frac{1}{4W} e^{2a(t_f - t)} (V'_X)^2, \tag{4.14}$$

with the boundary condition $V'(X, t_f) = X^2$, since $X = x$ at time $t = t_f$.

To solve this new, transformed problem, we look for a solution of the form $V'(X, t) = K(t)X^2$, and separate the variables. (4.14) yields

$K_t X^2 = \frac{1}{4W} e^{2a(t_f - t)} (2XK)^2$ (where $K_t := \frac{dK}{dt}$), which in turn yields the ODE

$$K_t = \frac{e^{2a(t_f - t)}}{W} K^2$$

and the end condition $K(t_f) = 1$ (or $X(x, t) = 0$, for which $u = 0$ yields the trivial optimum of $V' = h(0) = 0$). The solution of this initial value problem is

$$K = \frac{W}{W + \frac{e^{2a(t_f - t)} - 1}{2a}}$$

if $a \neq 0$ or $\frac{W}{W+t_f-t}$ if $a = 0$. This yields $u = -\frac{1}{2W}e^{a(t_f-t)}K(2X)$, that is,

$$u = -\frac{e^{a(t_f-t)}X}{W + \frac{e^{2a(t_f-t)}-1}{2a}} \tag{4.15}$$

if $a \neq 0$ or $-\frac{e^{a(t_f-t)}X}{W+t_f-t}$ if $a = 0$.

Law 2 (4.12) attempts to generalize (4.15), using the fact that $a = v_x$, $e^{a(t_f-t)} = X_x$, and thus the numerator is $\frac{1}{2}e^{a(t_f-t)}(2X) = \frac{1}{2}X_x h_x = \frac{1}{2}H_x$. Unlike V_x, X_x and thus H_x are technically smooth (assuming v and h are smooth) and easy to compute. However, it turns out that they are extremely sensitive. Specifically, this leads to chattering—bouncing back and forth of the trajectory—in the smooth but very narrow trenches that tend to appear in the surface of H. Hence instead of using (4.12) directly, we chose to substitute $e^{v_x(t_f-t)}$ for X_x, sacrificing some of the benefit of the knowledge of X_x in exchange for smoother trajectories. A method for implementing (4.12) more directly while avoiding chattering would be of interest.

Equivalence of LLC law 3 to the optimal control for 1D zero flow and quadratic end cost

Suppose the flow is divergence-free and the end cost is quadratic i.e. $v(x,t) = v(t)$ and (without loss of generality) $h(x) = x^2$. In this case the HJB equation is

$$V_t = \frac{1}{4W}V_x^2 - v(t)V_x,$$

but, as before, we look to the transformed problem.

The transformed state is $X(x,t) = x + \int_t^{t_f} v(\tau)d\tau$, the Jacobian is $X_x(x,t) = 1$, $H_x(x,t) = h_x(X(x,t))X_x(x,t) = 2X(x,t)$, and $\dot{X} = X_x u$ reduces to

$$\dot{X} = u.$$

Letting $V'(X,t) := V(x,t)$ and $\mathcal{H}'(X,u,p,t) := up + Wu^2$, the optimal control is $u = -\frac{1}{2W}p$, and the transformed HJB equation is

$$V'_t = \frac{1}{4W}(V'_X)^2, \tag{4.16}$$

with the boundary condition $V'(X,t_f) = X^2$, since $X = x$ at time $t = t_f$.

To solve this transformed problem, we look for a solution of the form

$$V'(X,t) = K(t)X^2,$$

and separate the variables. (4.16) becomes $K_t X^2 = \frac{1}{4W}(2XK)^2$ (where $K_t := \frac{dK}{dt}$). This yields $K_t = K^2/W$ or $X(x,t) = 0$ (for which $u = 0$ yields the trivial optimum of $V' = h(0) = 0$) and satisfies and $K(t_f) = 1$. The solution of this initial value problem is $K = \frac{W}{W+t_f-t}$ and the resulting optimal control is

$$u = -\frac{X}{W + t_f - t}.$$

In all three laws, the sensitivity of X_x is also reflected in H_x. However, we have found that the chattering issue can be avoided by simply computing H globally, and simulating the feedback control law using the same fixed time step semi-Lagrangian point-wise optimization scheme as in Chapter 3.

Global computation of LLC

Like the optimal control algorithm of Chapter 3, the present suboptimal control algorithm has two parts. First, it computes the flow map $X(x,t)$ and pulled back end cost $H(x,t)$ on a sequence of 1D adaptive grids. Second, it simulates feedback trajectories using the same algorithm as for the optimal control, but using H instead of V, and replacing W with the appropriate quantity from the denominator of (4.11), (4.12), or (4.13).

The computation of $H(x,t) = h(X(x,t))$ at a single point (x,t) comes down to a simple trajectory computation to obtain X; for simplicity and consistency we do this via the explicit Euler method. Unlike for V, H requires no pointwise optimization, and the time slices can be computed in any order (since we don't attempt to re-use later time slices in the computation earlier ones). However, we have found that for large enough times t_f H can take longer to compute than V for a given adaptive grid tolerance, $\Delta H_{\max} = \Delta V_{\max}$. Although H is smooth, it contains more variation than V, since there is no control input; hence the sensitivity of H with respect to x and the resulting chattering issues previously mentioned. Moreover, in 2D, the refinement criterion used in the end was $2^{L/2}\Delta H > \Delta_{\max}$, which turned out to be qualitatively better than either $\Delta H > \Delta_{\max}$ or $2^L \Delta H > \Delta_{\max}$.

This approach for computing the suboptimal control, globally, as with the optimal control, was chosen for convenience, and because the global picture of H

itself is very illustrative. However, one of the potential benefits of the suboptimal control over the optimal control would be the ability to compute it locally without computing it globally. The challenge would be the sensitivity and chattering issues.

4.5 1D Results: Time-Varying Example of Section 3.4.3

The three local Lagrangian control laws (4.11)-(4.13) were tested for the time-varying nonlinear 1D example of Section 3.4.3, with $e^{v_x(t_f-t)}$ replacing X_x in (4.12) as mentioned. Figures 4.1-4.4 show the detailed results for law 3 (4.13) alone, since it generally outperformed the other two, as shown in 4.5. The adaptive grid tolerance for computing H was $\Delta_{\max} =$ 1e-5.

The suboptimal trajectories are greedy in that they descend the pulled back end cost H monotonically. H is and H_x are shown in Figures 4.1 and 4.6 respectively. For example, the large negative gradients in H near $(x,t) = (0, 0.5)$ and $(2, 2.5)$ are sensed (not at a distance but rather locally) and rapidly descended by the two nearby trajectories (for which u saturates at $s = 1$ in Figure 4.4). These two are the only trajectories for which $u > 0$. This is because of how the positive gradients of H_x are squeezed by the stable fixed points at $x = 0$ and $x = 2$ in backward time. The gradient H_x is also very informative for the optimal control.

Compared to the optimal trajectories, the suboptimal trajectories generally arrive closer to the minima of the end cost, but expend more energy, shown in Figures 4.2 and 4.4 respectively. Instead of a long shock (a cusp in the cost-to-go V) at $x \approx 1.7$, the two groups of trajectories are delineated by the compact but smooth peak in H. Additionally, there is a smoother transition between the (few) trajectories for which $u > 0$ and the trajectories for which u is more negative.

The evaluation of the flow map itself, i.e. the transformed coordinate $X(t)$ is shown in Figure 4.7 for both the optimal trajectories and the suboptimal trajectories.

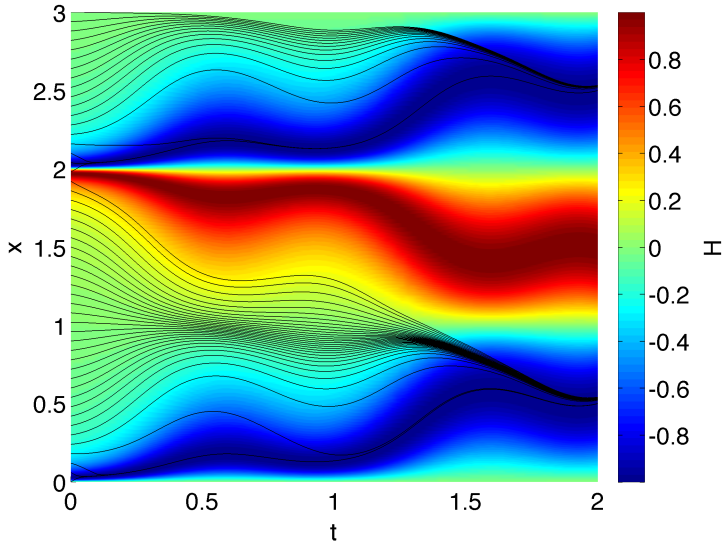


Figure 4.1: Pulled back end cost function $H(x,t)$ and trajectories resulting from LLC law no. 3 (4.13). The suboptimal trajectories compare fairly well to the optimal trajectories of Figure 3.10, but do not cross the red peak in H and tend to converge too quickly to the blue valley near $x = 2$.

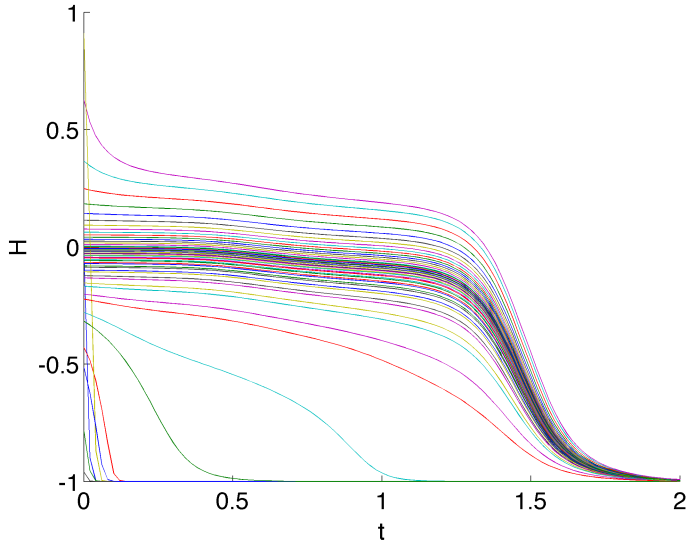


Figure 4.2: Pulled back end cost H evaluated along the trajectories of local Lagrangian control law no. 3 (4.13). Unlike the optimal trajectories of Figure 3.15 the suboptimal trajectories can only descend H monotonically.

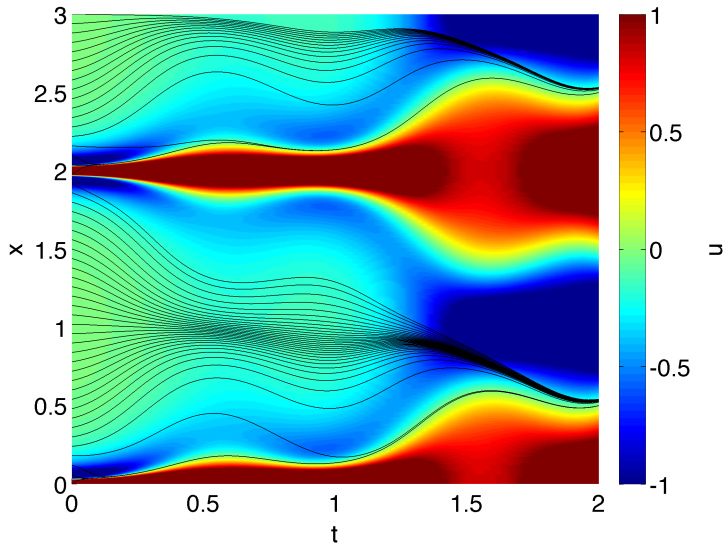


Figure 4.3: Suboptimal feedback control law $u(x,t)$ no. 3 (4.13) and resulting trajectories. u is technically smooth, but has an almost discontinuous jump from $-s=-1$ (blue) to $s=1$ (red) just slightly below $x=2$, exactly where $H(x,t)$ has a peak in Figure 4.1. This is slightly above where the optimal control law $u(x,t)$ has a discontinuity (a shock) in Figure 3.12.

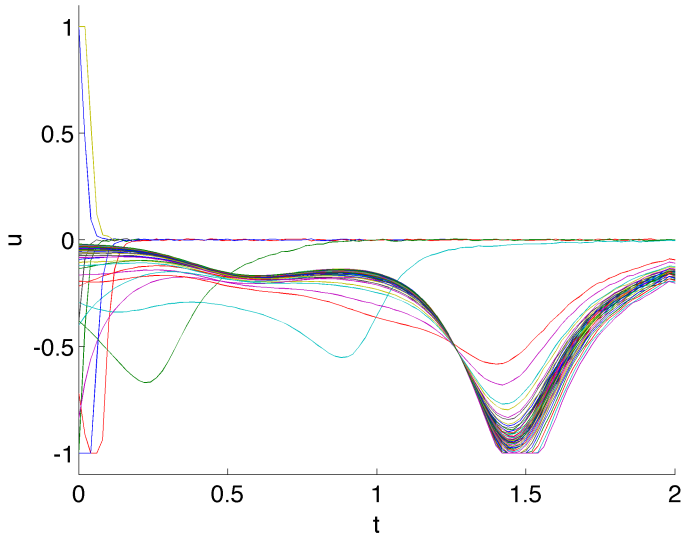


Figure 4.4: Control u along suboptimal trajectories of (4.13). Compare to optimal trajectories of Figure 3.13. Heuristic trajectories generally appear to consume more energy; in other words $|H_x|$ tends to overestimate $|V_x|$.

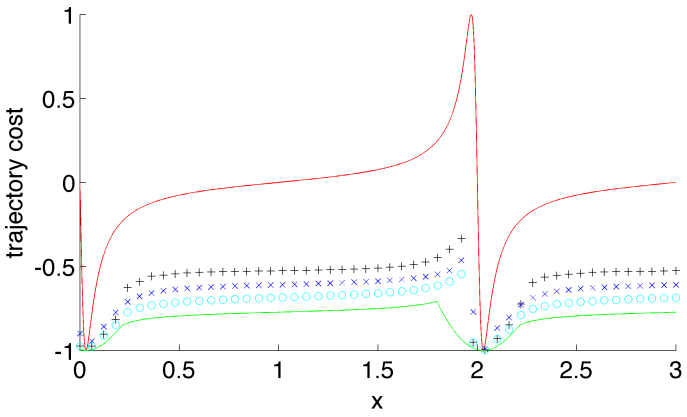


Figure 4.5: Total trajectory costs for three tested local Lagrangian control laws. The costs are bounded above by the pulled back end cost $H(x, 0)$ (the red curve) and below by the optimal cost-to-go $V(x, 0)$ (the green curve) of Figure 3.16 as they should be. LLC law 3 (4.13) (cyan os, bottom) outperformed both law 1 (4.11) (blue xs, middle) and law 2 (4.12) (black +s, top). Law 2 also outperformed law 1 for some points near the two minima.

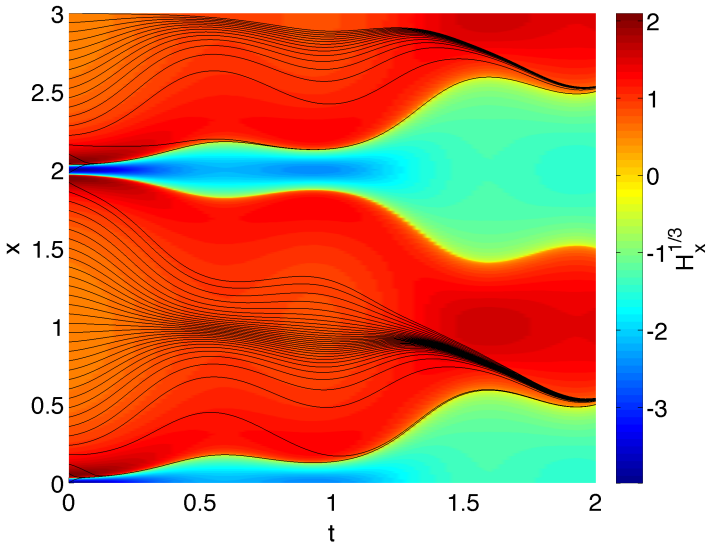
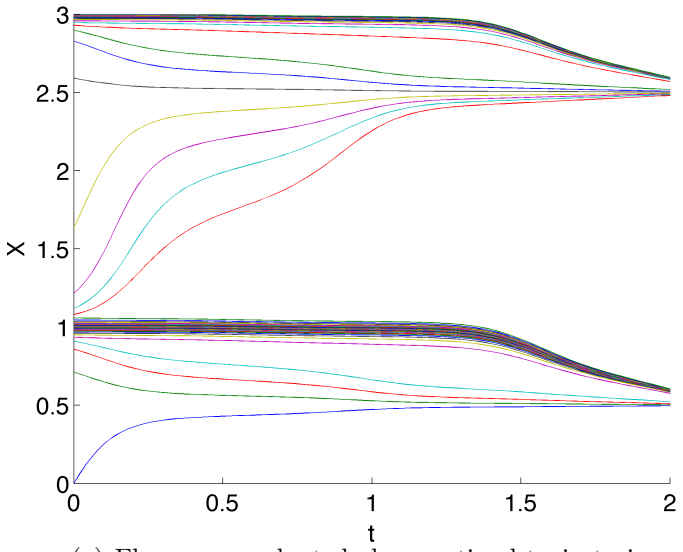
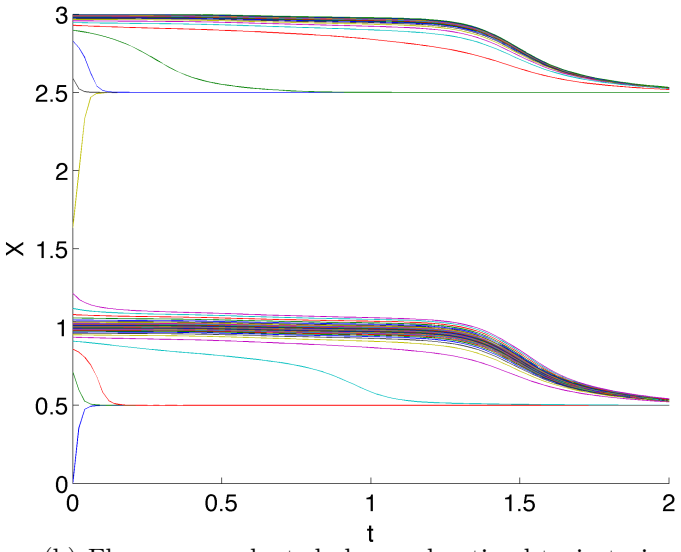


Figure 4.6: Gradient of pulled back end cost function $H_x(x,t)$ (scaled via cube root) and suboptimal trajectories $x(t)$ resulting from LLC law no. 3 (4.13). Recall $H_x = h_x X_x$ where h is the end cost and X_x is the Jacobian of the flow map X. In other words, H_x is the sensitivity of the future terminal cost h to the immediate control input u. Hence the proportionality of u to H_x in the proposed LLC laws. In the present case in particular, u also increases as the time remaining decreases; hence its saturation in much of the domain for $t = t_f = 2$ in Figure 4.3.



(a) Flow map evaluated along optimal trajectories



(b) Flow map evaluated along suboptimal trajectories

Figure 4.7: Transformed (a) optimal and (b) suboptimal trajectories $X(t)$ for the 1D time-invariant example of Section 3.4.3. Recall $X(t) := X(x(t), t) := X(x(t), t, t_f)$ is simply the evaluation of the fixed final time flow map along the trajectory $x(t)$ and $\dot{X} = X_x u$. $X = 0.5$ and 2.5 are the passive particle trajectories terminating at the desired minima of the end cost h. Fewer suboptimal than optimal trajectories end up near these points, but the ones that do approach them much more quickly, in other words too quickly.

4.6 2D Results: Time-varying Three-Gyre Example of Section 3.6.2

The extension of LLC law 3 (4.13) to 2D is straightforward and is given by $\mathbf{u}(\mathbf{x},t) = -\frac{1}{2(W+t_f-t)} H_{\mathbf{x}}(\mathbf{x},t)$ or

$$(u,v)(x,y,t) = -\frac{1}{2(W+t_f-t)}(H_x, H_y)(x,y,t), \tag{4.17}$$

with saturation at $|(u,v)| = s$. The equivalence to the optimal control follows as well for spatially-invariant flows $(f,g)(x,y,t) = (f,g)(t)$ and for the quadratic end cost $h(x,y) = x^2 + y^2$ but this obviously doesn't include all divergence free flows as in 1D and we don't show it rigorously.

This control was simulated for the time-varying three gyre flow of Section 3.6.2. The results are shown in Figures 4.8-4.22, and compared to the optimal trajectories. The maximum cost of a suboptimal trajectory was ≈ 0.4, as shown in Figure 4.20, compared to ≈ -0.4 for the optimal trajectoires. Measured relative to the minimum possible cost of -1, this is an increase by a factor of more than three 3; this isn't all that surprising given the local nature of the control, and the cost was more comparable to the optimal in much of the domain.

The bulk of the computation was to obtain the flow map $\mathbf{X}$ and its Jacobian $\mathbf{X}_{\mathbf{x}}$ from which the pulled back end cost H (shown along with snapshots of the suboptimal trajectories in Figures 4.8-4.9) and its gradient $H_{\mathbf{x}}$ and thus the LLC (4.17) are defined. This took on the order of half an hour, as mentioned in Section

3.6.2, where H was also visualized with the optimal trajectories. In addition is the marginal cost of simulating the controlled trajectories. $H_{\mathbf{x}}$ is used for visualization of $\mathbf{u}$ globally but recall that to avoid chattering issues the control defining the trajectories is computed by direct point-wise minimization of H at fixed time steps, in the same way as the optimal control is computed directly from V and not $V_{\mathbf{x}}$.

Snapshots of the size of the suboptimal control law and the suboptimal trajectories are shown in Figures 4.10 and 4.11. The suboptimal control is even less (numerically) smooth than the optimal control; it captures all of the sensitivity of H, which exists despite the fact that H is (analytically) smooth. It saturates at $s = 1$ in much of the domain, unlike the optimal control of Figures 3.31 and 3.32 (for which the color scales is different). Figures 4.18 and 4.19 show these things. In contrast, the gradient of H is "sensed" by the optimal control as well, but not just locally, so the control exploits the large negative gradients in a more paced manner, often ascending H en route.

The transformed state coordinate $\mathbf{X}(t) = \mathbf{X}(\mathbf{x}(t), t, t_f)$, that is, the evaluation of the fixed final time flow map along a trajectory $\mathbf{x}(t)$, is shown in Figure 4.22 for both the optimal and suboptimal trajectories, along with the original state coordinates in Figure 4.21. This perspective is discussed conceptually throughout this chapter. The idea is to isolate the dynamics of the vehicle position $\mathbf{x}$ *relative*

to the flow. The result is that $\dot{X} = 0 \iff \mathbf{u} = 0$ but that $\mathbf{u}$ no longer enters isotropically, since it is multiplied by the time-varying matrix $\mathbf{X_x}$.

The cost-to-go $V(\mathbf{x}, t) = V'(\mathbf{X}, t)$ is plotted versus $\mathbf{X}$ and t, along with the transformed optimal trajectories $\mathbf{X}(t)$ in Figures 4.12-4.13. Note that the relationship of the cost-to-go V and the the *pushed forward* cost-to-go V' is analogous to that of H and h, in that $V = C_{\mathbf{X}}(V')$, where $C_{\mathbf{X}}$ is the Koopman or composition operator defined by (4.10). To compute V' from V we use the inverse (backward in time) Koopman operator, computing the flow map backward in time from the grid points $\mathbf{X}$ at time t_f to yield non-grid points $\mathbf{x}$ at time t, and interpolating V (quadratically) at $\mathbf{x}$ from the adaptive triangular grid at time t.

Though not attempted in this dissertation, one could solve the HJB equation directly in $(\mathbf{X}, t)$ space, as discussed previously, perhaps simultaneously computing $\mathbf{X}$ and $\mathbf{X_x}$ in backward time. One of the potential advantages is that the transformed cost-to-go $V'(\mathbf{X}, t)$ decreases monotonically in backward time. As mentioned in Section 1.2.3, monotonicity like this is what allows one to replace the more general Level Set Methods for implicit front tracking with the faster Ordered Upwind Methods like the Fast Marching Method. One foreseen obstacle to deriving a similar method for the present time-varying case is that the performance of OUMs degrades as the minimum speed of the tracked front (in our case simply the norm of $\dot{\mathbf{X}} = \mathbf{X_x}\mathbf{u}$) goes to zero. The extreme dependence of the front speed on the direction of $\mathbf{u}$ is manifested by stretching in the level sets of V'. The

structures captured by $\mathbf{X_x}$ that guide this stretching and in turn the trajectories $\mathbf{X}(t)$ can be visualized more directly in terms of the finite time Lyapunov exponent (FTLE) field.

Figures 4.14-4.15 show a scaled version of the FTLE field, along with the transformed optimal trajectories $\mathbf{X}(t)$. In fact, the FTLE field is defined directly from the induced norm of $\mathbf{X_x}$, that is,

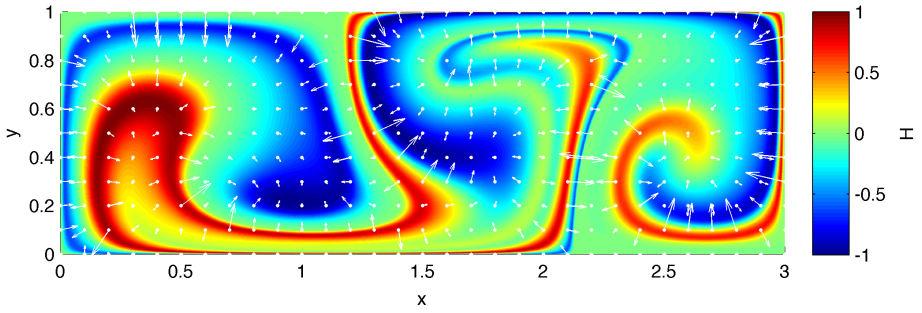
$$\max_{\mathbf{u}\in S^1} |\mathbf{X_x u}|,$$

i.e. the factor of increase in the transformed vehicle speed $\mathbf{X_x u}$ in the fastest direction. The induced norm is computed as the largest singular value of the symmetric matrix $\mathbf{X_x}^{\mathrm{T}}\mathbf{X_x}$—the square root of the maximum eigenvalue. The FTLE is defined as the natural logarithm of the induced norm divided by $|t_f - t|$, but we do not divide by time, since we are interested in the absolute speed in the $\mathbf{X}$ plane, not the time-averaged sensitivity of the flow. Moreover, the flow map $\mathbf{X}$ is computed backward in time from t_f to t. This "maximum speed" function is the most obvious way of visualizing the dynamics $\dot{\mathbf{X}} = \mathbf{X_x u}$. However, by multiplying $\mathbf{X_x}$ by its transpose, one throws away some of the information.

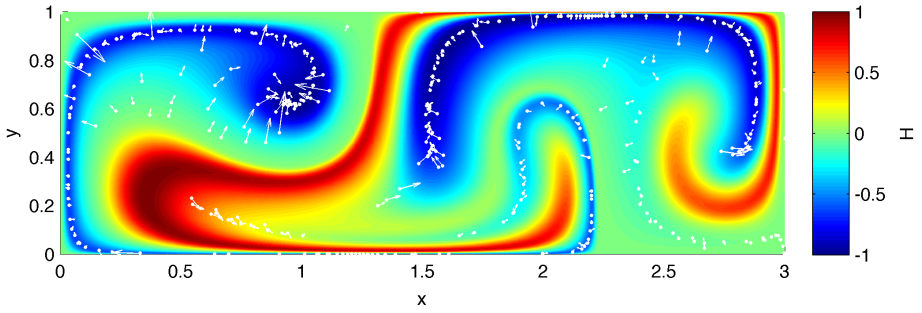
Figures 4.16-4.17 show (a scaled version of) the scalar field known as *mesohyperbolicity* first introduced in [36]. In forward time mesohyperbolicity is defined as

$$\det\left(\frac{\mathbf{X_x} - \mathrm{I}}{t_f - t}\right).$$

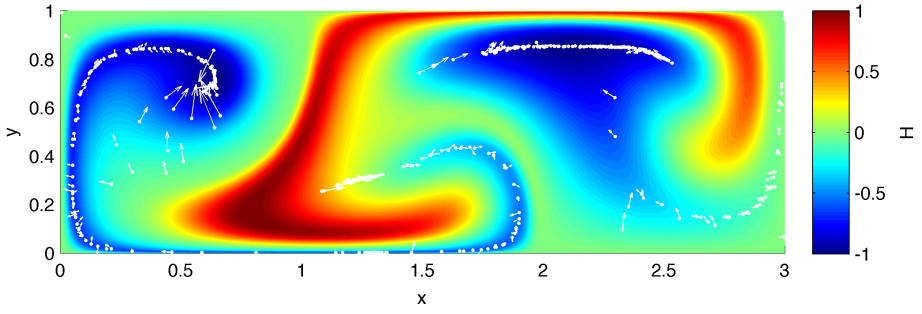
Like for the FTLE field, we instead compute this quantity in backward time, and in this case multiply it by $(t-t_f)^2$, so that the color values range from -4 to 8 for all time. Whereas the FTLE field is based on $\mathbf{X}_\mathbf{x}^\mathrm{T}\mathbf{X}_\mathbf{x}$ and thus doesn't consider the case of complex eigenvalues, mesohyperbolicity is based directly on $\mathbf{X}_\mathbf{x}$ itself and thus distinguishes not only the regions of hyperbolicity (stretching) in the flow but also ellipticity/rotation. Specifically, the blue, green, and red regions indicate ("meso-" or "on-average") hyperbolicity without rotation, ellipticity, and hyperbolicity with rotation, respectively.



(a) $t = 0$.

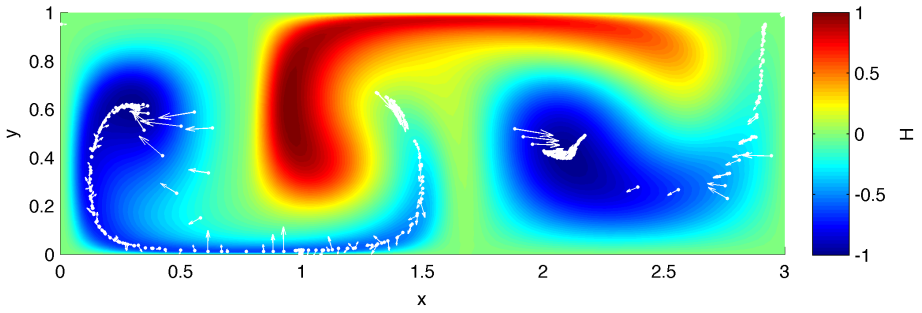


(b) $t = 2$.

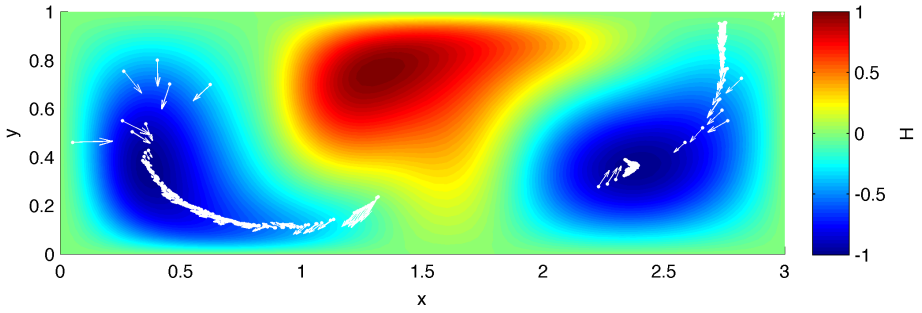


(c) $t = 4$.

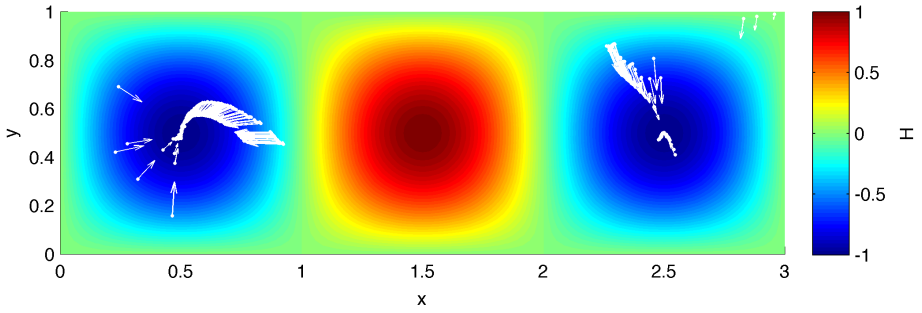
Figure 4.8: Initial time slices of the pulled back end cost function $H(x, y, t)$ (color) and suboptimal trajectories (white markers and arrows) for the time-varying gyre flow. The suboptimal trajectories are greedy in that they descend H monotonically, unlike the optimal trajectories shown in 3.35.



(a) $t = 6$.

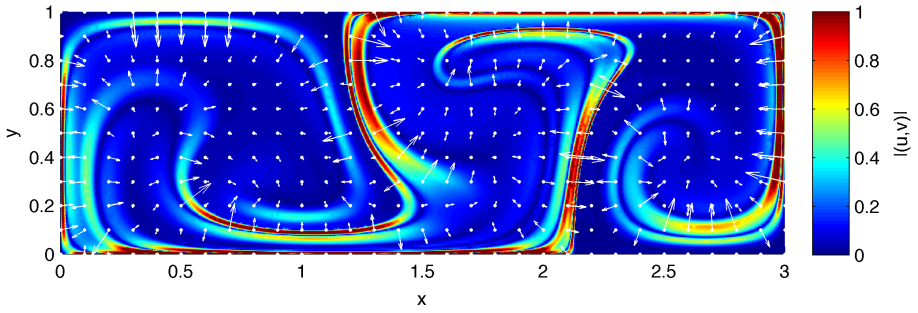


(b) $t = 8$.

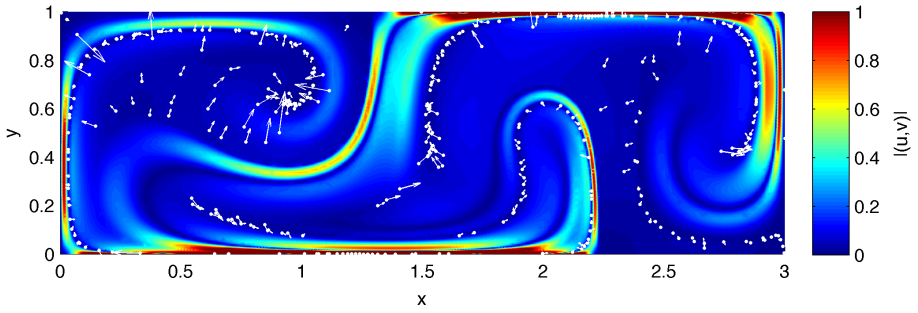


(c) $t = 10$.

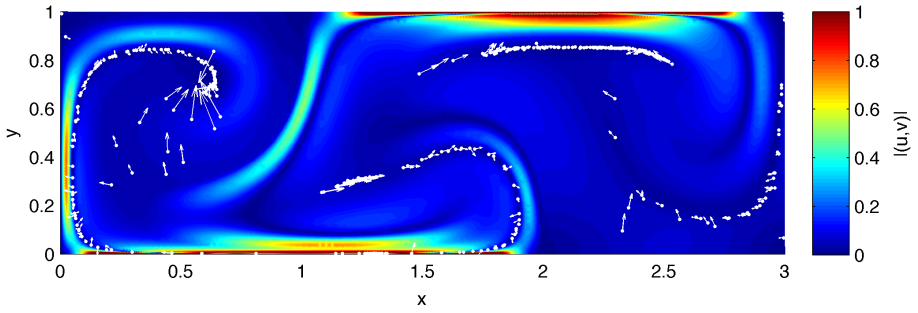
Figure 4.9: Final time slices of the pulled back end cost function $H(x, y, t)$ (color) and suboptimal trajectories (white markers and arrows) for the time-varying gyre flow



(a) $t = 0$.

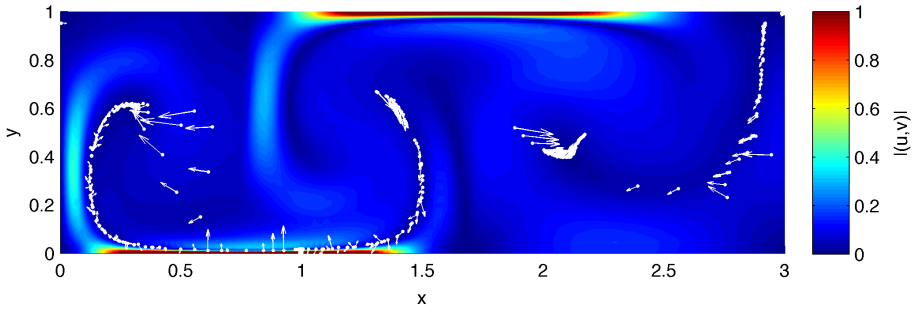


(b) $t = 2$.

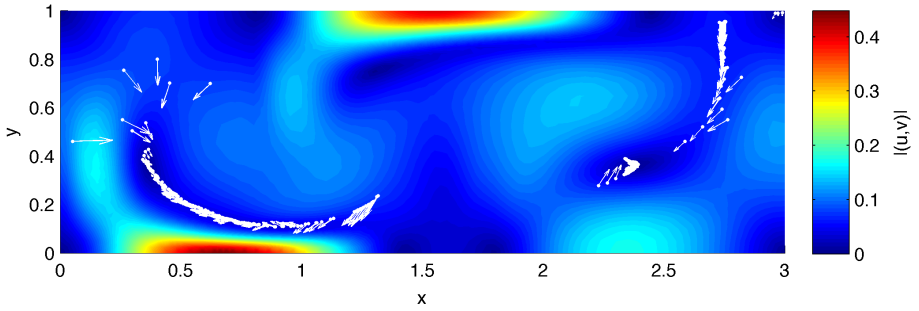


(c) $t = 4$.

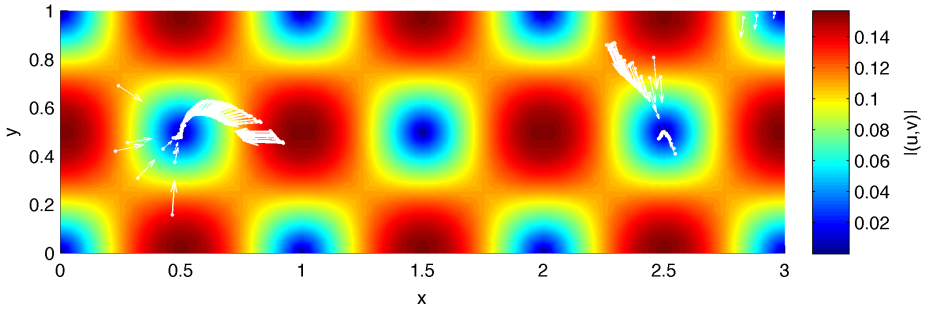
Figure 4.10: Initial time slices of the size of the suboptimal feedback control law $(u, v)(x, y, t)$ (color) and trajectories (white markers and arrows) for the time-varying gyre example. The suboptimal control is much larger than the optimal control shown in Figure 3.31, and saturated (red) in much of the domain.



(a) $t = 6$.

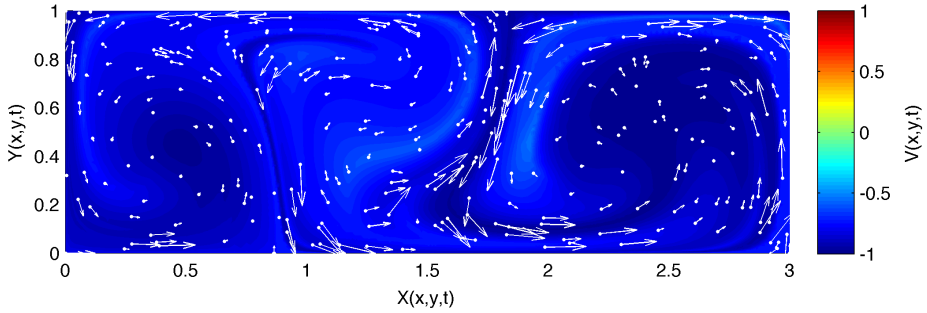


(b) $t = 8$.

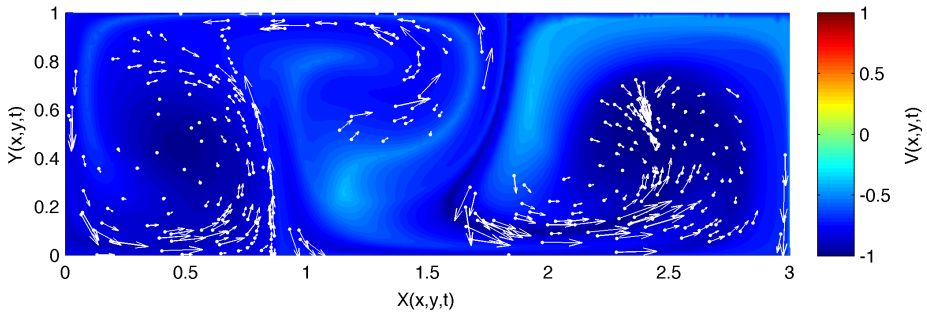


(c) $t = 10$.

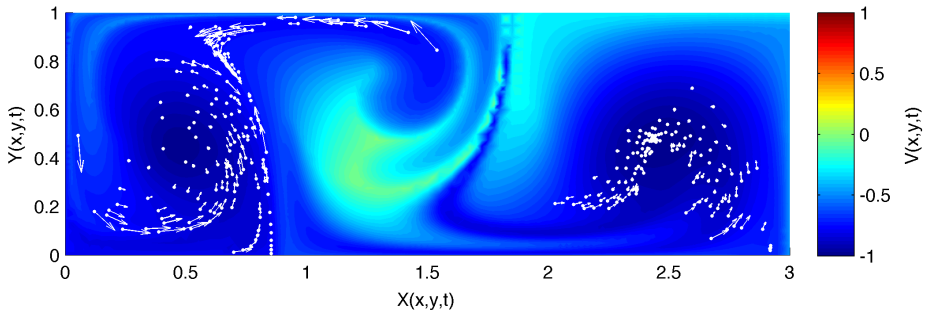
Figure 4.11: Final time slices of the size of the suboptimal feedback control law $(u, v)(x, y, t)$ (color) and trajectories (white markers and arrows) for the time-varying gyre example.



(a) $t = 0$.

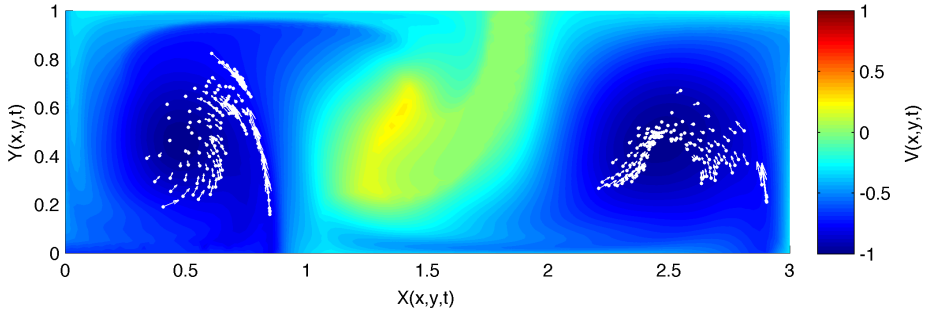


(b) $t = 2$.

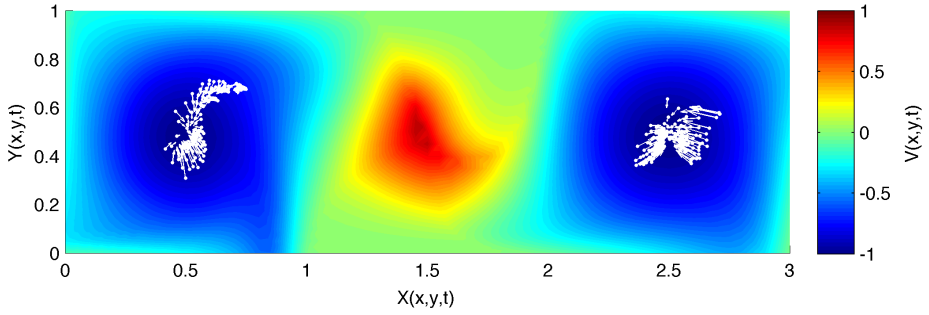


(c) $t = 4$.

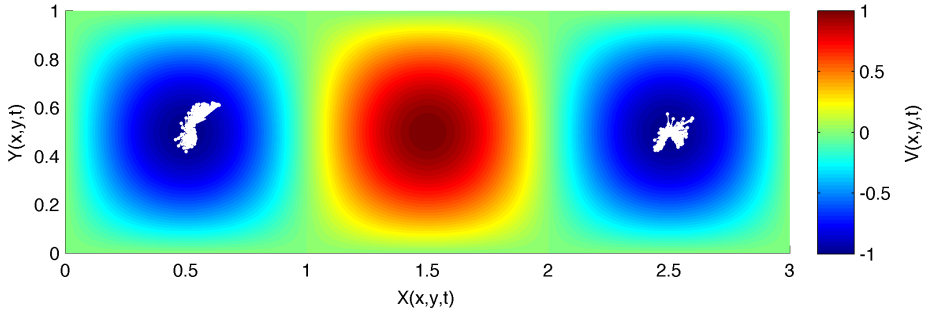
Figure 4.12: Initial time slices of the pushed forward cost-to-go $V'(X, Y, t)$ (color)—the transformed version of $V(x, y, t)$ of Figure 3.29—and the transformed optimal trajectories $(X(t), Y(t))$ (white markers and arrows) for the time-varying gyre flow. Recall in this transformed space $\dot{\mathbf{X}} = \mathbf{X}_\mathbf{x}\mathbf{u}$, i.e. the "flow" is zero but the control is multiplied by the time-varying matrix $\mathbf{X}_\mathbf{x}$. Notably, V' decreases monotonically in backward time.



(a) $t = 6$.

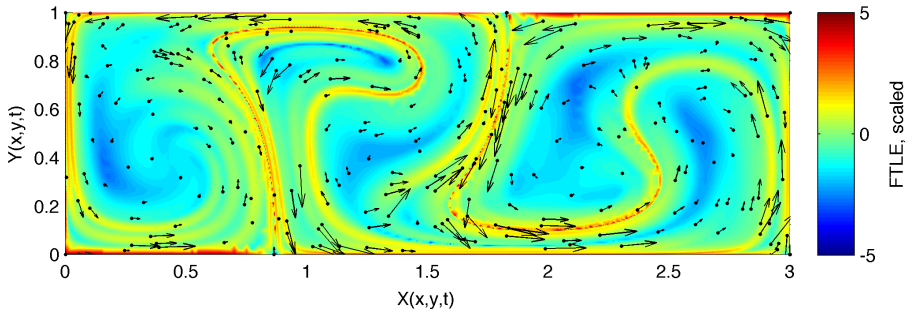


(b) $t = 8$.

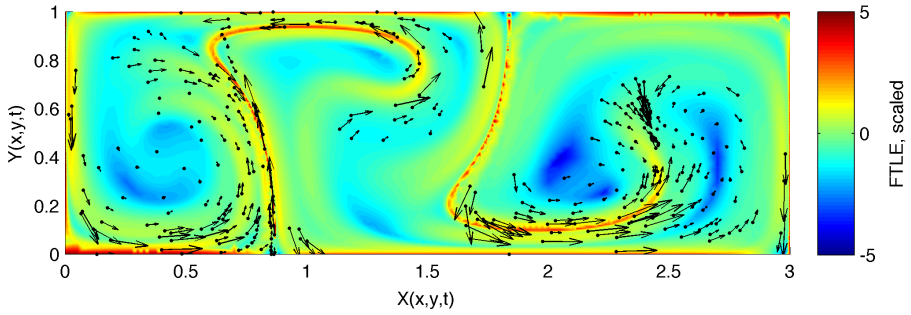


(c) $t = 10$.

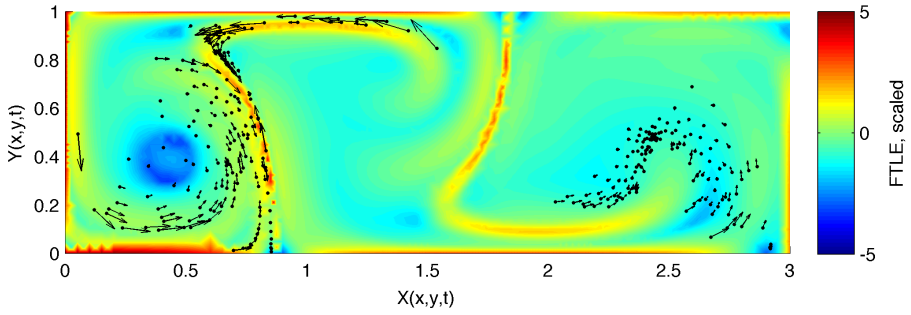
Figure 4.13: Final time slices of the pushed forward cost-to-go $V'(X, Y, t)$ (color)—the transformed version of $V(x, y, t)$ of Figure 3.30—and the transformed optimal trajectories $(X(t), Y(t))$ (white markers and arrows) for the time-varying gyre flow.



(a) $t = 0$.

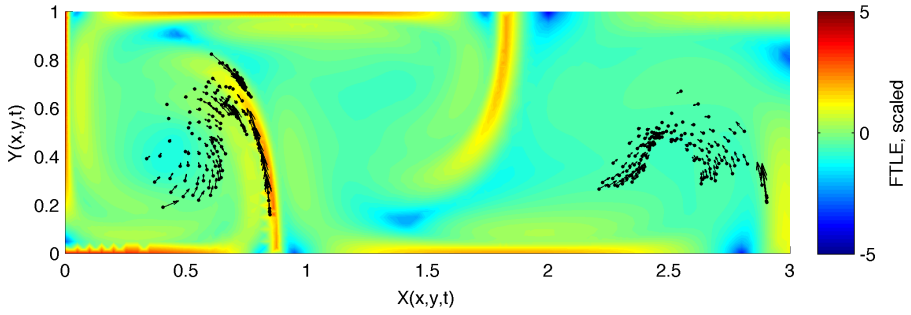


(b) $t = 2$.

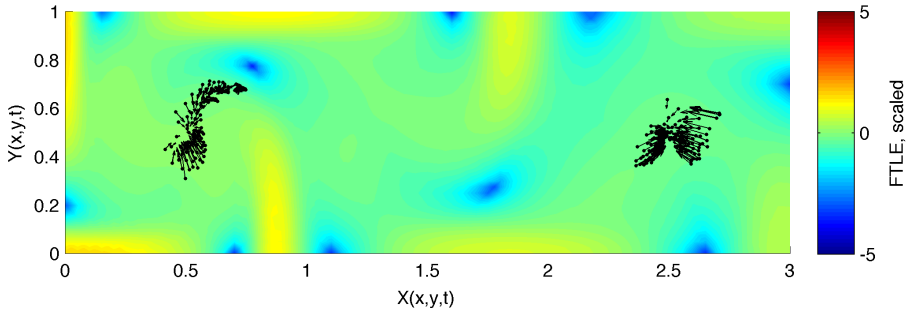


(c) $t = 4$.

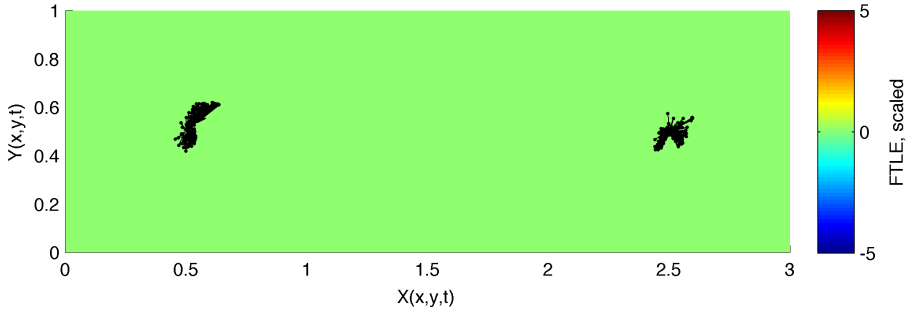
Figure 4.14: Initial time slices of the finite time Lyapunov exponent field described in Section 4.6 (color) and optimal trajectories (black markers and arrows) for the time-varying gyre flow. The FTLE field is defined from the induced norm of the Jacobian of the flow map $\mathbf{X_x}$, in other words the maximum speed of the transformed control $\mathbf{X_x u}$.



(a) $t = 6$.

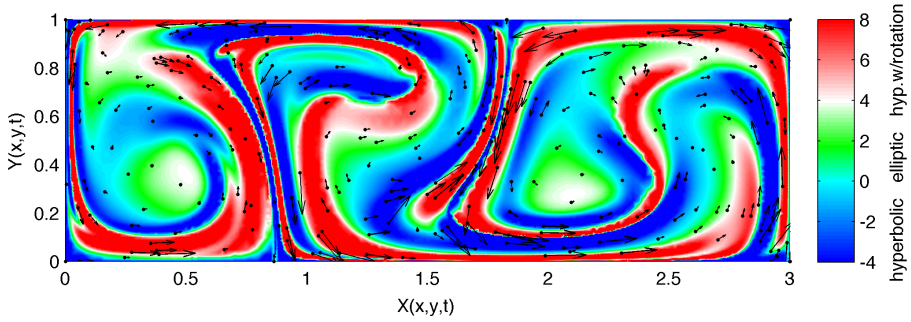


(b) $t = 8$.

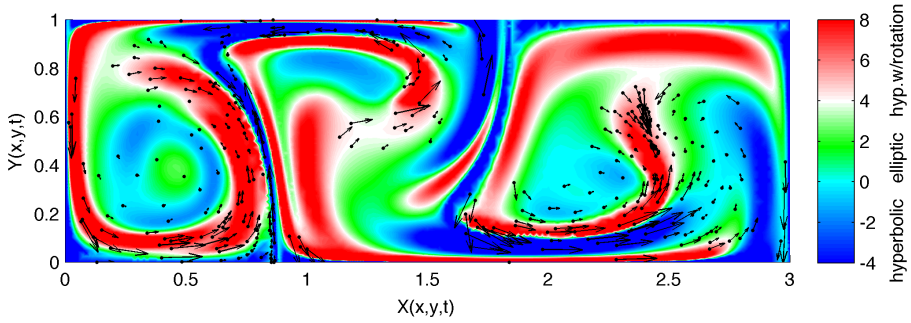


(c) $t = 10$.

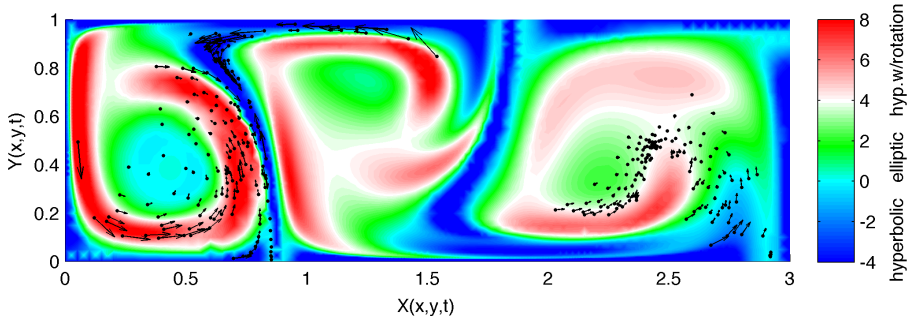
Figure 4.15: Final time slices of the finite time Lyapunov exponent field described in Section 4.6 (color) and optimal trajectories (black markers and arrows) for the time-varying gyre flow.



(a) $t = 0$.

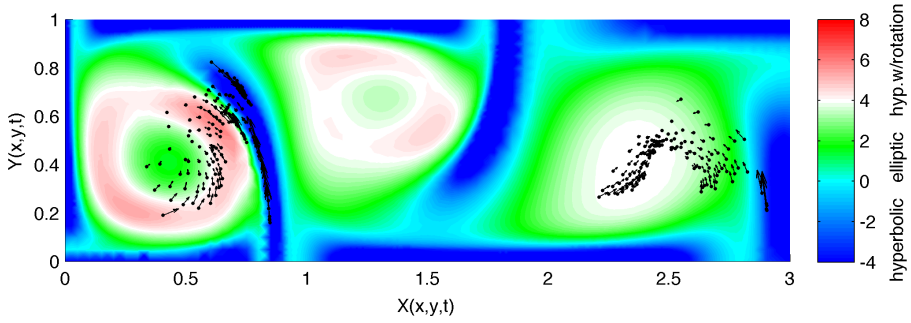


(b) $t = 2$.

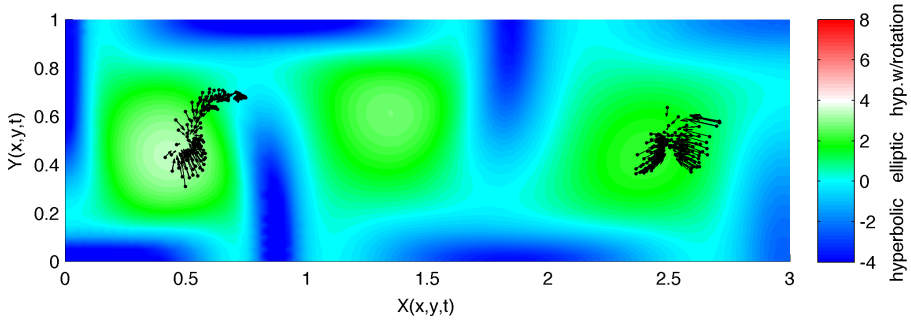


(c) $t = 4$.

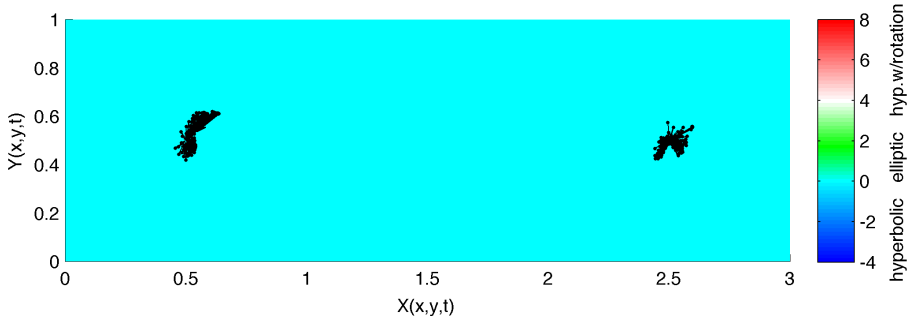
Figure 4.16: Initial time slices of the mesohyperbolicity field described in Section 4.6 (color) and optimal trajectories (black markers and arrows) for the time-varying gyre flow. In addition to hyperbolicity (stretching), the mesohyperbolicity field also distinguishes the ellipticity (rotation) of $\mathbf{u}$ by $\mathbf{X_x}$, unlike the FTLE field.



(a) $t = 6$.



(b) $t = 8$.



(c) $t = 10$.

Figure 4.17: Final time slices of the finite time Lyapunov exponent field described in Section 4.6 (color) and optimal trajectories (black markers and arrows) for the time-varying gyre flow.

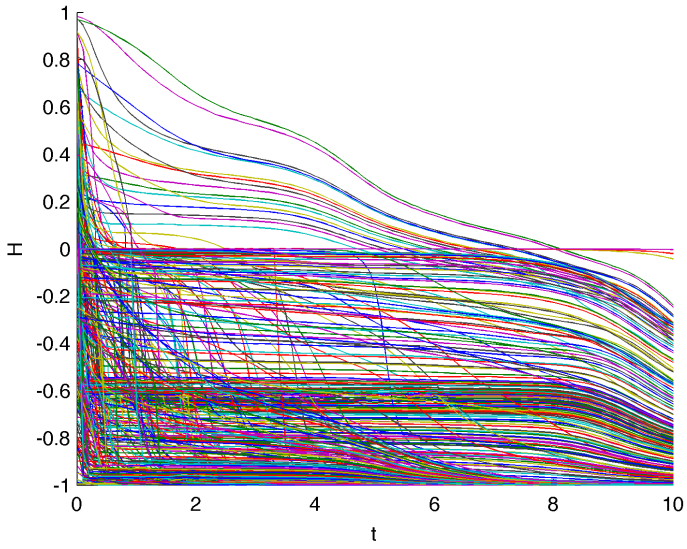


Figure 4.18: Pulled back end cost $H(x,t)$ along suboptimal trajectories for the time-varying gyre example. Compare to H along optimal trajectories of Figure 3.39.

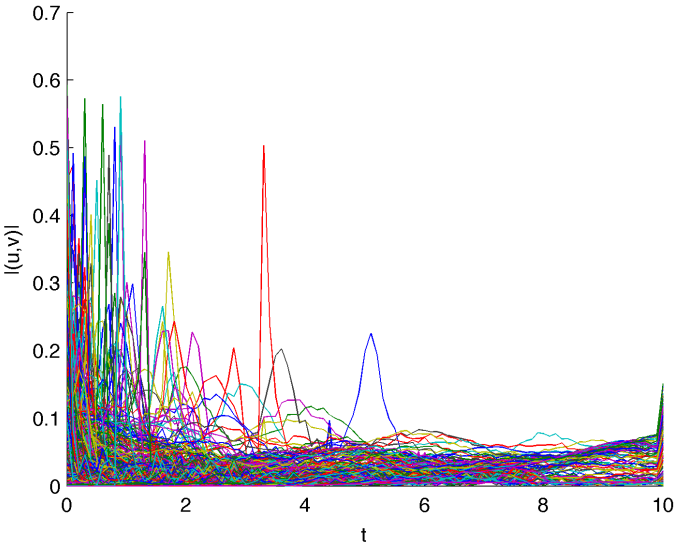


Figure 4.19: Size of control (u,v) along suboptimal trajectories for the time-varying gyre example. The signals are even less smooth than those of the optimal control signals of Figure 3.38, and the constraint $|u^2+v^2| = s = 1$ in much of the domain. This is largely due to the sensitive nature of the pulled back end cost H on which the suboptimal control is based.

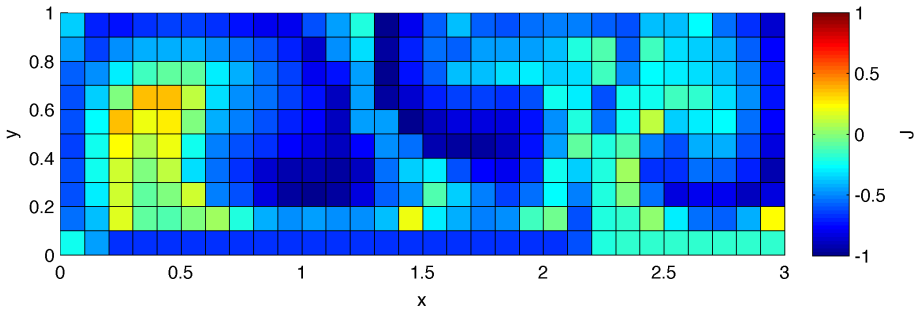
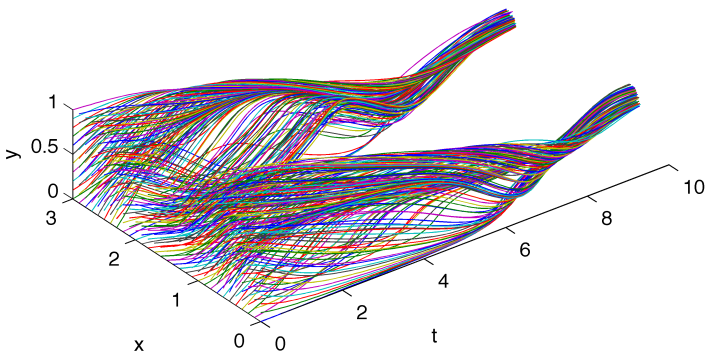
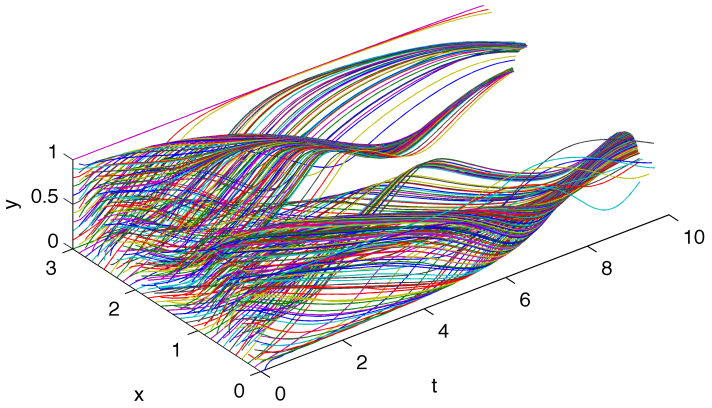


Figure 4.20: Total cost $J(x, y, 0)$ of the suboptimal trajectories for the time-varying gyre example. The cost is much higher than the optimal cost-go-go $V(x, y, 0)$ shown in Figure 3.29(a), with a maximum of about 0.4 versus ≈ -0.4. The difference is greater for instance in the "lobe" of the fluid near $(x, y) \approx (0.3, 0.5)$ encompassed by a ridge in the pulled back end cost H in Figure 3.35(a).

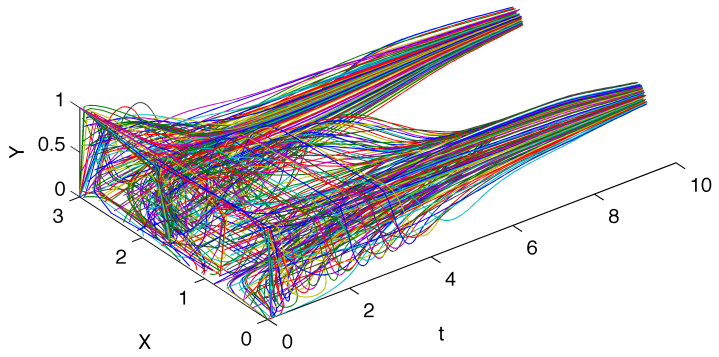


(a) Optimal trajectories

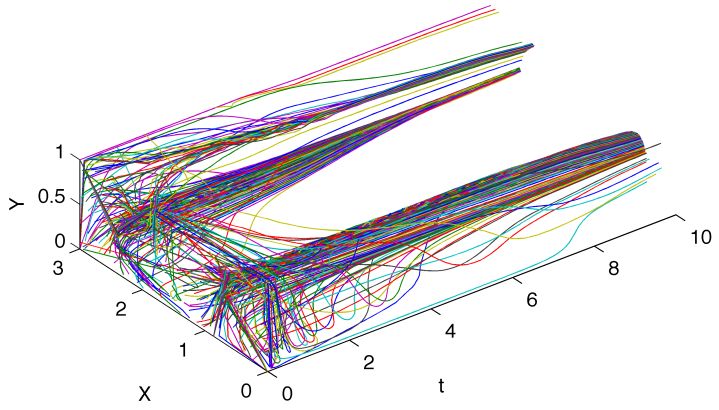


(b) Subptimal trajectories

Figure 4.21: (a) Optimal and (b) suboptimal trajectories $(x(t), y(t))$ for time-varying gyre flow field. Recall the sinusoidal end cost $h(x, y)$ has minima at $(x, y) = (0.5, 0.5)$ and $(2.5, 0.5)$.



(a) Tranformed optimal trajectories



(b) Transformed subptimal trajectories

Figure 4.22: Transformed (a) optimal and (b) suboptimal trajectories $(X(t), Y(t))$ for time-varying gyre flow field. The optimal trajectories expends more energy earlier, where the flow map is more sensitive to the input. In particular it's easier to see where $(\dot{X}, \dot{Y}) \approx 0$ and thus $(u, v) \approx 0$ here than in the original coordinates. On the other hand the suboptimal trajectories converge too quickly, consuming more energy but still not arriving as close to the minima of the end cost; their greedy nature keeps them from fully exploiting the sensitivity of the flow.

4.7 Summary and Outlook

This chapter explored the relationship between fixed final time minimum energy trajectories in a flow field and the spatial structure of that flow field, as captured by the so-called flow map $\mathbf{X}$, its Jacobian $\mathbf{X}_\mathbf{x}$, and what we call the pulled-back end cost function H. First, a simple coordinate transformation was proposed for the HJB equation based on the flow map that isolates the dynamics of the vehicle position relative to the flow. Second, the pulled back end cost H was shown to be a reasonable first approximation of the cost-to-go function V—the solution of the associated time-varying HJB equation—at least for the case where the energy weight $W = 0$ and the vehicle speed s is small. Third, three "local Lagrangian control" laws were proposed and briefly tested against the optimal control in 1D, and one of them was demonstrated in more detail in 1D and 2D. The suboptimal control performed generally well, but was significantly different from the optimal control. The most notable difference (not surprisingly) was the inability of the suboptimal control to descend H non-monotonically, which in some cases increased the trajectory cost by as much as a factor of 3.

The most obvious area of future work is to make the proposed suboptimal control law(s) more efficient to compute numerically. The current algorithm works by first computing H and thus $\mathbf{X}$ globally. As such it can take longer to compute the pulled back end cost H used in the suboptimal control than to compute the optimal cost-to-go V if t_f is very large. The true advantage of the proposed

suboptimal control laws would be the ability to compute it locally. The challenge is to avoid the chattering associated with the gradient $H_{\mathbf{x}}$ computed in the highly sensitive areas of the flow that tend to be utilized by the control.

A second possibility for future work is to solve the HJB equation directly in the transformed state space, perhaps with a variant of the Ordered Upwind Methods of Section 1.2.3 for monotonically advancing fronts. In the transformed space, the speed of propagation of the wavefronts defined by level sets of the (transformed) cost-to-go $V'(\mathbf{X}, t)$ is non-homogeneous (spatially dependent) and anisotropic (directionally dependent), as well as time-dependent, but it does not depend directly on any external flow, and thus the wavefronts advance monotonically outward and V' decreases monotonically in backward time.

The most straightforward direction of future work would be to test the efficiency and accuracy of the present triangular adaptive grid algorithm purely for the purpose of computing the flow map and related quantities such as the FTLE and mesohyperbolicity fields. Though not emphasized, the present adaptive grid algorithm appears to offer a simple and effective alternative to the present state of the art, namely the adaptive grid methods of [41, 45, 46]. The first step in this direction would be to implement the triangular adaptive grid algorithm of [47], which is functionally equivalent to but slightly simpler than the present algorithm. The second step would be to implement a flow map composition algorithm like

that of [40]. The third step would be to combine an adaptive grid algorithm and a flow map composition algorithm, which to our knowledge has never been done.

Chapter 5

Minimum Energy Feedback Control: a Backward-in-Time Lagrangian Approach

We now present results for one final algorithm. Like in the semi-Lagrangian approach of Chapter 3, we solve the fixed final time minimum energy feedback control problem in terms of the optimal cost-to-go V, backward in time. The present algorithm also relies heavily on adaptive Eulerian grids but is primarily Lagrangian. The results are preliminary and limited to 1D; they are considered a proof of concept. Nevertheless, we use vector notation where possible and the term "simplex" to refer to either a 1D grid "edge" or a 2D grid "triangle".

5.1 The Trouble with the Semi-Lagrangian Approach

The semi-Lagrangian approach is to compute V at a given point $(\mathbf{x}, t)$ from the subsequent time slice $V(\cdot, t + \Delta t)$ by first computing (an approximation of) the characteristic curve of the PDE *forward* from $(\mathbf{x}, t)$ to $t + \Delta t$. Fortunately, the optimal control $\mathbf{u}$ and thus the velocity $\mathbf{v} + \mathbf{u}$ of a characteristic of the HJB PDE can be expressed analytically. Unfortunately, they depend on the gradient $V_{\mathbf{x}}$ (or the costate $\mathbf{p}$). If one takes into account the familiar Euler Lagrange ODEs, the result is a TPBVP for the characteristic state trajectory $\mathbf{x}^*$ and the associated costate trajectory $\mathbf{p}^*$, with boundary conditions $\mathbf{x}^*(t) = \mathbf{x}$ and $\mathbf{p}^*(t + \Delta t) = V_{\mathbf{x}}(\mathbf{x}^*(t + \Delta t), t + \Delta t)$. Note that this small TPBVP is analogous to the larger TPBVP that describes the overall optimal trajectory problem (for a particular initial point $(\mathbf{x}_0, t_0)$), described in Section 1.1.1.

The semi-Lagrangian method of Chapter 3 assumes linear approximation of $\mathbf{x}^*$ and $V_{\mathbf{x}}(\cdot, t + \Delta t)$ to obtain $\mathbf{u}$ "exactly" via constrained quadratic optimization, sacrificing accuracy to avoid the iterative schemes entailed by more general optimization problems (like in the higher-order semi-Lagrangian method of [13]). Iterative solution of the above TPBVP would be better. But it would still add unnecessary computational complexity.

The bottom line is this: by considering each point $(\mathbf{x}, t)$ *separately*, one fails to exploit the fact that $V_{\mathbf{x}}$ and thus $\mathbf{p}$ and $\mathbf{u}$ are known a priori and *globally* at time $t + \Delta t$. This is where the Lagrangian approach comes in. Note that we do not attempt an Eulerian explicit time-stepping (for instance with an upwind Godunov shock-capturing scheme) because of the time step restriction associated with very high resolution adaptive grids.

5.2 The Lagrangian Approach

The present Lagrangian approach is to compute the characteristics globally, *backward* in time, as (projections of) solutions of the Euler Lagrange ODEs, and in effect to solve the small TPBVP described above for all points $\mathbf{x}$ at once (and with Δt generally larger). In other words, the present method is a backward-in-time extremal field method and a method of characteristics for the dynamic HJB equation, as described in Section 1.1.2, but for one interval $[t, t + \Delta t]$ at a time, with $V(\cdot, t + \Delta t)$ playing the role of the end cost h.

In addition to the problem parameters s, W, t_f, and $\mathbf{x}_0$ and the algorithm parameters $\Delta V_{\max}$ and n_t already defined for the semi-Lagrangian algorithm of Section 3.3, the present algorithm requires just two additional parameters:

- n_t^L, the number of intermediate time steps at which the Lagrangian extremal data is computed in each of the n_t intervals $[t, t + \Delta t]$ (where $\Delta t := t_f / n_t$) and

- $\Delta\mathbf{p}_{\max}$, the error tolerance dictating the refinement of the adaptive grid of extremals.

Recall the adaptive grid refinement is based on linear error estimates computed at the midpoints of grid edges (whether in 1D or 2D). In the grid of extremals, this means an entire extremal for each error estimate. The error $\Delta\mathbf{p}$ here is based on the terminal costate $\mathbf{p}(t)$ of the backward computed extremal. More generally, the error estimate could involve a combination of the errors in the state, costate, and (suboptimal) cost-to-go. However, we found the costate sufficient. Like the semi-Lagrangian algorithm, the Lagrangian algorithm has two parts: the backward-in-time integration of V and the forward-in-time extraction of optimal trajectories for individual initial points $(\mathbf{x}_0, t_0)$.

The end product of the backward-in-time integration is:

- V and $V_{\mathbf{x}}$ in a sequence of $n_t + 1$ Eulerian "slice" objects, and
- J, $\mathbf{p}$, and $\mathbf{x}$ (along the extremals of length $n_t^L + 1$ from $t + \Delta t$ to t) on a sequence of n_t Lagrangian "field" objects.

Recall that the costate $\mathbf{p}$ is equal to $V_{\mathbf{x}}$ along extremals that turn out to be globally optimal trajectories. The (suboptimal) cost-to-go J is defined like in (3.1) but with the integral reversed in time. Each "slice" or "field" object contains a unique "grid" object. The backward-in-time integration begins by computing the slice with $V(\mathbf{x}, t_f) = h(\mathbf{x})$ and $V_{\mathbf{x}}(\mathbf{x}, t_f) = h_{\mathbf{x}}(\mathbf{x})$, analytically. Then for each of the n_t

intervals $[t, t+\Delta t]$ it computes a new field from the slice at $t+\Delta t$, and a new slice from that field. To be clear, the field grid lies at $t+\Delta t$ but doesn't coincide with the slice grid.

In general one can compute the extremal trajectories $(\mathbf{x}, \mathbf{p}, J)$ using any higher-order ODE solver desired; this is one of the advantages of the Lagrangian approach (and of the semi-Lagrangian approach, if not attempting an exact point-wise optimization). For simplicity we use the explicit Euler method. For convenience, we restate the ODEs for J (from (3.1)) and for $\mathbf{x}$ and $\mathbf{p}$ (from the general Euler Lagrange equations (1.11)) for the present case of minimum energy control:

$$\dot{J} = -W\mathbf{u}\cdot\mathbf{u};$$

$$\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}, t) + \mathbf{u};$$

$$\dot{\mathbf{p}} = -\mathbf{v}_{\mathbf{x}}(\mathbf{x}, t)\mathbf{p}.$$

Recall that $\mathbf{u}$ here is given by $\mathbf{u} = -\frac{\mathbf{p}}{2W}$ if $|\mathbf{p}| \leq 2Ws$ or $\mathbf{u} = -s\frac{\mathbf{p}}{|\mathbf{p}|}$ otherwise.

For a given field grid point $(\chi, t+\Delta t)$, the backward extremal is initialized with $\mathbf{x}(t+\Delta t) = \chi$ and with $J(t+\Delta t)$ and $\mathbf{p}(t+\Delta t)$ interpolated from $V(\cdot, t+\Delta t)$ and $V_{\mathbf{x}}(\cdot, t+\Delta t)$. For a given slice grid point (χ, t), $V(\chi, t)$ and $V_{\mathbf{x}}(\chi, t)$ are interpolated from $J(t)$ and $\mathbf{p}(t)$ for which $\mathbf{x}(t)$ is in the vicinity of χ—more specifically, from a field simplex (e.g. 1D edge or 2D triangle) that contains χ after being evolved from $t+\Delta t$ to t. In the present 1D implementation we use cubic interpolation for J and V and the corresponding quadratic interpolation for $\mathbf{p}$ and $V_{\mathbf{x}}$.

The problem here is that the simplexes of the Lagrangian field grid generally overlap one another and thus $J(t)$ is multi-valued. Hence computing $V(\chi, t)$ requires not only interpolating J for a given simplex but directly minimizing J over all simplexes containing χ. The effect is to simultaneously "remesh" and "trim", as in the forward-in-time minimum time extremal field algorithm of [24], but much more efficiently, since the tracked object is parameterized by points $\mathbf{x}$ in a hyper-rectangle instead of points on a curved front.

The crux of this problem is to obtain the lists of the field grid simplexes containing slice grid points χ. For a uniform slice grid, this can be done in a single loop through all the field simplexes, using the $O(1)$ "floor" and "ceiling" functions. Even for an adaptive slice grid, if the N slice grid points are chosen beforehand, one can at least find all the relevant slice grid points for a particular field simplex in $O(\log N)$ time via a simple binary search. But our grid points are generated adaptively based on V.

In the present 1D implementation we address this difficulty by maintaining lists of field simplexes (edges) not only for the slice grid points but also for the slice grid edges. Recall that we compute V (and V_x) at the midpoints of the field grid edges ahead of time—upon bisection of the parent edge—for the purpose of the error estimates ΔV. In addition, each such bisection then empties the list of the parent edge, moving each field simplex into one or both of the lists of the child slice grid edges.

This process of continually emptying old lists and populating new lists adds significant overhead. In particular, the memory allocation alone accounts for as much as 80-90% of the computation time. This made the overall algorithm slower than the semi-Lagrangian algorithm for a given number of grid points (though more accurate). The actual computational complexity has yet to be investigated very thoroughly.

5.3 1D Results: a Proof of Concept

The above Lagrangian algorithm was applied to the $W = 1$ example of Section 3.4.2, but with $\Delta V_{\max}$=1e-4 (instead of 1e-5), $n_t = 10$ (instead of 100), $n_t^L = 100$, and $\Delta p_{\max}$=1e-2. The key results are shown in Figures 5.1-5.4. Color plots of the cost-to-go $V(x,t)$, optimal control law $u(x,t)$, and pulled back end cost $H(x,t)$ are not included here, since they are approximated well enough by the semi-Lagrangian results of Figures 3.10, 3.12, and 3.14 respectively. Overall, the accuracy was better than for the semi-Lagrangian results, even with fewer (slice) grid points (about 500 at $t = 0$ instead of 1000); given the much higher-order interpolation schemes, this is not surprising. In addition, the algorithm was also expected to be faster, and for a given grid size, the speed was on the order of two times slower. However, no rigorous timings have been performed, and both algorithms ran in under a minute even for smaller tolerances and grid sizes on the order of 10,000 to 20,000 (slice grid) vertices.

Figure 5.1 shows the time slices of the cost-to-go V (black) and the multi-valued cost-to-go J (green) at $t = t_f = 2$, $t = \frac{t_f}{2} = 1$, and $t = 0$. As previously described, V is defined on the adaptive grid of a "slice" object while J is defined on the adaptive grid of the "field" object. It is these (green) surfaces J from which the (black) surfaces V are computed, via the all-in-one interpolation/minimization i.e. remeshing/trimming procedure. For each time slice (except $t = t_f$) the latter grid is an Eulerian grid in its native time slice at $t + \Delta t$ but can be thought of as a Lagrangian grid at time t. The resulting "swallowtails" are analogous for instance to the self-intersections of the minimum time extremal front of Figure 2.6(b).

The pair of additional cusps in each swallowtail is due solely to the projection of the Lagrangian extremal data from (x, p, J) space into (x, J) space. The size of the swallowtails in (x, J) space is proportional to the size $\Delta t = \frac{t_f}{n_t} = 0.2$ of the outer time steps. However, regardless of Δt, their size in (x, p, J) space is bounded below by the size of the jump discontinuity of V_x in the slice grid from which the initial values $J(t + \Delta t)$ and $p(t + \Delta t)$ are interpolated; this makes the optimal size of the field grids (or at least the size that we settled on) on the order of twice as large as the slice grids, which adds significant overhead but is not perceived as a major limitation of the algorithm.

Figure 5.2 shows the level of refinement L along the adaptive (slice) grid at $t = 0$. The corresponding field grid is similar and not shown. Though it does not seem to make much of a practical difference (e.g. in the efficiency of the algorithm),

the level L is noticeably "smoother" than in the semi-Lagrangian result. This is thought to be due to the higher-order interpolation.

Figure 5.3 makes the idea of the algorithm more clear. It shows the extracted optimal trajectories (black) and the backward extremal trajectories (green). The x values of optimal trajectories are indistinguishable from the semi-Lagrangian case. The extremals begin intersecting one another in backward time beginning at around $t = 1.8$, in other words where the shock seen in $V(x,t)$ and $u(x,t)$ dies out. These intersections of nearby extremals in (x,t) space are associated with the swallowtails—the self-intersections of the time slices of the extremal field—in (x,J) space. Conversely, in the remainder of the domain the extremals splay out from one another in backward time. Though hardly mentioned in this dissertation (since we do not include results where the end cost $V(\mathbf{x},t_f) = h(\mathbf{x})$ is non-smooth), this splaying out is similar to (but not quite as extreme as) that observed in the phenomena known as *rarefactions*, where characteristics actually merge (in this case in forward time, at sharp local minima of $V(\mathbf{x},t_f) = h(\mathbf{x})$).

Figure 5.4 provides the best evidence of the accuracy of the Lagrangian algorithm, in terms of the optimal control input signals $u(t)$. These signals are much smoother than the semi-Lagrangian ones. This was not as much the case if we replaced Δp with Δx or ΔJ in the adaptive grid refinement criterion. A more thorough analysis, empirical and theoretical, of the accuracy and stability that results from different combinations of these criteria would be of interest, and may

also be tied to the inefficiency of the current implementation. Also, note that the signals shown here are defined at $n_t n_t^L = 1000$ grid points instead of $n_t = 100$ as in the semi-Lagrangian solution. The ability to increase the resolution in this way—via n_t^L instead of n_t is one of the benefits of the present Lagrangian approach (or of a semi-Lagrangian approach that computes and stores forward-in-time extremal trajectories e.g. via the shooting method for the TPBVP described in Section 1.1.1); however, the signals $u(t)$ are much better even with $n_t^L = 10$.

In addition to the computational complexity of moving around lists of indices of Lagrangian grid simplexes described in Section 5.2, the algorithm did seem to have some issues with stability. The issues tended to occur for grid sizes upwards of 20,000 to 30,000, for different combinations of refinement criteria Δp, Δx and ΔJ, and for n_t too small. (In fact, we would have used $n_t = 100$ instead of $n_t = 10$, but that would require close-up plots of the swallowtails and the intersecting extremal trajectories in order to convey the idea of the algorithm). Usually, the cause is that a swallowtail is under-resolved in such a way that the resulting slice of V develops a discontinuity (which makes the level of refinement L blow up, since, recall, we don't bound L explicitly). This is associated with both $\Delta p_{\max}$ and $\Delta V_{\max}$, the latter of which determines how precisely to initialize extremal trajectories near a shock (cusp of V). The solution of these issues may involve some finite differencing in the computation of $V_{\mathbf{x}}$, perhaps making use of the essentially non-oscillatory (ENO) schemes commonly used in Level Set Methods. Basic central differencing

was tried but did not seem to work. Additionally, it may also be necessary to introduce numerical diffusion by explicitly bounding L for slice grids. A more thorough investigation of these issues is of interest but outside the scope of this dissertation.

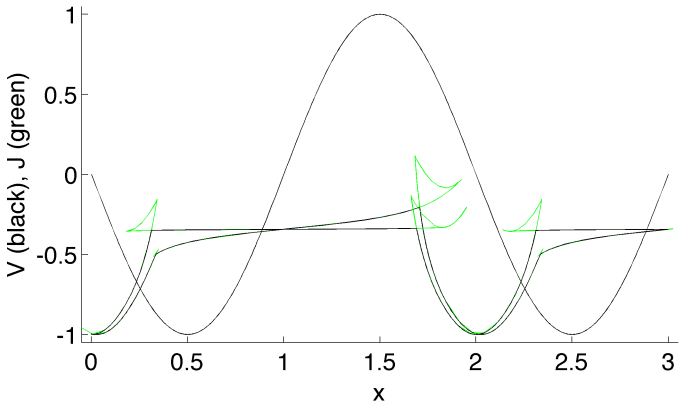


Figure 5.1: Three of the eleven recorded slices of the optimal cost-to-go V (black) and the suboptimal cost-to-go J (green), at times $t_f = 2$, $\frac{t_f}{2} = 1$, and 0. V is obtained by direct minimization of J (simultaneously remeshing and trimming the extremal field). The green "swallowtails" result from the adaptive initialization of extremals precisely at the shock (cusp) of the subsequent slice of V, which compares well to the semi-Lagrangian result of Figure 3.8(a).

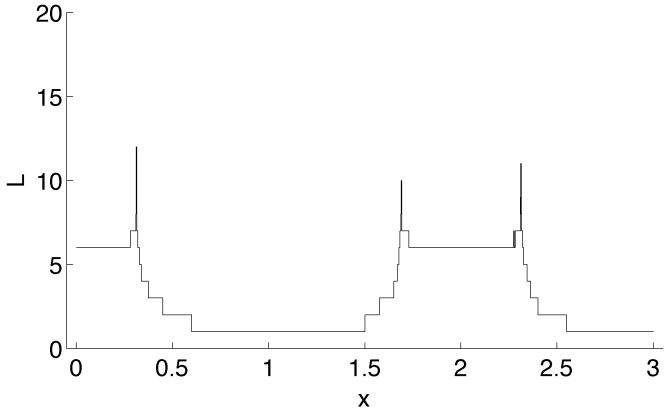


Figure 5.2: Level of refinement L of the adaptive grid for the time 0 slice of the optimal cost-to-go V. L compares well to the semi-Lagrangian result of Figure 3.9(a), for which the total number of adaptive grid points is on the order of twice as large.

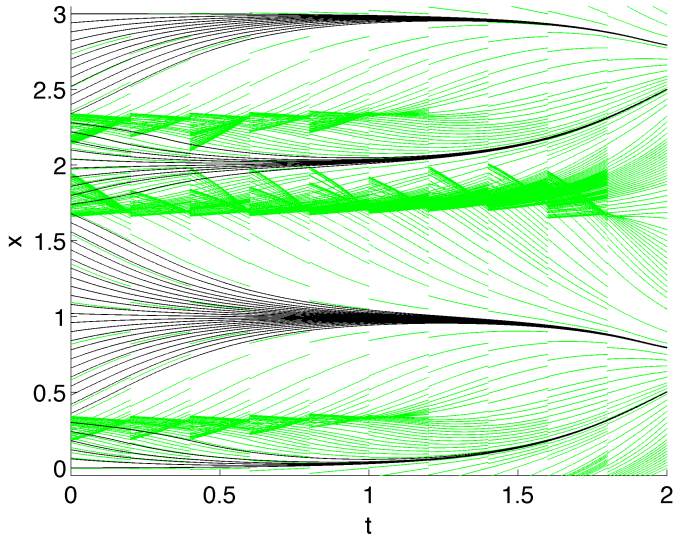


Figure 5.3: Extracted optimal trajectories x (black) and the ten backward-in-time fields of extremal trajectories (green). The optimal trajectories here are indistinguishable from the semi-Lagrangian result of Figure 3.2(a), but the control signals of Figure 5.4 are significantly better.

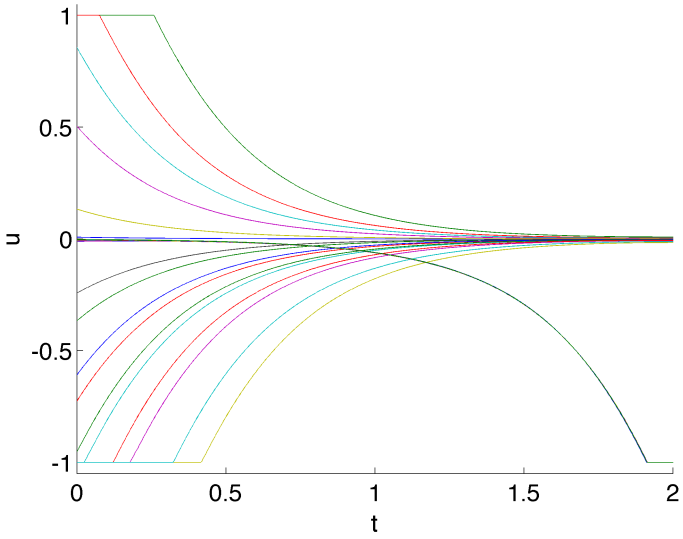


Figure 5.4: The control input u along the optimal trajectories of Figure 5.3. The results are much smoother than in the semi-Lagrangian result of Figure 3.5(a), with fewer grid points at each time step.

Chapter 6

Summary and Outlook

The purpose of this chapter is to restate the key results of Chapters 2-5 and to make recommendations for future work. These items fall into four categories: minimum time control (Chapter 2), minimum energy control (Chapters 3 and 5), more general optimal control problems, and feedback control and the flow map (Chapter 4).

6.1 Minimum Time Control

In Chapter 2 and in [24], we presented a forward-in-time Lagrangian front-tracking method for open-loop minimum time trajectories. The tracked "reachability front" evolves along the boundary of the reachable set in space-time, but the Lagrangian marker particles evolve in a higher-dimensional state-costate space. The accuracy and efficiency of the method in the presence of sensitivity and shocks are enabled by remeshing and trimming procedures, respectively. In addition to shocks (cusps in the boundary of the reachable set where optimal trajectories ter-

minate), the method sharply captures discontinuities in the "arrival time function" where the flow overpowers the vehicle. For the time-invariant case, this ability to handle strong flows is a clear advantage of the proposed Lagrangian method over the semi-Lagrangian and Eulerian Ordered Upwind Methods. For the time-varying case, the method works exactly the same as in the time-invariant case; in contrast, Ordered Upwind Methods must be replaced by a one-higher-dimensional Level Set Method like that of [25]. Finally, Chapter 2 and [24] also validate the proposed forward Lagrangian method against the prior backward Lagrangian method of [23], which in fact solves the underlying dynamic HJB equation.

Though the main results were for a numerically-defined 2D flow from a high-resolution 3D model of the Adriatic Sea, the method could be extended to the case of a depth-varying (but still 2D) time-varying ocean flow assuming the depth of the glider is given as a function of time. Using the formalism $(u, v)(x, y, t) := (\tilde{u}, \tilde{v})(x, y, z(t), t)$, this is expected to be straightforward, and mostly a matter of interpolating the flow $(\tilde{u}, \tilde{v})$ from a 3D grid which often uses isopycnal coordinates instead of simple z values.

An easy way of improving the accuracy of the forward Lagrangian algorithm is to use the a "costate-preserving" cubic interpolation scheme, as mentioned briefly in Section 2.3. Specifically, if interpolating between adjacent marker particles (x_1, y_1, Θ_1) and (x_2, y_2, Θ_2) then the curve of interpolated (x, y) values should be consistent with Θ_1 and Θ_2—the known normal angles. This would be analogous

to the preservation of the consistency between the costate p and the gradient V_x in the preliminary 1D backward Lagrangian method for minimum energy control demonstrated in Chapter 5.

The main drawback of the forward Lagrangian method (and indeed of all Lagrangian methods discussed in this dissertation) is the trimming. As shown, trimming prevents exponential growth over time in the number of marker particles, but dramatically complicates the overall algorithm. Though a better trimming procedure for the present 1D reachability front is certainly feasible, this is not thought worthwhile for more general minimum time problems in higher dimensions. In that case, we recommend a Level Set Method like that of [25], but made more efficient using some kind of adaptive grid, and made more accurate using a standard reinitialization procedure, among other things.

Alternatively, using an adaptive (but less focussed) grid of the same dimension, one could in fact solve the HJB equation directly, as in [23], but on a structured grid in a closed domain, using a backward time-stepping scheme (semi-Lagrangian like that of Chapter 3, Lagrangian like that of Chapter 5, or Eulerian). In particular, for trajectories not controllable to the target $\mathbf{x}_f$ within the time limit $T = t_f - t_0$ one could add the end cost $h(\mathbf{x}(t_f)) = |\mathbf{x}(t_f) - \mathbf{x}_f|^2$ (instead of ∞), defining the terminal hypersurface $\mathbf{m}(\mathbf{x}, t) = 0$ of (1.3) to include *both* the line $\mathbf{x} - \mathbf{x}_f = 0$ used in the "minimum time" problem of Chapter 2 and the plane $t - t_f = 0$ used in the "minimum energy" problem of Chapter 3.

6.2 Minimum Energy Control

In Chapter 3 we presented a backward-in-time semi-Lagrangian method for a minimum energy feedback control problem. The method effectively transforms the optimal control problem into a point-wise optimization problem. Unlike prior semi-Lagrangian methods for similar problems, the proposed method captures shocks (discontinuities in the gradient of the cost-to-go function) very precisely using an adaptive triangular grid, and avoids less accurate iterative point-wise optimization by restricting to first-order accurate spatio-temporal discretizations. The main solution example was for an analytically-defined time-varying three-gyre flow field. The adaptive grid performs very well, and the method was sufficient to characterize the minimum energy control for the present problem, but is not as accurate as desired, since it does not scale well with the number of grid points or the number of time steps. This is thought to be inherent in the semi-Lagrangian approach, which does not take advantage of the global structure of the problem; whereas the optimal control at time t requires a point-wise optimization (and thus a loop through all the simplexes of the adaptive grid reachable in one time step of length Δt), the optimal control at time $t + \Delta t$ is known a priori, in terms of the gradient of the cost-to-go function.

This suggests either an Eulerian or a Lagrangian backward-in-time method. An Eulerian method was not pursued in this dissertation because of the strict time step restriction associated with high resolution adaptive grids. However, it's

possible that the present adaptive grid is resolving shocks more precisely than necessary, and an Eulerian (for instance an upwind Godunov) scheme for this problem would be of great interest. Another alternative worth exploring is to implement an iterative point-wise optimization scheme (preferably more sophisticated than that of [13]) and higher-order semi-Lagrangian discretizations.

In Chapter 5 we presented a 1D proof of concept of a backward-in-time Lagrangian method for the same minimum energy problem. The method computes the cost-to-go at a given time slice $V(\cdot, t)$ from $V(\cdot, t + \Delta t)$ by (a) computing an backward-in-time extremal field from $V(\cdot, t + \Delta t)$ and (b) computing $V(\cdot, t)$ from the extremal field via interpolation and direct minimization. The crux of the algorithm is to efficiently find the relevant extremal simplexes for a given grid point of $V(\cdot, t)$. In the present implementation this is done less efficiently than hoped, and accounts for up to $\approx 90\%$ of the computational time. Moreover, there were some issues with stability; these suggest the introduction of numerical diffusion via finite differences and/or an explicit bound on the resolution of the adaptive grid.

6.3 More General Optimal Control

There are many generalizations, some tested, others not, that ought to be considered in future implementations of the present algorithms.

Perhaps the most obvious is the case of $W = 0$ in the minimum energy problem—i.e. the "pure end cost" problem. If the minima of the end cost h are not reachable, the optimal control is like minimum time control. Otherwise, the solution V is flat, and the optimal control is non-unique. The best foreseen way to remedy this is to explicitly define the passive particle trajectories terminating at the minima of h as subsets of the terminal hypersurface $\mathbf{m} = 0$ (similar to in the last paragraph of Section 6.1, and add the minimum time term 1 to the minimum energy term $W\mathbf{u} \cdot \mathbf{u}$ in the cost functional. In this sense, the generalization $W = 0$ goes hand-in-hand with generalizing the types of terminal hyper surfaces $\mathbf{m} = 0$ (i.e. boundary conditions) considered.

There are several cases that were tested but not included in this dissertation. For instance, generally h may be non-smooth. With the addition of an explicit bound on the grid resolution $L_{\max}$, h may be discontinuous, as mentioned in Chapter 3. And with the addition of a few well-placed if-statements, the h may even be infinite, with the adaptive grid resolving the jump to infinity as any other discontinuity; more generally, the cost-to-go V may be jump to infinity in backward time if $|\mathbf{v}| > s$ i.e. the flow overpowers the vehicle at the boundary—a case not discussed elsewhere in this dissertation.

Other cases of interest include the addition of a position penalty term $c(\mathbf{x}(t))$ and more general control systems such as under-actuated systems.

6.4 Feedback Control and the Flow Map

Chapter 4 was centered around the use of the fixed final time flow map for (a) a different perspective on the minimum energy trajectories of Chapter 3 and (b) a class of simple suboptimal control laws. Specifically, the fixed final time flow map $\mathbf{X}(\mathbf{x}, t)$ yields transformed dynamics of the form $\dot{\mathbf{X}} = \mathbf{X}_{\mathbf{x}}\mathbf{u}$ from the original dynamics $\dot{\mathbf{x}} = \mathbf{v} + \mathbf{u}$, and the suboptimal control laws are of the form $\mathbf{u} = -\tilde{s}\frac{H_{\mathbf{x}}}{|H_{\mathbf{x}}|}$, where $H(\mathbf{x}, t) := h(\mathbf{X}(\mathbf{x}, t))$ is the pulled back end cost. In the transformed coordinates, the system can be controlled in any direction, but with a directionally dependent speed that may be arbitrarily small. One such suboptimal control law was compared to the optimal for the 2D time-varying three-gyre example of Chapter 3. The main advantage of the optimal control is that it descends H non-monotonically. The main advantage of the suboptimal control is that the flow map $H_{\mathbf{x}}$ can be defined locally; however, this advantage was not fully demonstrated, due to the sensitive nature of $H_{\mathbf{x}}$, which leads to chattering behavior in trenches of H.

There are several possibilities for extending this work. The most directly practical topic of future work would be a numerical definition of the suboptimal control that is not so vulnerable to this chattering. Another area of interest is the possibility of solving the transformed HJB equation directly by somehow taking advantage of the monotonicity of the transformed cost- to-go $V'(\mathbf{X}, t)$ with respect to t. Yet another area of interest is to improve the state of the art in numerical

methods for efficient computation of the flow map and related quantities such as the finite time Lyapunov exponent and mesohyperbolicity fields. Toward this end, the triangular adaptive grid proposed in this dissertation and in [47] appears to offer a simple and effective alternative to the present state of the art. Of particular interest would be to combine this or another adaptive grid method with a flow map composition method like that of [40].

Chapter 7

Appendices

The three appendices are all for Chapter 2.

7.1 Derivation of the relaxed minimum time necessary condition

In this appendix we justify the use of the $\dot{\Theta}$ component of (2.4) as a relaxed necessary condition for a minimum time trajectory of (2.1), as well as the forward-in-time approach used here versus the backward-in-time approach used in [23].

Suppose $(x, y, \Theta) : [t_0, t_f] \to \mathrm{R}^2 \times \mathrm{S}^1$ is a minimum time trajectory of the system (2.1) from (x_0, y_0, t_0) to (x_f, y_f). Then there exists a co-state trajectory $(p, q) : [t_0, t_f] \to \mathrm{R}^2$ such that:

1. $\forall t \in [t_0, t_f]$,

$$\dot{p} = -\frac{\partial \mathcal{H}}{\partial x}(x, y, \Theta, p, q, t);$$

$$\dot{q} = -\frac{\partial \mathcal{H}}{\partial y}(x, y, \Theta, p, q, t),$$

(where $p = p(t), x = x(t)$, etc),

2. $\forall t \in [t_0, t_f]$,

$$\mathcal{H}(x, y, \Theta, p, q, t) = \min_{\tilde{\Theta} \in \mathrm{S}^1} \mathcal{H}(x, y, \tilde{\Theta}, p, q, t),$$

and

3.

$$\mathcal{H}(x_f, y_f, \Theta_f, p_f, q_f, t_f) = 0,$$

where $(p_f, q_f) = (p(t_f), q(t_f))$, and where $\mathcal{H}$ is the Hamiltonian function, in this case given by $\mathcal{H}(x, y, \Theta, p, q, t) := [u(x, y, t) + s\cos\Theta]p + [v(x, y, t) + s\sin\Theta]q + 1$. These are the well-known necessary conditions arising from the calculus of variations.

The two ODEs composing Condition 1 are the co-state equations. Computing the partial derivatives of $\mathcal{H}$ yield

$$\begin{aligned} \dot{p} &= -u_x p - v_x q; \\ \dot{q} &= -u_y p - v_y q, \end{aligned} \tag{7.1}$$

where $u_x = u_x(x, y, t)$, etc. are the partial derivatives of the flow field (u, v).

Condition 2 is the Pontryagin principle. It says that the input Θ must minimize the Hamiltonian point-wise. This boils down to minimizing the dot product of the co-state vector (p, q) and the control vector $(s\cos\Theta, s\sin\Theta)$, in which, recall, the speed $s > 0$ is fixed. It follows from Conditions 1 and 3 that $\forall t \in [t_0, t_f], (p, q) \neq 0$; otherwise, by Equation (7.1) i.e. Condition 1, $\forall t \in [t_0, t_f], (p, q) = (0, 0)$, which

implies $\mathcal{H}(x_f, y_f, \Theta_f, p_f, q_f, t_f) = 1$, which violates Condition 3. Thus $\forall t \in [t_0, t_f]$, the optimal input angle is well-defined and is given by

$$\Theta = \text{atan2}(-q, -p),$$

where the function atan2 is defined for $(x, y) \neq (0, 0)$ by

$$\text{atan2}(y, x) = \left\{ \begin{array}{ll} \arctan \frac{y}{x} & x > 0 \\ \arctan \frac{y}{x} + \pi & x < 0, y \geq 0 \\ \arctan \frac{y}{x} - \pi & x < 0, y < 0 \\ +\frac{\pi}{2} & x = 0, y > 0 \\ -\frac{\pi}{2} & x = 0, y < 0 \end{array} \right\}$$

In other words, the minimum time control is to steer the vehicle in the opposite direction of the co-state vector (p, q).

Moreover, there is a one-to-one mapping between co-state vectors $(p, q) \neq (0, 0)$ and polar coordinate pairs $(\Theta, r) \in \mathrm{S}^1 \times (0, \infty)$ given by

$$(\Theta, r) = (\text{atan2}(-q, -p), \sqrt{p^2 + q^2})$$

and

$$(p, q) = (-r \cos \Theta, -r \sin \Theta), \tag{7.2}$$

and thus the co-state equations can be transformed to this polar domain. Substituting these expressions for p and q into (7.1) and differentiating the left hand side yields the following linear system for $\dot{\Theta}$ and $\dot{r}$:

$$r \sin \Theta \dot{\Theta} - \dot{r} \cos \Theta = u_x r \cos \Theta + v_x r \sin \Theta;$$

$$-r \cos \Theta \dot{\Theta} - \dot{r} \sin \Theta = u_y r \cos \Theta + v_y r \sin \Theta.$$

Multiplying by $\sin\Theta$ and $-\cos\Theta$ respectively yields

$$r\sin^2\Theta\dot{\Theta} - \dot{r}\sin\Theta\cos\Theta = u_x r\sin\Theta\cos\Theta + v_x r\sin^2\Theta;$$

$$r\cos^2\Theta\dot{\Theta} + \dot{r}\sin\Theta\cos\Theta = -u_y r\cos^2\Theta - v_y r\sin\Theta\cos\Theta.$$

Summing and dividing through by r yields

$$\dot{\Theta} = -u_y\cos^2\Theta + (u_x - v_y)\sin\Theta\cos\Theta + v_x\sin^2\Theta, \tag{7.3}$$

i.e. the third component of Equation (2.4). Equations similar to (7.3) are derived in [7] and [61], for a minimum time example and a flame propagation model, respectively. Similarly, one obtains

$$\dot{r} = r\left[u_x\cos^2\Theta + (u_x + v_x)\sin\Theta\cos\Theta + v_y\sin^2\Theta\right].$$

The expressions for $\dot{x}$, $\dot{y}$, and $\dot{\Theta}$ do not depend on r. Hence, it is not actually necessary to solve the above ODE for r. This is a special property of the minimum time problem which ultimately allows us to compute globally optimal minimum time trajectories of a time-varying system without solving the associated time-varying HJB equation.

Condition 3 is the free final time end condition. Applying (7.2) i.e. Condition 2 at time $t = t_f$ yields

$$(p_f, q_f) = -r_f(\cos\Theta_f, \sin\Theta_f),$$

where $p_f = p(t_f)$, etc., which allows one to write Condition 3 as

$$[u_f + s\cos\Theta_f](-r_f)\cos\Theta_f + [v_f + s\sin\Theta_f](-r_f)\sin\Theta_f + 1 = 0,$$

where $u_f = u(x_f, y_f, t_f)$ and $v_f = v(x_f, y_f, t_f)$, which simplifies to

$$-r_f u_f \cos\Theta_f - r_f s \cos^2\Theta_f - r_f v_f \sin\Theta_f - r_f s \sin^2\Theta_f = -1,$$

which simplifies to

$$r_f[u_f \cos\Theta_f + v_f \sin\Theta_f + s] = 1,$$

which may be written as

$$r_f = \frac{1}{u_f \cos\Theta_f + v_f \sin\Theta_f + s}. \tag{7.4}$$

This implies $-u_f \cos\Theta_f - v_f \sin\Theta_f < s$, which says that the component of the flow vector (u_f, v_f) in the direction opposite that of the control Θ_f must be less than the speed s of the vehicle. This make practical sense, since otherwise the vehicle would be steering directly away from the target at the time it arrives instead of directly towards it.

In the present computational framework, however, it is better to relax this free final time end condition entirely, and allow any final input angle Θ_f. The reasons for this are the following:

1. If computing backwards in time, as in [23], it's easier to include suboptimal values of $(\Theta_f, t_f) \in \mathrm{S}^1 \times [t_0, t_0 + T^{max}]$ than numerically "trimming" them away (which would not be the same as the trimming procedure of the present work).

2. If computing forwards in time, one *must* consider all values of $\theta = \Theta(t_0)$. This is illustrated in the example of Section 2.4.2, where the reachability front folds back on itself due to the time-variability of the flow.

3. If the magnitude of (u_f, v_f) exceeds (respectively, equals) the speed s of the vehicle, there are two values (is one value) of Θ_f for which solving (2.4) does yield a globally optimal trajectory, but Condition 3 above is not technically satisfied, because the denominator $-u_f \cos\Theta_f - v_f \sin\Theta_f + s$ in (7.4) is 0 and thus the magnitude r_f of the final co-state is undefined. A trajectory of this type was shown in Figure 2.4(b).

Recall our algorithm determines the globally optimal trajectory from among these locally optimal or extremal trajectories by direct comparison of their costs.

In summary, we have a relaxed minimum time necessary condition encompassed by the single equation (7.3). The present solution method works by sampling $\theta \in \mathrm{S}^1$ and solving the resulting minimum time Euler Lagrange equations (2.4) forward-in-time. Alternatively, one can sample $(\Theta_f, t_f) \in \mathrm{S}^1 \times [t_0, t_0 + T^{max}]$ and solve (2.4) backward-in-time, thus obtaining the time-varying time-to-go function on an irregular grid (as in [23]), but this requires an order of magnitude more computation.

7.2 Modification to handle boundaries of the flow domain

To deal with domain boundaries such as coastline, we introduce the function $g : \mathrm{R}^2 \to [0, 1]$ and modify the state equation (2.1) as follows:

$$\dot{x} = u(x, y, t) + g(x, y)s\cos\Theta;$$

$$\dot{y} = v(x, y, t) + g(x, y)s\sin\Theta.$$

Then the Hamiltonian function becomes

$$\mathcal{H}(x, y, \Theta, p, q, t) := [u + gs\cos\Theta]p + [v + gs\sin\Theta]q + 1,$$

where $u = u(x, y, t), v = v(x, y, t)$, and $g = g(x, y)$, and the co-state equations become

$$\dot{p} = -(u_x + g_x s\cos\Theta)p - (v_x + g_x s\sin\Theta)q;$$

$$\dot{q} = -(u_y + g_y s\cos\Theta)p - (v_y + g_y s\sin\Theta)q,$$

where $g_x = g_x(x, y)$ and $g_y = g_y(x, y)$ are the partial derivatives of g. These are a generalization of (7.1). Similarly, they lead to

$$\dot{\Theta} = -u_y\cos^2\Theta + (u_x - v_y)\sin\Theta\cos\Theta + v_x\sin^2\Theta + (g_x s\sin\Theta - g_y s\cos\Theta).$$

This is a generalization of (7.3) and appears in (2.5).

The point-wise optimal input angle becomes

$$\Theta = \mathrm{atan2}(-gq, -gp).$$

As in Appendix 7.1, for this angle to be well-defined, we rely upon the fact that $\forall t \in [t_0, t_f], (p, q) \neq 0$. In addition, we choose g such that $g(x, y) = 0 \implies \forall t \in$

R, $u(x, y, t) = v(x, y, t) = 0$. Then if along an extremal trajectory $g(x, y) = 0$, we know $\forall t \in [t_0, t_f], \dot{x} = 0$ and $\dot{y} = 0$ identically, and thus the extremal trajectory is entirely outside the flow domain.

The free final time end condition (7.4) becomes

$$r_f = \frac{1}{u_f \cos\Theta_f + v_f \sin\Theta_f + sg_f},$$

where $g_f = g(x_f, y_f)$.

Near the flow domain boundaries, the speed of the vehicle will effectively be reduced from s to $g(x, y)s$. Strictly speaking, this is not in keeping with a true minimum time criterion. However, the trajectories in the interior of the flow domain which are the main focus of this work are not effected. An alternative approach that would have the same effect is to enforce the fixed vehicle speed s but explicitly change the cost function—the integrand in (2.3), currently equal to 1—to include a position penalty term that is nonzero where our function g is 0, and infinite outside the domain, thereby imposing a "soft constraint" on the state of the vehicle.

7.3 Minimum time control in linear time-invariant flows

We now consider flow fields of the form

$$\begin{bmatrix} u(x,y,t) \\ v(x,y,t) \end{bmatrix} = \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},$$

where the partials u_x, u_y, v_x, and v_y are constants. For such flow fields, the evolution of θ, given by Equation (2.4), is independent of x and y. Thus solutions are of the form

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = e^{At} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + s \int_0^t e^{A(t-\tau)} \begin{bmatrix} \cos(\theta(\tau)) \\ \sin(\theta(\tau)) \end{bmatrix} d\tau. \tag{7.5}$$

Remember, although the dynamics of the flow field are linear and the control vector enters linearly here, the problem of determining the optimal control is nonlinear, since it involves constraining the control vector to a circle.

The three examples considered are a pure rotation flow field, i.e. an elliptic fixed point, a pure strain flow field, i.e. a saddle, and a pure shear flow field. These are often viewed as building blocks for more complex, nonlinear flow fields. Note however that, unlike the rest of the examples, these linear flow fields are unboundedly strong away from the origin, and thus the long-term behavior of the closed form solutions we obtain is somewhat unrealistic for the application at hand.

For the first of these examples, we express the extremal trajectories $x = x(\theta_0, t)$, $y = y(\theta_0, t)$, and $\theta = \theta(\theta_0, t)$ in closed form. For the second and third, we only express θ in closed form, not attempting to compute the integrals for x and y, since they are more difficult.

Finally, one may observe an absence of shocks in the solutions for these examples. We do not attempt to explain this analytically.

7.3.1 Pure rotation case

Consider the case where

$$A = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix}$$

and thus the matrix exponential is

$$e^{At} = \begin{bmatrix} \cos(\omega t) & \sin(\omega t) \\ -\sin(\omega t) & \cos(\omega t) \end{bmatrix}.$$

Equation (2.4) yields $\dot{\theta}(t) = -\omega$, which has solutions of the form $\theta(t) = \theta_0 - \omega t$. Thus the integrand in Equation (7.5) is

$$\begin{bmatrix} \cos(\omega t - \omega\tau) & \sin(\omega t - \omega\tau) \\ -\sin(\omega t - \omega\tau) & \cos(\omega t - \omega\tau) \end{bmatrix} \begin{bmatrix} \cos(\theta_0 - \omega\tau) \\ \sin(\theta_0 - \omega\tau) \end{bmatrix} = \begin{bmatrix} \cos(\theta_0 - \omega t) \\ \sin(\theta_0 - \omega t) \end{bmatrix},$$

where we have applied the identities

$$\cos\alpha\cos\beta + \sin\alpha\sin\beta = \cos(\alpha - \beta) = \cos(\beta - \alpha)$$

and

$$-(\sin\alpha\cos\beta - \cos\alpha\sin\beta) = -\sin(\alpha-\beta) = \sin(\beta-\alpha).$$

This integrand does not depend on the variable of integration τ. Thus the solution is simply

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \cos(\omega t) & \sin(\omega t) \\ -\sin(\omega t) & \cos(\omega t) \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + st \begin{bmatrix} \cos(\theta_0 - \omega t) \\ \sin(\theta_0 - \omega t) \end{bmatrix},$$

or, if r_0 and ξ_0 are the polar coordinates of the initial position (x_0, y_0),

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = r_0 \begin{bmatrix} \cos(\xi_0 - \omega t) \\ \sin(\xi_0 - \omega t) \end{bmatrix} + st \begin{bmatrix} \cos(\theta_0 - \omega t) \\ \sin(\theta_0 - \omega t) \end{bmatrix}. \tag{7.6}$$

In summary, we have written θ as well as x and y in closed form, as functions of time t and initial angle θ_0. Thus there is no need to numerically integrate Equation (2.4), just to determine the values of θ_0 and t for which $(x(t), y(t)) = (x_f, y_f)$ in Equation (7.6) and t is minimized.

The first term is due to the flow field itself. It consists of a circular periodic orbit about the origin. The second term is due to the minimum time control. It consists of a steady spiral outward from the origin.

In the special case where θ_0 is aligned with ξ_0 (including the case where $(x_0, y_0) = 0$), the two terms can be combined into one, with the coefficient $r_0 + st$ describing the distance from the origin at time t, and it is easy to see that the control vector remains normal to the streamlines of the flow field, i.e. directed radially outward. However, in general, the angle of the control vector relative to

the flow field varies over time, and the effect of summing the two terms depends on the speed of the vehicle relative to the frequency of rotation of the flow field.

Figure 7.1 shows the pure rotation flow field, the arrival time function, and 12 minimum time trajectories for each of three different initial positions (x_0, y_0). The final positions (x_f, y_f) are simply sampled from a circle of radius 0.5 about the origin. For all three cases $y_0 = 0$. The case $x_0 = 0$ is the one alluded to, where the control is normal to the streamlines. For $x_0 = \frac{1}{2\pi}$, the magnitude of the flow field matches the vehicle speed exactly at time $t = 0$. And for $x_0 = \frac{1}{\pi}$, it exceeds the vehicle speed by a factor of two, resulting in a sharp jump discontinuity in the arrival time function, which bears a striking resemblance to that of Section 2.4.2 seen in Figure 2.5(b). In both cases, the nominal trajectories and the reachability fronts are circles. But in the case of Section 2.4.2, the flow is time-varying, whereas in the current example, the control is time-varying.

7.3.2 Pure strain case

Consider the case where

$$A = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} = \begin{bmatrix} \gamma & 0 \\ 0 & -\gamma \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

and thus the matrix exponential is

$$e^{At} = \begin{bmatrix} e^{\gamma t} & 0 \\ 0 & e^{-\gamma t} \end{bmatrix}$$

Equation (2.4) yields

$$\dot{\theta} = 2\gamma \sin\theta \cos\theta,$$

which has solutions of the form

$$\theta = \left\{ \begin{array}{ll} \theta_0 & \theta_0 = -\frac{1}{2}\pi \\ \arctan\left(\tan\theta_0 e^{2\gamma t}\right) & \theta_0 \in \left(-\frac{1}{2}\pi, \frac{1}{2}\pi\right) \\ \theta_0 & \theta_0 = \frac{1}{2}\pi \\ \arctan\left(\tan\theta_0 e^{2\gamma t}\right) + \pi & \theta_0 \in \left(\frac{1}{2}\pi, \frac{3}{2}\pi\right) \end{array} \right\}, \tag{7.7}$$

where for convenience we consider $\theta_0 \in \left[-\frac{1}{2}\pi, \frac{3}{2}\pi\right)$ instead of $\theta_0 \in [0, 2\pi)$. Moreover,

$$\cos\theta = \left\{ \begin{array}{ll} 0 & \theta_0 = -\frac{1}{2}\pi \\ \frac{1}{\sqrt{\tan^2\theta_0 e^{4\gamma t}+1}} & \theta_0 \in \left(-\frac{1}{2}\pi, \frac{1}{2}\pi\right) \\ 0 & \theta_0 = \frac{1}{2}\pi \\ \frac{-1}{\sqrt{\tan^2\theta_0 e^{4\gamma t}+1}} & \theta_0 \in \left(\frac{1}{2}\pi, \frac{3}{2}\pi\right) \end{array} \right\}$$

and

$$\sin\theta = \left\{ \begin{array}{ll} -1 & \theta_0 = -\frac{1}{2}\pi \\ \frac{\tan\theta_0 e^{2\gamma t}}{\sqrt{\tan^2\theta_0 e^{4\gamma t}+1}} & \theta_0 \in \left(-\frac{1}{2}\pi, \frac{1}{2}\pi\right) \\ 1 & \theta_0 = \frac{1}{2}\pi \\ \frac{-\tan\theta_0 e^{2\gamma t}}{\sqrt{\tan^2\theta_0 e^{4\gamma t}+1}} & \theta_0 \in \left(\frac{1}{2}\pi, \frac{3}{2}\pi\right) \end{array} \right\}.$$

In summary, we have analytically computed θ and the corresponding optimal control vector in closed form, as a function of time t and initial angle θ_0, but we do not attempt to compute the integral for x and y, leaving the solution in the

form

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + s \int_0^t \begin{bmatrix} 1 & t-\tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta(\tau)) \\ \sin(\theta(\tau)) \end{bmatrix} d\tau.$$

We must still solve this numerically, as well as extract the optimal trajectories.

Figure 7.2 shows the Equation (7.7) control signals for various initial angles θ_0, using flow parameter $\gamma = -1$. The control is repelled from $\theta = \pm\frac{\pi}{2}$ and stabilizes at $\theta \in \{0, \pi\}$, regardless of the initial position (x_0, y_0). This should mean that the reachability front tends to flatten out vertically over time, as the control becomes balanced by the strength of the y component of the flow field.

Figure 7.4 shows the pure strain flow field, the arrival time function, and 12 minimum time trajectories for each of three different values of x_0. Indeed the reachability front flattens out as expected. The final positions (x_f, y_f) are simply sampled from a 0.5 by 1.5 ellipse about the origin. For all three cases $y_0 = 0$. For $x_0 = 1$, the magnitude of the flow field matches the vehicle speed exactly at time $t = 0$. And for $x_0 = 2$, it exceeds the vehicle speed by a factor of two, resulting not only in a sharp jump discontinuity in the arrival time function, but a jump to infinity. In general the reachability front gets driven out of the region where the flow is stronger than the vehicle, and begins stretching to fill the region $(-1, 1) \times \mathrm{R}$.

7.3.3 Pure shear case

Consider the case where

$$A = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} = \begin{bmatrix} 0 & \xi \\ 0 & 0 \end{bmatrix}$$

and thus the matrix exponential is

$$e^{At} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}.$$

Equation (2.4) yields

$$\dot{\theta} = -\xi \cos^2 \theta,$$

which has solutions of the form

$$\theta = \left\{ \begin{array}{ll} \theta_0 & \theta_0 = -\frac{1}{2}\pi \\ \arctan\left(\tan\theta_0 - \xi t\right) & \theta_0 \in \left(-\frac{1}{2}\pi, \frac{1}{2}\pi\right) \\ \theta_0 & \theta_0 = \frac{1}{2}\pi \\ \arctan\left(\tan\theta_0 - \xi t\right) + \pi & \theta_0 \in \left(\frac{1}{2}\pi, \frac{3}{2}\pi\right) \end{array} \right\}, \tag{7.8}$$

where again for convenience we consider $\theta_0 \in \left[-\frac{1}{2}\pi, \frac{3}{2}\pi\right)$ instead of $\theta_0 \in [0, 2\pi)$.

Moreover,

$$\cos\theta = \left\{ \begin{array}{ll} 0 & \theta_0 = -\frac{1}{2}\pi \\ \frac{1}{\sqrt{(\tan\theta_0 - \xi t)^2 + 1}} & \theta_0 \in \left(-\frac{1}{2}\pi, \frac{1}{2}\pi\right) \\ 0 & \theta_0 = \frac{1}{2}\pi \\ \frac{-1}{\sqrt{(\tan\theta_0 - \xi t)^2 + 1}} & \theta_0 \in \left(\frac{1}{2}\pi, \frac{3}{2}\pi\right) \end{array} \right\}$$

and

$$\sin\theta = \left\{ \begin{array}{ll} -1 & \theta_0 = -\frac{1}{2}\pi \\ \frac{\tan\theta_0 - \xi t}{\sqrt{(\tan\theta_0 - \xi t)^2 + 1}} & \theta_0 \in \left(-\frac{1}{2}\pi, \frac{1}{2}\pi\right) \\ 1 & \theta_0 = \frac{1}{2}\pi \\ \frac{-\tan\theta_0 + \xi t}{\sqrt{(\tan\theta_0 - \xi t)^2 + 1}} & \theta_0 \in \left(\frac{1}{2}\pi, \frac{3}{2}\pi\right) \end{array} \right\}.$$

In summary, we have analytically computed θ and the corresponding optimal control vector in closed form, as a function of time t and initial angle θ_0, but we do not attempt to compute the integral for x and y, leaving the solution in the form

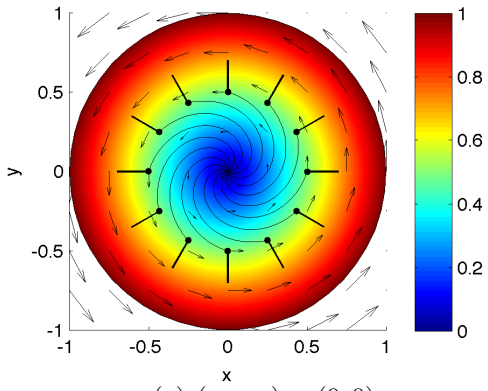
$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + s \int_0^t \begin{bmatrix} 1 & t-\tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta(\tau)) \\ \sin(\theta(\tau)) \end{bmatrix} d\tau.$$

We must still solve this numerically, as well as extract the optimal trajectories.

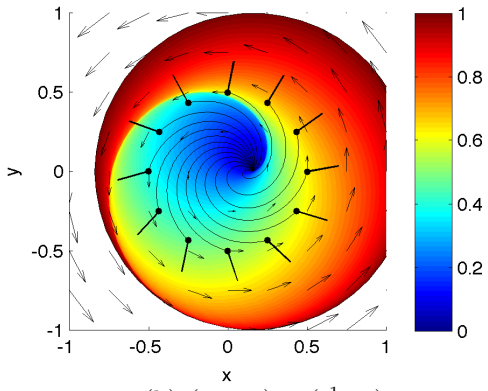
Figure 7.3 shows the Equation (7.8) control signals for various initial angles θ_0, using flow parameter $\xi = 1$. The headings θ are attracted to the fixed points $\theta = \pm\frac{1}{2}\pi$ from above and repelled from below, regardless of the initial position (x_0, y_0). This should mean that the reachability front tends to flatten out horizontally over time, as the control becomes transverse to the streamlines.

Figure 7.5 shows the pure shear flow field, the arrival time function, and 12 minimum time trajectories for each of three different values of y_0. Indeed the reachability front flattens out as expected. The final positions (x_f, y_f) are simply sampled from a circle of radius 0.4 about the origin. For all three cases $x_0 = 0$. For the case $y_0 = 0.1$, the magnitude of the flow field matches the vehicle speed

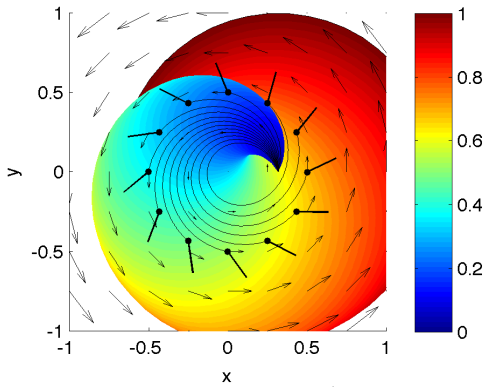
exactly at time $t = 0$. And for $y_0 = 0.2$, it exceeds the vehicle speed by a factor of two, resulting in a sharp jump discontinuity in the arrival time function.



(a) $(x_0, y_0) = (0, 0)$



(b) $(x_0, y_0) = (\frac{1}{2\pi}, 0)$



(c) $(x_0, y_0) = (\frac{1}{\pi}, 0)$

Figure 7.1: Arrival time function and 12 minimum time trajectories for pure rotation flow example of Section 7.3.1, for the three different initial positions. In (a), (b), and (c), the magnitude of flow at the initial position is 0, 1, and 2 times the speed of the vehicle, respectively; hence the appearance of a discontinuity. In (a) only, the difference between the control angle θ and the angle of the local flow vector is constant.

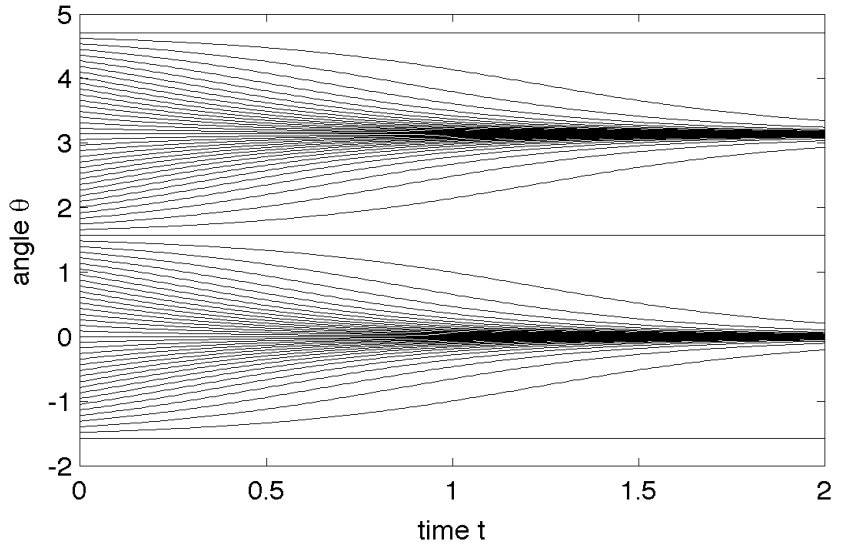


Figure 7.2: Minimum time heading control θ in for pure strain flow example of Section 7.3.2 for various initial angles θ_0 and flow parameter $\gamma = -1$. The heading is repelled from $\theta \in \{0, \pi\}$ and attracted to $\theta = \pm\frac{\pi}{2}$, regardless of the initial position (x_0, y_0).

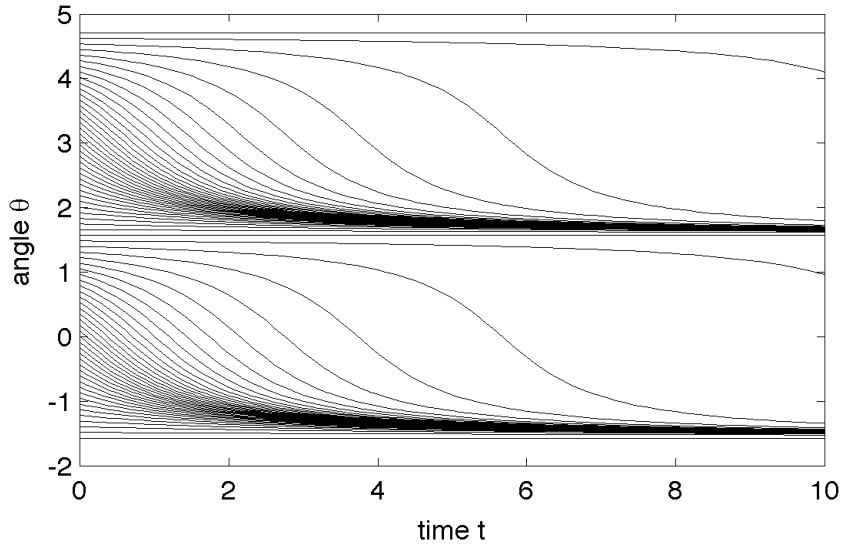
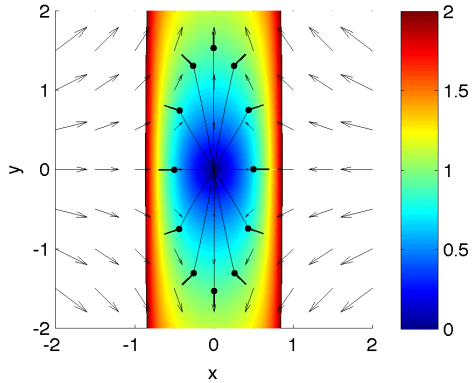
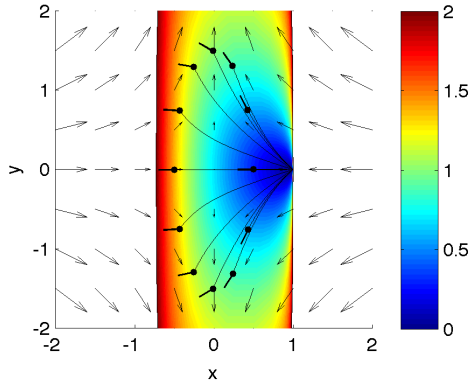


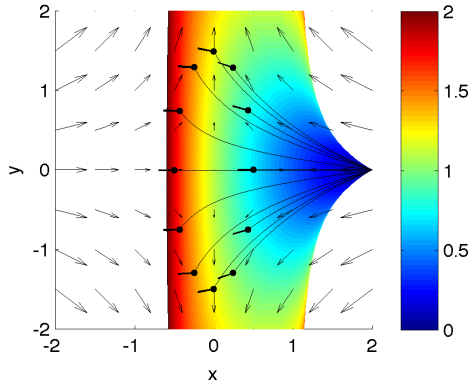
Figure 7.3: Minimum time heading control θ in for pure shear flow example of Section 7.3.3 for various initial angles θ_0 and flow parameter $\xi = 1$. The heading is both repelled by $\theta = \pm\frac{\pi}{2}$ from below and attracted from above, regardless of the initial position (x_0, y_0).



(a) $(x_0, y_0) = (0, 0)$

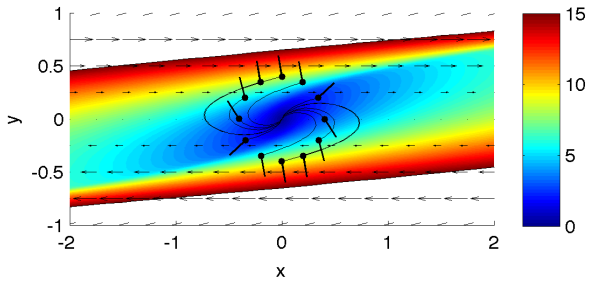


(b) $(x_0, y_0) = (1, 0)$

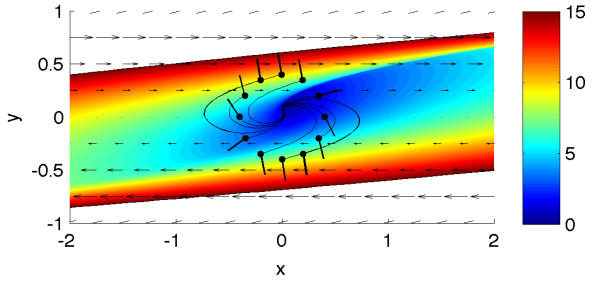


(c) $(x_0, y_0) = (2, 0)$

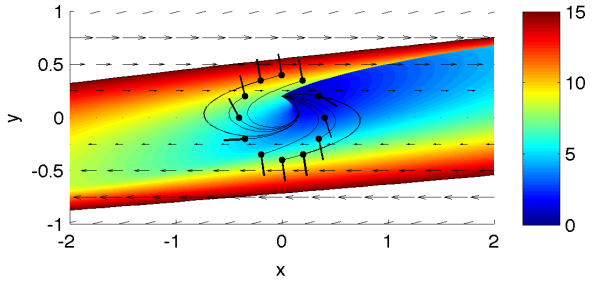
Figure 7.4: Arrival time function and 12 minimum time trajectories for pure shear flow example of Section 7.3.2, for the three different initial positions. In (a), (b), and (c), the magnitude of flow at the initial position is 0, 1, and 2 times the speed of the vehicle, respectively; hence the appearance of a discontinuity. As the reachable set flattens out vertically, the vehicle heading balances out the x component of the flow.



(a) $y_0 = 0.0$



(b) $y_0 = 0.1$



(c) $y_0 = 0.2$

Figure 7.5: Arrival time function and 12 minimum time trajectories for pure shear flow example of Section 7.3.3, for the three different initial positions. In (a), (b), and (c), the magnitude of flow at the initial position is 0, 1, and 2 times the speed of the vehicle, respectively; hence the appearance of a discontinuity. As the reachable set flattens out horizontally, the vehicle heading aligns itself transverse to the flow.

Bibliography

[1] CC Eriksen, TJ Osse, RD Light, T Wen, TW Lehman, PL Sabin, JW Ballard, and AM Chiodi. Seaglider: A long-range autonomous underwater vehicle for oceanographic research. *IEEE Journal of Oceanic Engineering*, 26(4):424–436, Oct 2001.

[2] J. Sherman, R.E. Davis, WB Owens, and J. Valdes. The autonomous underwater glider spray. *Oceanic Engineering, IEEE Journal of*, 26(4):437–446, 2001.

[3] DC Webb, PJ Simonetti, and CP Jones. SLOCUM: An underwater glider propelled by environmental energy. *IEEE Journal of Oceanic Engineering*, 26(4):447–452, Oct 2001.

[4] N.E. Leonard, D.A. Paley, F. Lekien, R. Sepulchre, D.M. Fratantoni, and R.E. Davis. Collective motion, sensor networks, and ocean sampling. *Proceedings of the IEEE*, 95(1):48 –74, Jan. 2007.

[5] NE Leonard and JG Graver. Model-based feedback control of autonomous underwater gliders. *IEEE Journal of Oceanic Engineering*, 26(4):633–645, Oct 2001.

[6] L. Lapierre and D. Soetanto. Nonlinear path-following control of an auv. *Ocean engineering*, 34(11-12):1734–1744, 2007.

[7] A.E. Bryson and Y.C. Ho. *Applied optimal control*. Taylor & Francis Bristol, PA, 1975.

[8] T. Holzhüter. Optimal regulator for the inverted pendulum via Euler–Lagrange backward integration. *Automatica*, 40(9):1613–1620, 2004.

[9] H.M. Osinga and J. Hauser. The geometry of the solution set of nonlinear optimal control problems. *Journal of Dynamics and Differential Equations*, 18(4):881–900, 2006.

[10] Richard Ernest Bellman. An introduction to the theory of dynamic programming. *Rand Corporation*, 1953.

[11] M.G. Crandall and P.L. Lions. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 277(1):1–42, 1983.

[12] Martino Bardi and Italo Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer, 2008.

[13] M. Falcone and R. Ferretti. Semi-lagrangian schemes for hamilton–jacobi equations, discrete representation formulae and godunov methods. *Journal of computational physics*, 175(2):559–575, 2002.

[14] A. Kumar and A. Vladimirsky. An efficient method for multiobjective optimal control and optimal control subject to integral constraints. *arXiv preprint arXiv:0901.3977*, 2009.

[15] Alexander Vladimirsky and Changxi Zheng. A fast implicit method for time-dependent hamilton-jacobi pdes. *arXiv preprint arXiv:1306.3506*, 2013.

[16] T. Nishida, K. Sugihara, and M. Kimura. Stable marker-particle method for the Voronoi diagram in a flow field. *Journal of Computational and Applied Mathematics*, 202(2):377–391, 2007.

[17] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[18] John Strain. Tree methods for moving interfaces. *Journal of Computational Physics*, 151(2):616–648, 1999.

[19] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 457–462. ACM, 2004.

[20] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[21] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer, 2003.

[22] J.A. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Number 3. Cambridge Univ Pr, 1999.

[23] B. Rhoads, I. Mezić, and A. Poje. Minimum time feedback control of autonomous underwater vehicles. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5828–5834. IEEE, 2010.

[24] Blane Rhoads, Igor Mezić, and Andrew C Poje. Minimum time heading control of underpowered vehicles in time-varying ocean currents. *Ocean Engineering*, 66:12–31, 2013.

[25] T. Lolla, MP Ueckermann, K. Yigit, PJ Haley Jr, and PFJ Lermusiaux. Path planning in time dependent flow fields using level set methods. In *IEEE International Conference on Robotics and Automation*, 2012 (Accepted).

[26] Ian M Mitchell. The flexible, extensible and efficient toolbox of level set methods. *Journal of Scientific Computing*, 35(2-3):300–329, 2008.

[27] JA Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4):1591, 1996.

[28] JN Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[29] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[30] J.A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms. *SIAM Journal on Numerical Analysis*, pages 325–363, 2004.

[31] James A Sethian. Fast marching methods. *SIAM review*, 41(2):199–235, 1999.

[32] Ron Kimmel and James A Sethian. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences*, 95(15):8431–8435, 1998.

[33] JA Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton–Jacobi equations. *Proceedings of the National Academy of Sciences of the United States of America*, 98(20):11069, 2001.

[34] A Vladimirsky. Static PDEs for time-dependent control problems. *Interfaces and Free Boundaries*, 8(3):281, 2006.

[35] Shawn C Shadden, Francois Lekien, and Jerrold E Marsden. Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, 212(3):271–304, 2005.

[36] Igor Mezić, S Loire, Vladimir A Fonoberov, and P Hogan. A new mixing diagnostic and gulf oil spill movement. *Science*, 330(6003):486–489, 2010.

[37] Tamer Inanc, Shawn C Shadden, and Jerrold E Marsden. Optimal trajectory generation in ocean flows. In *American Control Conference*, 2005.

[38] Carmine Senatore and Shane D Ross. Fuel-efficient navigation in complex flows. In *American Control Conference*, 2008.

[39] F Lekien and J Marsden. Tricubic interpolation in three dimensions. *International Journal for Numerical Methods in Engineering*, 63(3):455–471, 2005.

[40] Steven L Brunton and Clarence W Rowley. Fast computation of finite-time lyapunov exponent fields for unsteady flows. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(1):017503–017503, 2010.

[41] Philippe Miron, Jérôme Vétel, André Garon, Michel Delfour, and Mouhammad El Hassan. Anisotropic mesh adaptation on lagrangian coherent structures. *Journal of Computational Physics*, 231(19):6419–6437, 2012.

[42] Clement Petres. *Trajectory planning for autonomous underwater vehicles*. PhD thesis, Heriot-Watt University, 2007.

[43] Ken Alton and Ian M Mitchell. An ordered upwind method with precomputed stencil and monotone node acceptance for solving static convex hamilton-jacobi equations. *Journal of Scientific Computing*, 51(2):313–348, 2012.

[44] James A Sethian and Alexander Vladimirsky. Fast methods for the eikonal and related hamilton–jacobi equations on unstructured meshes. *Proceedings of the National Academy of Sciences*, 97(11):5699–5703, 2000.

[45] Christoph Garth, Florian Gerhardt, Xavier Tricoche, and Hans Hagen. Efficient computation and visualization of coherent structures in fluid flow applications. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1464–1471, 2007.

[46] Francois Lekien and Shane D Ross. The computation of finite-time lyapunov exponents on unstructured meshes and for non-euclidean manifolds. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(1):017505–017505, 2010.

[47] Joseph M Maubach. Local bisection refinement for n-simplicial grids generated by reflection. *SIAM Journal on Scientific Computing*, 16(1):210–227, 1995.

[48] Laszlo Techy and Craig A. Woolsey. Minimum-Time Path Planning for Unmanned Aerial Vehicles in Steady Uniform Winds. *Journal of Guidance, Control, and Dynamics*, 32(6):1736–1746, Nov-Dec 2009.

[49] Russ E. Davis, Naomi E. Leonard, and David M. Fratantoni. Routing strategies for underwater gliders. *Deep-Sea Research Part II-Topical Studies in Oceanography*, 56(3-5):173–187, Feb 2009.

[50] J. Witt and M. Dunbabin. Go with the flow: Optimal AUV path planning in coastal environments. In *Proceedings of the Australasian Conference on Robotics and Automation*, 2008.

[51] D. Kruger, R. Stolkin, A. Blum, and J. Briganti. Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments. In *Proceedings of the International Conference on Robotics and Automation*, pages 4265–4270, 2007.

[52] D. Rao and S.B. Williams. Large-scale path planning for underwater gliders in ocean currents. In *Australasian Conference on Robotics and Automation (ACRA)*, 2009.

[53] B. Garau, A. Alvarez, and G. Oliver. Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an a* approach. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 194–198. IEEE, 2005.

[54] M. Soulignac, P. Taillibert, and M. Rueher. Time-minimal path planning in dynamic current fields. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 2473–2479. IEEE, 2009.

[55] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[56] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane. Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics*, 23(2):331–341, 2007.

[57] J. Elston and E. Frew. Unmanned aircraft guidance for penetration of pretornadic storms. In *Proc. AIAA Guidance, Navigation and Control Conf. and Exhibit (electronic), number AIAA-2008-6513, Honolulu, Hawaii*, 2008.

[58] Efstathios Bakolas and Panagiotis Tsiotras. Minimum-time paths for a light aircraft in the presence of regionally-varying strong winds. *Georgia Institute of Technology*, 2010.

[59] Kaleem Siddiqi, Sylvain Bouix, Allen Tannenbaum, and Steven W Zucker. Hamilton-jacobi skeletons. *International Journal of Computer Vision*, 48(3):215–231, 2002.

[60] B. Rhoads, I. Mezić, and A. Poje. Efficient guidance in finite time flow fields. In *Decision and Control (CDC), 2013 52nd IEEE Conference on*. IEEE, 2013 (Accepted).

[61] J. Mahoney, D. Bargteil, M. Kingsbury, K. Mitchell, and T. Solomon. Invariant barriers to reactive front propagation in fluid flows. *EPL (Europhysics Letters)*, 98(4):44005, 2012.

[62] PJ Martin, JW Book, JD Doyle, and Naval Research Lab Stennis Space Center MS Oceanography Div. Simulation of the northern Adriatic circulation during winter 2003. 2006.

[63] Milena Veneziani, Annalisa Griffa, and Pierre-Marie Poulain. Historical drifter data and statistical prediction of particle motion: A case study in the central Adriatic Sea. *Journal of Atmospheric and Oceanic Technology*, 24(2):235–254, Feb 2007.

[64] A. C. Haza, A. Griffa, P. Martin, A. Molcard, T. M. Ozgoekmen, A. C. Poje, R. Barbanti, J. W. Book, P. M. Poulain, M. Rixen, and P. Zanasca. Model-based directed drifter launches in the Adriatic Sea: Results from the DART experiment. *Geophysical Research Letters*, 34(10), May 23 2007.

[65] P. J. Martin, J. W. Book, D. M. Burrage, C. D. Rowley, and M. Tudor. Comparison of model-simulated and observed currents in the central Adriatic during DART. *Journal of Geophysical Research-Oceans*, 114, Feb 10 2009.

[66] Igor Mezic. Controllability of hamiltonian systems with drift: Action-angle variables and ergodic partition. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 3, pages 2585–2592. IEEE, 2003.

[67] Tino Weinkauf and Holger Theisel. Streak lines as tangent curves of a derived vector field. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1225–1234, 2010.