Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

OPTIMAL SAMPLING PATHS FOR AUTONOMOUS VEHICLES IN UNCERTAIN OCEAN FLOWS

by
Andrew J. de Stefan

Despite an extensive history of oceanic observation, researchers have only begun to build a complete picture of oceanic currents. Sparsity of instrumentation has created the need to maximize the information extracted from every source of data in building this picture. Within the last few decades, autonomous vehicles, or AVs, have been employed as tools to aid in this research initiative. Unmanned and self-propelled, AVs are capable of spending weeks, if not months, exploring and monitoring the oceans. However, the quality of data acquired by these vehicles is highly dependent on the paths along which they collect their observational data. The focus of this research is to find optimal sampling paths for autonomous vehicles, with the goal of building the most accurate estimate of a velocity field in the shortest time possible.

The two main numerical tools employed in this work are the level set method for time-optimal path planning, and the Kalman filter for state estimation and uncertainty quantification. Specifically, the uncertainty associated with the velocity field is defined as the trace of the covariance matrix corresponding to the Kalman filter equations. The novelty in this work is the covariance tracking algorithm, which evolves this covariance matrix along the time-optimal trajectories defined by the level set method, and determines the path expected to minimize the uncertainty corresponding to the flow field by the end of deployment. While finding optimal sampling paths using this method is straightforward for the single-vehicle problem, it becomes increasingly difficult as the number of AVs grows. As such, an iterative procedure is presented here for multi-vehicle problems, which in simple cases can be

proven to find controls that collectively minimizes the expected uncertainty, assuming that such a minimum exists.

This work demonstrates the utility of combining methods from optimal control theory and estimation theory for learning uncertain fields using fleets of autonomous vehicles. Additionally, it shows that by optimizing over long-duration, continuous trajectories, superior results can be obtained when compared to ad hoc approaches such as a gradient-following control. This is demonstrated for both single-vehicle and multi-vehicle problems, and for static and evolving flow models.

OPTIMAL SAMPLING PATHS FOR AUTONOMOUS VEHICLES IN UNCERTAIN OCEAN FLOWS

by
Andrew J. de Stefan

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology and
Rutgers, The State University of New Jersey – Newark
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Mathematical Sciences

Department of Mathematical Sciences, NJIT
Department of Mathematics and Computer Science, Rutgers-Newark

August 2019

APPROVAL PAGE

OPTIMAL SAMPLING PATHS FOR AUTONOMOUS VEHICLES IN UNCERTAIN OCEAN FLOWS

Andrew J. de Stefan

Richard O. Moore, Dissertation Advisor — Date
Professor of Mathematics, New Jersey Institute of Technology

Wooyoung Choi, Committee Member — Date
Professor of Mathematics, New Jersey Institute of Technology

David G. Shirokoff, Committee Member — Date
Professor of Mathematics, New Jersey Institute of Technology

M. Ani Hsieh, Committee Member — Date
Professor of Mechanical Engineering and Mechanics, University of Pennsylvania

---

Stephen R. Guimond, Committee Member — Date
Professor of Physics, University of Maryland - Baltimore County

BIOGRAPHICAL SKETCH

Author: Andrew J. de Stefan

Degree: Doctor of Philosophy

Date: August 2019

Undergraduate and Graduate Education:

- Doctor of Philosophy in Mathematical Sciences,
  New Jersey Institute of Technology, Newark, NJ, 2019
- Bachelor of Science in Engineering Physics
  Ramapo College of New Jersey, Mahwah, NJ, 2014
- Associate of Science in Business Administration
  Bergen Community College, Paramus, NJ, 2011

Major: Mathematical Sciences

Presentations and Publications:

Andrew J. de Stefan, "Airborne Radar Analysis of Hurricanes," NASA Goddard Space Flight Center Summer Intern Poster Session, Greenbelt, MD, August 2018.

Andrew J. de Stefan, "Optimal Sampling Paths for Autonomous Vehicles in Uncertain Ocean Flows," Center for Applied Mathematics and Statistics (CAMS) Research Day, Newark, NJ, April 2018.

Andrew J. de Stefan, "Numerical Methods for Finding Optimal Sampling Paths for Autonomous Vehicles," 14^{th} Annual Conference on Frontiers in Applied and Computational Mathematics (FACM), Newark, NJ, August 2017.

Andrew J. de Stefan and Yunjie Xu, "Magnetoelastic Properties of an FeNiMoB Amorphous Alloy," School of Theoretical and Applied Science (TAS) Research Symposium, Mahwah, NJ, April 2013.

This work is dedicated to my wife and best friend, Liz.
You are a continuous source of inspiration to me,
and I couldn't have done this without you.
Thank you for always believing in me.

ACKNOWLEDGMENT

First and foremost, I would like to thank my dissertation advisor, Richard Moore, who has served as a role model for me both professionally and personally, and has taught me more than I could ever give him credit for here.

In addition to my advisor, I would like to express my great appreciation to the rest of my dissertation committee: Wooyoung Choi, David Shirokoff, M. Ani Hsieh, and Stephen Guimond. Thank you for always making yourselves available to provide guidance and encouragement whenever it has been needed.

I am profoundly grateful to my classmates: R.J., Mahdi, Malik, and Matt. Your friendship and support has always kept me going, despite the many challenges we have encountered these last five years.

Last, but not least, I would like to thank my family. To my wife, my parents, my siblings, and my in-laws: your unwavering support and unconditional love have been paramount to me, and I would never have made it here without each and every one of you.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A better understanding of our oceans and ocean currents is of growing importance for a variety of environmental, ecological, and economic reasons. One of the most pressing is the large accumulation of marine debris (garbage, plastics, abandoned fishing gear, etc.) in the oceans. Having a better knowledge of ocean currents would aid in the modeling of debris transport and allow for more targeted cleanups [3]. A similar benefit would apply to other cleanup initiatives, such as those following the 2010 Deepwater Horizon Oil Spill in the Gulf of Mexico [7]. Accurate ocean models are also needed for efficiently transporting goods, providing boundary conditions for climate and weather models, search and rescue missions [2], understanding marine ecosystems [6], and much more.

While some of these issues have only recently garnered scientific interest, oceanic data assimilation is by no means a new field. Between the years of 1904 and 1906 the Marine Biological Association of the United Kingdom released over one-thousand weighted bottles into the Northern Sea to study deep water currents. This idea of using weighted devices (also known as floats, or Lagrangian drifters) to study currents continued, and was further advanced in 1955 by John Swallow. Known as the Swallow float, his device used a magnetostrictive "pinger" to send out signals, which allowed the float's depth and position to be determined via acoustic triangulation. Unfortunately, not only could early floats be quite expensive, they also had to remain fairly close to the acoustic receivers in order to be tracked. This gave way to the SOFAR float in 1966. The SOFAR float made use of the Sound Fixing and Ranging (SOFAR) channel, which is an ocean layer (typically 1,000-1,300 meters below the surface) that acts as an underwater waveguide. Using this channel allowed these floats

to send signals much further, approximately 846 kilometers. In addition, they were ultimately constructed using commercially available aluminum tubes and piezoelectric plates, which together acted as a resonating tube. This reduced the cost of fabrication and made it possible to perform studies using large numbers of floats. The first major study to do this was in 1973, in the Mid-Ocean Dynamics Experiment (MODE), where a fleet of twenty SOFAR floats were used to study sub-mesoscale dynamics. [14]

Around this same time period (specifically in the early 1960's) the first autonomous vehicles, or AVs, were constructed. While both AVs and floats serve the same purpose of ocean sampling via direct and indirect measurements, they differ in their capabilities. Floats simply advect with ocean currents, which limits their ability to effectively sample a given domain. AVs, on the other hand, are self-propelled waterborne drones, capable of steering against currents to obtain more expansive data sets. The feedback capabilities of AVs allow them to avoid issues such as clustering, which results in redundant measurements and inefficient sampling [31]. Unfortunately, the first AVs created were typically large, inefficient and expensive, which led many organizations to prefer remotely operated vehicles (ROVs) [53]. It was not until the early 1990's when an economical AV, called Odyssey, was developed at the MIT Sea Grant College Program Autonomous Underwater Vehicle (AUV) Laboratory [1]. The success of Odyssey led to an increase of funding for autonomous vehicle research, which to this day remains a very vital field of interest for a variety of organizations and institutions [8].

Two widely deployed types of AUVs, or gliders [4], are the Spray and Slocum gliders [5], which achieve propulsion by changing their buoyancy using an internal system of bladders. As they ascend and descend through the water, their fixed hydrofoils produce a horizontal thrust, slowly driving the vehicle forward. This energy-efficient method of propulsion allows for gliders to remain deployed for weeks,

if not months, at a time. It is exactly this reason that makes gliders so appealing for oceanographic studies, and that has motivated the work presented here.

Despite the extensive history of oceanic data assimilation, a plethora of different experiments, and various forms of ocean monitoring (which include satellites, boats, buoys, tide stations, etc.), ocean currents still remain largely unknown due to their complexity and high-dimensionality, combined with the sparsity of measurements. It is exactly this reason which motivates the investigation of more efficient ocean sampling methods, and has ultimately resulted in considerable research focused on optimal control for autonomous vehicles. This field of study is incredibly extensive, and includes topics such as optimal control for single AVs [26] [35] [38] [43], coordinated control of multiple vehicles [11] [15] [29] [30] [32] and multi-objective Pareto-optimal control [27] [36] [40], to name a few.

While optimal control theory dates back to the brachistochrone problem and the early development of the calculus of variations, its use in ocean navigation is somewhat more recent. In 1931, Ernst Zermelo proposed his well-known navigation problem, and derived a general solution for the time-minimizing path between any two points in a known velocity field. The advent of computers has greatly expanded the ability to perform path planning in complex flows and with complex objective functions. Many new methods to do this have been developed, including, but not limited to, level set methods [35] [41] [44] [55], graph-based search techniques (Dijkstra's algorithm, A*, fast marching method, etc.) [10] [12] [26] [44] [52], relaxation methods [38], rapidly-exploring random trees (RRTs) [28] [50], and extremal field algorithms [21] [43].

While each of these methods have certain advantages, they also come with drawbacks. The level set method is a front-tracking algorithm capable of finding globally optimal paths, but is subject to CFL conditions. This can result in a high computational cost, depending on the desired refinement of the spatial and temporal discretization. Both graph-based search techniques and RRTs are well

suited for problems which contain obstacles or forbidden regions, however graph-based methods perform poorly for complicated or time-dependent velocities [34], and RRTs typically converge to non-optimal solutions [25]. The relaxation method of [38] used an iterative scheme derived from the Euler-Lagrange equations in order to find locally optimal controls for simple flows. However, this method could potentially run into nonexistence issues for more complex velocities. The extremal field approach essentially solves a Hamilton-Jacobi-Bellman equation using the method of characteristics [21]. Although this method is easy to parallelize [43], it requires calculating an entire family of optimal paths, known as a field of extremals. This requires a large amount of storage, and can be wasteful if only interested in a single optimal path [21].

The research described here focuses on using a level set methodology in combination with data assimilation techniques in order to quantify uncertain ocean currents. The level set method was developed by Sethian and Osher in 1988 [42]. Although this initial paper focused on tracking flame propagation, there has since been an abundance of work investigating the applications of this method. Sethian explores the optimal control applications of his fast marching method in [44], where he discusses first arrival times and shortest paths on manifolds with weighted metrics. Although the problem of first arrival is of most interest to us at the moment, there are a variety of other interesting applications for the level set method in regards to path planning. Some of these include aircraft collision avoidance [51], risk minimization in static environments [54], and determining controls to avoid unsafe states [39].

Most relevant to our current research, however, is the work introduced by Lolla in [34], and briefly described in Sections 2.1.2 and 2.1.3 for finding time-minimizing paths of autonomous vehicles in known velocity fields. This work has been used for a variety of scenarios and applications including flows with forbidden regions, coordinated control of multiple AVs [34], time-optimal path planning in stochastic

flows [32], and finding energy-optimal paths using a stochastic dynamically-orthogonal (DO) level set method [47].

Apart from these highly specialized examples, there has been little work employing the level set method of [34] to find AV controls which optimize more general objective functions. The work described here addresses this gap for autonomous surface vehicles (ASVs) navigating in two-dimensional flows, where the objective function quantifies uncertainty in the flow itself. Similar to [47], the optimal paths we seek will be selected from a set of time-minimizing trajectories, as determined by the level set method.

In Chapter 2, we present an introduction to the level set method that we use to generate optimal AV trajectories, as well as the Kalman filtering methods that we use for state estimation and uncertainty quantification. Chapter 3 makes use of the level set method and Kalman filter to describe an algorithm for determining the optimal path for a single AV in an uncertain ocean flow. Chapter 4 extends the methodology of Chapter 3 to multiple vehicles, and introduces an iterative procedure for finding the optimal controls for all vehicles collectively. Chapter 5 describes a method for modeling more realistic ocean dynamics, making use of the shallow water equations (SWE). Finally, in Chapter 6 we conclude and discuss potential future work.

CHAPTER 2

BACKGROUND

2.1 Level Set Method

2.1.1 Level Set Equation

Consider an autonomous surface vehicle (ASV) initially located at the point $\mathbf{x}_s \in \mathbb{R}^2$ at time t_0, within a body of water that has known surface current $\mathbf{v}(\mathbf{x}, t)$. Now consider the problem of finding the path from this starting location $\mathbf{x}_s$, to some other point $\boldsymbol{\xi}$, which minimizes the objective functional

$$J[\mathbf{x}(t), \hat{\mathbf{u}}(t)] = \int_{t_0}^{\mathcal{T}} \kappa\big(\mathbf{x}(s; \hat{\mathbf{u}}(s))\big) ds, \tag{2.1}$$

where the motion of the AV is dictated by

$$\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}, t) + F(\mathbf{x})\hat{\mathbf{u}}(t) \tag{2.2}$$

with terminal condition $\mathbf{x}(\mathcal{T}) = \boldsymbol{\xi}$. Here, $\dot{\mathbf{x}}$ represents the lab frame velocity of the AV, $\mathbf{v}$ is the velocity field in the medium, F is the speed of the AV which can be spatially-dependent, and $\hat{\mathbf{u}}$ is the unit vector defining the AV's heading.

Assume that each pair $(\boldsymbol{\xi}, \mathcal{T})$ uniquely determines an optimal path $\mathbf{x}(t; \boldsymbol{\xi}, \mathcal{T})$ and optimal control $\hat{\mathbf{u}}(t; \boldsymbol{\xi}, \mathcal{T})$. If $(\boldsymbol{\xi}_1, \mathcal{T}_1)$ sits on this optimal path, then

$$\mathbf{x}(t; \boldsymbol{\xi}_1, \mathcal{T}_1) = \mathbf{x}(t; \boldsymbol{\xi}, \mathcal{T}), \tag{2.3}$$

$$\hat{\mathbf{u}}(t; \boldsymbol{\xi}_1, \mathcal{T}_1) = \hat{\mathbf{u}}(t; \boldsymbol{\xi}, \mathcal{T}). \tag{2.4}$$

Thus, we can define the feedback control

$$\hat{\mathcal{U}}(\boldsymbol{\xi}, \mathcal{T}) = \hat{\mathbf{u}}(\mathcal{T}; \boldsymbol{\xi}, \mathcal{T}). \tag{2.5}$$

Plugging these controls and their corresponding paths into augmented objective functional

$$J[\mathbf{x}(t), \hat{\mathbf{u}}(t)] = \int_{t_0}^{\mathcal{T}} \left[\kappa(\mathbf{x}) + \boldsymbol{\lambda}^T(s)\big(\mathbf{v}(\mathbf{x}, s) + F(\mathbf{x})\hat{\mathbf{u}}(s) - \dot{\mathbf{x}}(s)\big)\right] ds, \tag{2.6}$$

identifies a value function

$$W(\boldsymbol{\xi}, \mathcal{T}) = \min_{\hat{\mathbf{u}}} J[\mathbf{x}(t), \hat{\mathbf{u}}(t)] = J[\mathbf{x}_{\hat{\mathcal{U}}}(t), \hat{\mathcal{U}}(t)], \tag{2.7}$$

where $\mathbf{x}_{\hat{\mathcal{U}}}$ is the path corresponding to control $\hat{\mathcal{U}}$.

Consider replacing the last bit of control $\hat{\mathcal{U}}$ on $(\mathcal{T} - \Delta\mathcal{T}, \mathcal{T})$ with some sub-optimal control $\hat{\boldsymbol{\nu}}$ in Equation (2.7). This gives

$$W(\boldsymbol{\xi}, \mathcal{T}) \leq W(\boldsymbol{\xi} - \Delta\mathbf{x}, \mathcal{T} - \Delta\mathcal{T}) + \int_{\mathcal{T}-\Delta\mathcal{T}}^{\mathcal{T}} \kappa(\mathbf{x}) ds. \tag{2.8}$$

If we assume that W is differentiable, we can Taylor-expand to yield

$$\frac{\partial W}{\partial \mathcal{T}}\Delta\mathcal{T} + \frac{\partial W}{\partial \boldsymbol{\xi}}\Delta\mathbf{x} \leq \int_{\mathcal{T}-\Delta\mathcal{T}}^{\mathcal{T}} \kappa(\mathbf{x}) ds \approx \kappa(\boldsymbol{\xi})\Delta\mathcal{T}. \tag{2.9}$$

Dividing across by $\Delta\mathcal{T}$ and using Equation (2.2) we have that

$$\frac{\partial W}{\partial \mathcal{T}} \leq -\frac{\partial W}{\partial \boldsymbol{\xi}}\Big(\mathbf{v}(\boldsymbol{\xi}, \mathcal{T}) + F(\boldsymbol{\xi})\hat{\boldsymbol{\nu}}\Big) + \kappa(\boldsymbol{\xi}). \tag{2.10}$$

It is important to note that when $\hat{\boldsymbol{\nu}}$ is replaced by $\hat{\mathcal{U}}$, this inequality becomes an equality

$$\begin{aligned} \frac{\partial W}{\partial \mathcal{T}} &= \kappa(\boldsymbol{\xi}) - \frac{\partial W}{\partial \boldsymbol{\xi}}\Big(\mathbf{v}(\boldsymbol{\xi}, \mathcal{T}) + F(\boldsymbol{\xi})\hat{\mathcal{U}}\Big) \\ &= \kappa(\boldsymbol{\xi}) + \min_{\hat{\boldsymbol{\nu}}}\left[- \frac{\partial W}{\partial \boldsymbol{\xi}}\Big(\mathbf{v}(\boldsymbol{\xi}, \mathcal{T}) + F(\boldsymbol{\xi})\hat{\boldsymbol{\nu}}\Big)\right], \end{aligned} \tag{2.11}$$

which suggests that

$$\hat{\mathcal{U}} = \frac{\left(\frac{\partial W}{\partial \boldsymbol{\xi}}\right)^T}{\left|\left|\frac{\partial W}{\partial \boldsymbol{\xi}}\right|\right|}, \tag{2.12}$$

and simplifies Equation (2.11) to a Hamilton-Jacobi-Bellman equation of the form

$$\frac{\partial W}{\partial \mathcal{T}} + \frac{\partial W}{\partial \boldsymbol{\xi}}\mathbf{v}(\boldsymbol{\xi}, \mathcal{T}) + F(\boldsymbol{\xi})\left|\left|\frac{\partial W}{\partial \boldsymbol{\xi}}\right|\right| = \kappa(\boldsymbol{\xi}). \tag{2.13}$$

If we are interested in minimizing the "time-to-go" from $\mathbf{x}_s$ to $\boldsymbol{\xi}$, we define our running cost to be $\kappa(\mathbf{x}) = 1$. To solve Equation (2.13) in this case, we embed W into a smooth, higher dimensional function $\phi(\boldsymbol{\xi}, \mathcal{T})$ such that $\phi(\boldsymbol{\xi}, \mathcal{T}) = 0$ identifies the curve with a minimum time-to-go value of $W(\boldsymbol{\xi}, \mathcal{T}) = \mathcal{T}$. Stated more simply, the zero-level set of ϕ at some time $\mathcal{T}$ represents the set of all points $\boldsymbol{\xi}$ that our AV can reach at that time, assuming an optimal trajectory is followed. By setting

$$\phi(\boldsymbol{\xi}, \mathcal{T}) = W(\boldsymbol{\xi}, \mathcal{T}) - \mathcal{T}, \tag{2.14}$$

and plugging this into Equation (2.13) with $\kappa(\mathbf{x}) = 1$, we get

$$\frac{\partial \phi}{\partial \mathcal{T}} + 1 + \nabla\phi \cdot \mathbf{v} + F||\nabla\phi|| = 1, \tag{2.15}$$

which can be rewritten as

$$\frac{\partial \phi}{\partial \mathcal{T}} + \nabla\phi \cdot \mathbf{v} + F||\nabla\phi|| = 0. \tag{2.16}$$

Equation (2.16) is known as the level set equation, which is an initial value problem that is typically initialized with a signed distance function [41]

$$\phi(\boldsymbol{\xi}, t_0) = |\boldsymbol{\xi} - \mathbf{x}_s|. \tag{2.17}$$

Although we could choose to initialize with any function having the correct zero-level set, a signed distance function has the advantage of having a smooth gradient almost everywhere.

Note that while the level set equation here has been formulated through an optimal path-planning perspective, this equation has a multitude of other applications

including, but not limited to: image enhancement, shape and boundary detection, combustion, fluid mechanics, etching and deposition in microchip fabrication, etc. [41] [44] [48] [49]

2.1.2 Level Set Method

The level set method, developed by James Sethian and Stanley Osher in [42], is a technique used to numerically solve Equation (2.16). The implicit function ϕ, introduced in Section 2.1.1, is initialized as any suitable function with the required zero-level set, and evolved in such a way as to propagate the zero-level set as correctly as possible in time. In addition to countless applications to various contexts where an interface must be tracked numerically, this idea was used by Lolla et al. in [34] [35] for the purpose of finding the minimum time-to-travel in time-varying ocean currents. In that work, the authors develop the following finite difference algorithm for the propagation and advection of the implicit function

$$\begin{aligned}\frac{\phi(\mathbf{x}, t+\Delta t) - \phi(\mathbf{x}, t)}{\Delta t} = &-F\frac{|\nabla\phi(\mathbf{x}, t)| + |\nabla\phi(\mathbf{x}, t)^{**}|}{2} \\ &-\mathbf{v}\left(\mathbf{x}, t+\frac{\Delta t}{2}\right)\cdot\nabla\phi(\mathbf{x}, t)^{*}.\end{aligned} \tag{2.18}$$

This is a multi-step method which is solved using a first-order Godunov scheme for the propagation terms and a second-order Total Variation Diminishing advection scheme for the advective term. Note that the propagation terms are those corresponding to the vehicle's self-propulsion (first term on the right hand side of Equation (2.18)), whereas the advection term corresponds to the vehicle's motion from the underlying current (second term on the right hand side of Equation (2.18)).

Using a level set methodology has a number of benefits. The algorithm can easily be extended into higher spatial dimensions, since calculations in each dimension are independent of one another. The method easily handles topological changes to the zero-level set (i.e., intersections and splitting), which is an advantage over

many particle-tracking algorithms. There are additionally well known adaptations for reducing computational time, including the narrow-band level set method [9] and fast-marching methods [44].

2.1.3 Backtracking Algorithm

The level set algorithm discussed in Section 2.1.2 calculates a reachable set $S(t)$ for the AV at discrete time steps $m\Delta t$ for $m \in \mathbb{N}$. Conditions to terminate the algorithm take two typical forms: one either prescribes an end location $\mathbf{x}_f$ for the AV, or prescribes a maximum amount of time t_f that the AV can be deployed for. In the case where an endpoint is determined beforehand, the algorithm is run until the zero-level set of ϕ passes over $\mathbf{x}_f$. In the latter scenario, the algorithm is simply run until $m\Delta t \geq t_f$. In the latter case, an additional criterion is applied to determine which extremal the AV should follow, i.e., pick $\mathbf{x}_f \in S(t)$.

To find the time-minimizing trajectory, we follow the backtracking method discussed by Lolla et al. in [35]. We consider the AV to be a particle lying on the zero-level set of $\phi(\mathbf{x}, t)$ and use a particle-tracking method backwards in time to find the optimal trajectory $\mathbf{x}_{opt}$. More specifically, starting from the end point $\mathbf{x}_{opt}(t_f) = \mathbf{x}_f$, we use the following first-order finite difference scheme to calculate $\mathbf{x}_{opt}(t - \Delta t)$ until we ultimately reach $\mathbf{x}_{opt}(t_0) = \mathbf{x}_s$

$$\frac{\mathbf{x}_{opt}(t - \Delta t) - \mathbf{x}_{opt}(t)}{\Delta t} = -\mathbf{v}(\mathbf{x}, t) - F(\mathbf{x}) \cdot \hat{\mathbf{n}}, \tag{2.19}$$

where $\hat{\mathbf{n}}$ is the outward facing normal-vector to the zero-level set of ϕ at time t, equal to

$$\hat{\mathbf{n}} = \frac{\nabla\phi(\mathbf{x}, t)}{|\nabla\phi(\mathbf{x}, t)|}. \tag{2.20}$$

It is important to realize that Equation (2.19) calculates the optimal path, and not the optimal control. The optimal control is obtained subsequently from Equation

(2.20). Since we are interested in a control represented by the vehicle's heading, we can assume without loss of generality that $\hat{\mathbf{u}}$ has the form

$$\hat{\mathbf{u}} = \begin{bmatrix} \cos\big(\theta(t)\big) \\ \sin\big(\theta(t)\big) \end{bmatrix}, \tag{2.21}$$

and define our optimal control to be

$$\begin{aligned} \theta(t) &= \text{atan2}(\hat{\mathbf{n}}_y, \hat{\mathbf{n}}_x) \\ &= \text{atan2}\left(\frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial x}\right). \end{aligned} \tag{2.22}$$

2.1.4 Example

To demonstrate the level set method, we will consider the following example from [26]. Assume we have an autonomous surface vehicle in an oceanic domain with surface current modeled by

$$\mathbf{v} = \begin{bmatrix} -\pi A \sin(\frac{\pi x}{s}) \cos(\frac{\pi y}{s}) \\ \pi A \cos(\frac{\pi x}{s}) \sin(\frac{\pi y}{s}) \end{bmatrix}, \tag{2.23}$$

where $A = 0.02$ and $s = 1$. Furthermore, assume our ASV is initially at point $\mathbf{x}_s = (-1, -1)$ at $t_0 = 0$, and would like to find the quickest path from $\mathbf{x}_s$ to $\mathbf{x}_f = (1, 1)$ given a control speed of $F = 0.05$. We compute the solution using the level set methodology described earlier, and compare our solution to the analytical solution given by Zermelo's navigation problem [21].

More explicitly, we use Zermelo's approach to find the following system of ODEs governing the time-optimal solution:

$$\dot{\theta} = -\frac{\pi^2 A}{s}\left[\sin\left(\frac{\pi x}{s}\right)\sin\left(\frac{\pi y}{s}\right) + \cos\left(\frac{\pi x}{s}\right)\cos\left(\frac{\pi y}{s}\right)\sin\big(2\theta\big)\right], \tag{2.24}$$

$$\dot{\mathbf{x}} = \mathbf{v} + F\hat{\mathbf{u}}, \tag{2.25}$$

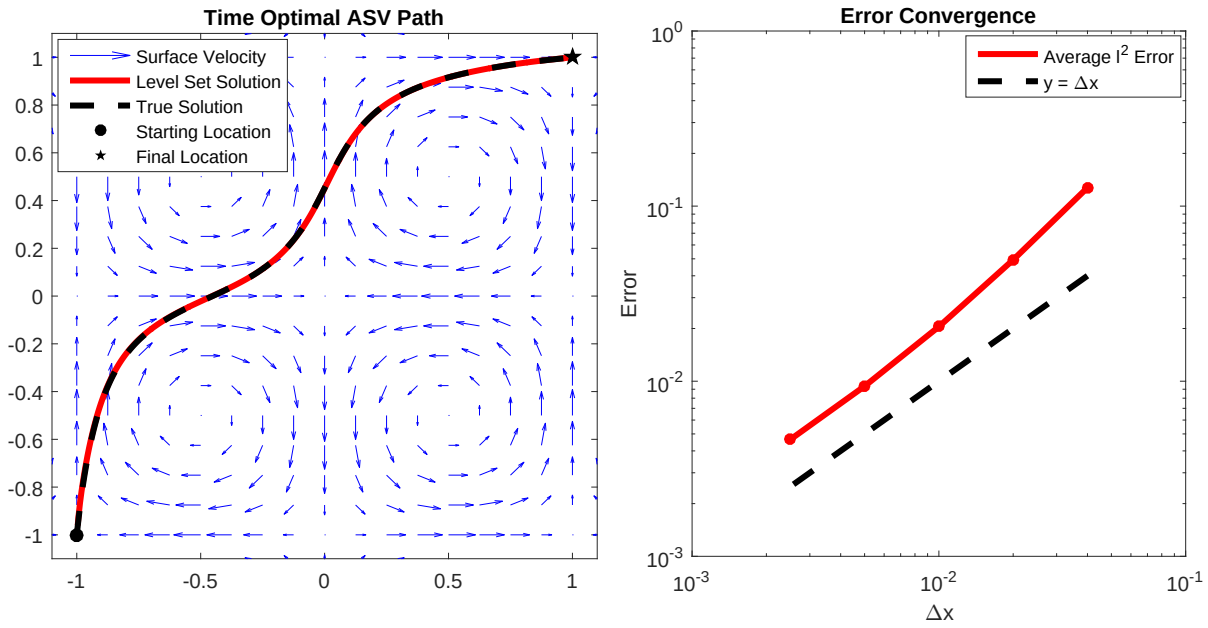

Figure 2.1 Left: Comparison of the level set solution to the true solution. Right: Error convergence plot. The error is computed by finding the Euclidean distance between the numerical path and true path at each time step, and averaging over all of these deviations.

where $\mathbf{v}$ and $\hat{\mathbf{u}}$ are defined in Equations (2.23) and (2.21), respectively. We then solve this system using an explicit adaptive-step Runge-Kutta (4,5) method in MATLAB combined with a shooting method. We refer to this as the true solution for comparison purposes. Our results are shown in Figure 2.1 for $\Delta x = \frac{1}{400}$ and $\Delta t = \frac{1}{1000}$, along with an error convergence plot for $\frac{1}{400} \leq \Delta x \leq \frac{1}{25}$. We can see that the two solutions (left panel of Figure 2.1) are in good agreement with each other and act as expected (that is, the ASV makes use of favorable currents while avoiding opposing currents). Additionally, we see first order convergence in space (right panel of Figure 2.1), which is what we expect for this numerical scheme.

2.2 Kalman Filter

2.2.1 Sequential Kalman Filter

The Kalman filter (KF) provides the optimal (in the sense of mean-squared error) estimate of a linearly evolving Gaussian-distributed state observed through

measurements with Gaussian-distributed error [23]. It can be regarded as a predictor-corrector method involving a forecast (predictor) step in which the state's distribution evolves to the next observation time, followed by an analysis (corrector) step in which noisy observations are optimally incorporated into the state estimate.

Following prediction steps (i.e., before observation), variables will be denoted with a "-" subscript, whereas following correction steps (i.e., post-observation) they will be denoted with a "+" subscript. Observations are assumed to be taken sequentially in time and are indexed by subscript m.

The equations governing the prediction step of the sequential Kalman filter are given by

$$\hat{\mathbf{x}}_{m+1,-} = \mathcal{F}\hat{\mathbf{x}}_{m,+}, \tag{2.26}$$

$$\mathbf{P}_{m+1,-} = \mathcal{F}\mathbf{P}_{m,+}\mathcal{F}^T + \mathbf{Q}, \tag{2.27}$$

where $\hat{\mathbf{x}} \in \mathbb{R}^N$ is the mean value of our state-vector, $\mathcal{F}$ is the $N \times N$ system operator matrix, $\mathbf{P}_{m+1}$ is the covariance matrix of our mean state-vector at observation step $m+1$, and $\mathbf{Q}$ is the covariance matrix of the system noise.

We assume a vector of L linear observations $\mathbf{z}_{m+1} \in \mathbb{R}^L$, of the true signal given by

$$\mathbf{z}_{m+1} = \mathbf{H}_{m+1}\hat{\mathbf{x}}_{m+1} + \sigma^0_{m+1}, \tag{2.28}$$

where the observation operator $\mathbf{H}_{m+1}$ maps $\mathbb{R}^N$ to $\mathbb{R}^L$, and $\sigma^0_{m+1} \in \mathbb{R}^L$ is the zero-mean Gaussian distributed observational noise with covariance matrix $\mathbf{R}^0$.

Using these observations, we can update our predicted values of $\hat{\mathbf{x}}_{m+1,-}$ and $\mathbf{P}_{m+1,-}$, in the following way

$$\hat{\mathbf{x}}_{m+1,+} = \hat{\mathbf{x}}_{m+1,-} + \mathbf{K}_{m+1}(\mathbf{z}_{m+1} - \mathbf{H}_{m+1}\hat{\mathbf{x}}_{m+1,-}), \tag{2.29}$$

$$\mathbf{P}_{m+1,+} = (\mathbf{I} - \mathbf{K}_{m+1}\mathbf{H}_{m+1})\mathbf{P}_{m+1,-}, \tag{2.30}$$

where $\mathbf{K}_{m+1}$ is the Kalman gain matrix at step $m+1$, which weighs the relative importance of our predicted value against our observations, and is defined by

$$\mathbf{K}_{m+1} = \mathbf{P}_{m+1,-}\mathbf{H}_{m+1}^T(\mathbf{H}_{m+1}\mathbf{P}_{m+1,-}\mathbf{H}_{m+1}^T + \mathbf{R}^0)^{-1}. \tag{2.31}$$

The Kalman gain matrix is optimally chosen to minimize the trace of the posterior covariance matrix. We minimize with respect to $\text{tr}[\mathbf{P}_{m+1,+}]$ because the diagonal elements of $\mathbf{P}_{m+1,+}$ are the variances of the state components. Since we wish to minimize the uncertainty of the state, we want the variance of each state component to be as small as possible. Thus, the trace of $\mathbf{P}_{m+1,+}$ is an appropriate metric to use to quantify the state uncertainty.

2.2.2 Kalman-Bucy Filter

The Kalman filter described above is appropriate to discrete observations in time. Adapting the filter to continuous observations in time is straightforward and is referred to as the Kalman-Bucy filter [24]. By letting the time-increment in the sequential Kalman filter approach zero, we find the following ODEs which describe the evolution of the state estimate and covariance estimate, respectively

$$\dot{\hat{\mathbf{x}}}(t) = \mathcal{F}(t)\hat{\mathbf{x}}(t) + \mathbf{K}(t)\big(\mathbf{z}(t) - \mathbf{H}(t)\hat{\mathbf{x}}(t)\big), \tag{2.32}$$

$$\dot{\mathbf{P}}(t) = \mathcal{F}(t)\mathbf{P}(t) + \mathbf{P}(t)\mathcal{F}(t)^T + \mathbf{Q}(t) - \mathbf{K}(t)\mathbf{R}^0(t)\mathbf{K}(t)^T. \tag{2.33}$$

Note here that the continuous-time Kalman gain matrix now takes the form

$$\mathbf{K}(t) = \mathbf{P}(t)\mathbf{H}(t)^T\mathbf{R}^0(t)^{-1}. \tag{2.34}$$

2.2.3 Extended Kalman Filter

While the sequential Kalman filter and Kalman-Bucy filter described in Sections 2.2.1 and 2.2.2 are optimal for linear problems, they fail in the presence of nonlinearities, even when adapted in obvious ways. This is due to the fact that a Gaussian

distribution evolving under a nonlinear system will not remain Gaussian [46]. In cases where the nonlinearity is weak, the evolving distribution can often still be approximated as a Gaussian and we can use what is known as the Extended Kalman filter (EKF). The EKF is formulated by linearizing about the mean state, at both the forecast step and the analysis step, as necessary. For example, replacing $\mathcal{F}\hat{\mathbf{x}}$ and $\mathbf{H}\hat{\mathbf{x}}$ by nonlinear operators $\mathbf{f}(\hat{\mathbf{x}})$ and $\mathbf{h}(\hat{\mathbf{x}})$, respectively, requires that $\mathcal{F}$ and $\mathbf{H}$ in Equations (2.27), (2.30), (2.31), (2.33), and (2.34) be replaced by the Jacobian matrices $\frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}}\Big|_{\hat{\mathbf{x}}}$ and $\frac{\partial \mathbf{h}}{\partial \hat{\mathbf{x}}}\Big|_{\hat{\mathbf{x}}}$.

2.2.4 Alternative Methods

While this work relies mainly on the Extended Kalman filter described above, it is important to note that a variety of other data assimilation methods exists, and could also be used. The purpose of this section is to briefly explain some of these alternative methods.

Unscented Kalman Filter The Unscented Kalman filter, or UKF, is another variant of Kalman filtering which often yields superior results to the EKF, particularly for highly-nonlinear systems. The idea behind this method is to evolve a specifically chosen set of points (known as sigma points) according to their nonlinear model, which yields a "cloud" of transformed points. These transformed points can in turn be used to determine the evolved statistics of the problem [22]. It is important to note that while the UKF provides better results in general, it still relies on the Gaussian assumption that underlies all Kalman filtering methods.

Markov chain Monte Carlo Methods Unlike the Kalman filter, Markov chain Monte Carlo (MCMC) methods do not make any prior assumptions on the distribution of the state. Rather, these methods generate a sequence of samples by simulating a Markov chain, which are then used to approximate the distribution, as

well as the sample mean [18]. While MCMC methods can provide more detailed information regarding the state compared to KF methods, they require a large set of samples to do so. This can therefore result in a high degree of computational complexity to run these types of methods.

CHAPTER 3

COVARIANCE-TRACKING

The focus of this section is to use the level set method, outlined in Section 2.1, in conjunction with the Extended Kalman filter, described in Section 2.2, to reduce a measure of the uncertainty of an unknown velocity field $\mathbf{v}(\mathbf{x}, t)$ using a single autonomous vehicle.

3.1 Problem Statement

Consider an autonomous surface vehicle, initially located at the point $\mathbf{x}_s \in D \subseteq \mathbb{R}^2$ at time t_0. Assume that D represents a sufficiently large oceanic domain (i.e., large enough to approximate the vehicle as a point source), and has an uncertain flow modeled by $\mathbf{v}(\mathbf{x}, t)$, which is assumed to be C^1. It is further assumed that the model $\mathbf{v}(\mathbf{x}, t)$ is robust enough that any model mismatch can be attributed to a noise term.

This vehicle travels with a constant speed of F with respect to $\mathbf{v}$ and is constrained by a finite deployment time, $t_0 \leq t \leq t_f$. As this vehicle travels, it continuously takes direct and noisy measurements of the ocean currents, and uses this information to update its estimate of $\mathbf{v}(\mathbf{x}, t)$.

It is assumed here that $F > 0$, and without loss of generality, it is expected that there will be regions where $|\mathbf{v}| \geq F$. With this in mind, we seek controls which will exploit the underlying ocean dynamics, and allow the vehicle to traverse as much of the domain as possible. To accomplish this goal, we restrict the vehicle to follow trajectories which satisfy an objective of time-minimization. Therefore, the optimal control we seek is the time-minimizing path which steers this autonomous vehicle to a point of minimum uncertainty at time t_f.

3.2 Covariance-Tracking Algorithm

Since it is assumed here that observations are being made continually in time, the Kalman-Bucy filter (see Section 2.2.2), or the Extended Kalman-Bucy filter, is used for state estimation and uncertainty quantification. Specifically, the trace of the expected covariance matrix, which evolves according to Equation (2.33), will be used here as our metric of uncertainty.

Conceptually, the idea behind this section is to determine a family of controls for the vehicle via the level set method, and evolve the expected covariance matrix along each of these controls from time t_0 until time t_f. Upon reaching the final time, we calculate the trace of each of these expected covariance matrices, providing an approximate measure of the uncertainty at the end of deployment. Having this knowledge, we have the vehicle follow the control which culminates in the lowest expected uncertainty. By picking the covariance-minimizing time-optimal path, our hope is to find a good approximation for a true-covariance minimizing path which will not get caught in ineffective local minima

While under ideal circumstances, the optimal control would be updated continuously as new observations are made, this would be computationally impractical for any sort of real-time application. As such, we will assume that there will be M path updates at equally spaced time intervals $t = t_0 + \frac{(m-1)}{M}\big(t_f - t_0\big)$, for $m = 1, 2, \ldots, M$. These updates will allow us to use the newly-gathered information from the AV to revise our understanding of $\mathbf{v}(\mathbf{x}, t)$ and improve the heading for our vehicle, while remaining computationally tractable.

Finally, before explaining the algorithm, we require a parametric representation of $\mathbf{v}$, which in general will depend on $\mathbf{x}$, t, and $\hat{\mathbf{x}}$ (position, time, and our state-vector). Since determining an accurate inference model is not a straightforward procedure (see Chapter 5), for the purposes of this section we will assume that the exact functional form for $\mathbf{v}(\mathbf{x}, t; \hat{\mathbf{x}})$ is known beforehand.

Once a representation for $\mathbf{v}(\mathbf{x}, t; \hat{\mathbf{x}})$ is chosen, the first step of the algorithm is to formulate an initial estimate $\mathbf{v}_0(\mathbf{x}, t)$ for the velocity subject to the initial estimate of the state $\hat{\mathbf{x}}_0$ (i.e., $\mathbf{v}_0(\mathbf{x}, t) = \mathbf{v}(\mathbf{x}, t; \hat{\mathbf{x}}_0)$). Then, starting from the initial position $\mathbf{x}_s$ at $t = t_0$, we run the level set algorithm out until time t_f. A set of points along the final reachable set $S(t_f)$ is then chosen, and the backtracking algorithm discussed in Section 2.1.3 is utilized to determine a family of possible controls for the vehicle. Note here that the points along $S(t_f)$ are chosen to be equidistant, and to provide a thorough sampling of the contour.

Having a family of controls for the AV, the covariance matrix is evolved along each of the extremal trajectories from t_0 until time t_f using Equation (2.33). From here, the trace of each expected covariance matrix is found, providing an approximation for the uncertainty at time t_f. We can now determine which control results in the lowest expected value for the $\text{tr}[\mathbf{P}(t_f)]$, and choose that to be the optimal control for our AV, which we will call $\theta_{opt}(t)$. Note that is necessary to have an initial estimate for the covariance matrix $\mathbf{P}(t_0)$, as well as for the matrices $\mathbf{Q}(t)$ and $\mathbf{R}^0(t)$.

Having our optimal control $\theta_{opt}(t)$, we allow our vehicle to run subject to the dynamics given by Equation (2.2). The AV will follow the heading dictated by this control (although not necessarily the desired course given uncertainty in the velocity field), taking noisy measurements, $\mathbf{z}(t)$, of the true velocity. As new measurements are taken, we update $\hat{\mathbf{x}}(t)$ using Equation (2.32), giving us a more accurate estimate of the velocity $\mathbf{v}(\mathbf{x}, t; \hat{\mathbf{x}})$.

The vehicle will continue to follow along $\theta_{opt}(t)$, until it reaches a pre-determined path update time at $t = t_0 + \frac{(m-1)}{M}\big(t_f - t_0\big)$, for $m = 2, 3 \ldots, M$. At these points we will repeat the algorithm as follows:

1. Starting from the current position of the vehicle, run the level set method from time $t = t_0 + \frac{(m-1)}{M}\big(t_f - t_0\big)$, until time t_f, making use of the current estimate of the velocity $\mathbf{v}(\mathbf{x}, t; \hat{\mathbf{x}})$.

2. Determine a set of points along the final reachable set, $S(t_f)$, and use the backtracking algorithm to determine a family of possible controls for the vehicle.

3. Evolve the covariance matrix along each of the extremal trajectories from time $t_0 + \frac{(m-1)}{M}\big(t_f - t_0\big)$ until time t_f.

4. Determine which control culminates in the smallest expected value for $\text{tr}[\mathbf{P}(t_f)]$, and set this as the new optimal control, $\theta_{opt}(t)$.

5. Allow the AV to move with control $\theta_{opt}(t)$ subject to the dynamics of Equation (2.2). As new measurements are taken, update $\hat{\mathbf{x}}(t)$ with Equation (2.32), as well the estimate of the velocity $\mathbf{v}(\mathbf{x}, t; \hat{\mathbf{x}})$.

6. Repeat at the next path update time.

While the aforementioned algorithm is used throughout this work, it is important to note that many of the choices made while defining this algorithm come down to a matter of preference. For instance, there is no requirement to run the algorithm out until time t_f. While we have chosen to do this, all that is truly required is that the algorithm is run until at least the next path update. Additionally, while we have chosen to have path updates which are equally spaced in time, this is not required either. Path updates can be made at any desired time, so long as the vehicle has a control defined for all $t_0 \leq t \leq t_f$.

3.3 Time-Independent Systems

For state models which are independent of time, and as such assume no system noise, Equation (2.33) simplifies to

$$\frac{d}{dt}\mathbf{P}(t) = -\mathbf{P}(t)\mathbf{H}(t)^T\mathbf{R}^0(t)^{-1}\mathbf{H}(t)\mathbf{P}(t). \tag{3.1}$$

Equation (3.1) is a matrix Bernoulli equation, and can be simplified further by looking at the inverse matrix $\mathbf{P}(t)^{-1}$. Differentiating with respect to time yields

$$\begin{aligned}\frac{d}{dt}\big(\mathbf{P}(t)^{-1}\big) &= -\mathbf{P}(t)^{-1}\left(\frac{d}{dt}\mathbf{P}(t)\right)\mathbf{P}(t)^{-1}, \\ &= \mathbf{H}(t)^T\mathbf{R}^0(t)^{-1}\mathbf{H}(t).\end{aligned} \tag{3.2}$$

Integrating and inverting gives us

$$\mathbf{P}(t) = \left(\mathbf{P}(t_0)^{-1} + \int_{t_0}^{t}\mathbf{H}(s)^T\mathbf{R}^0(s)^{-1}\mathbf{H}(s)ds\right)^{-1}. \tag{3.3}$$

Since $\mathbf{H}(t)$ is known, and we assume knowledge of $\mathbf{R}^0(t)$ and $\mathbf{P}(t_0)$, we can solve for $\mathrm{tr}[\mathbf{P}(t)]$ using Equation (3.3). For this work, we assume that $\mathbf{P}(t_0) = \mathbf{I}$ and $\mathbf{R}^0(t)$ is positive definite, and thus we can use Equation (3.3) and the Woodbury formula [16] to show that

$$\mathrm{tr}[\mathbf{P}(t)] = \sum_{i=1}^{N}\frac{1}{1+\lambda_i}, \tag{3.4}$$

where N is the number of state components, and λ_i is the i^{th} eigenvalue of $\int_{t_0}^{t}\mathbf{H}(s)^T\mathbf{R}^0(s)^{-1}\mathbf{H}(s)ds$.

3.3.1 Example 1

As a first example, consider a linear center of the form

$$\mathbf{v} = \begin{bmatrix}\alpha(y-y_0)\\ -\alpha(x-x_0)\end{bmatrix}, \tag{3.5}$$

where we are interested in reducing the uncertainty of

$$\hat{\mathbf{x}} = \begin{bmatrix}\alpha\\ x_0\\ y_0\end{bmatrix}. \tag{3.6}$$

The equations of motion for our AV are

$$\dot{\mathbf{x}} = \begin{bmatrix} \alpha(y - y_0) + F\cos(\theta) \\ -\alpha(x - x_0) + F\sin(\theta) \end{bmatrix}, \tag{3.7}$$

which, for $t_0 = 0$, has solution

$$x(t) = x_0 + Ft\cos(\theta_0 - \alpha t) + (y_s - y_0)\sin(\alpha t) + (x_s - x_0)\cos(\alpha t), \tag{3.8}$$

$$y(t) = y_0 + Ft\sin(\theta_0 - \alpha t) - (x_s - x_0)\sin(\alpha t) + (y_s - y_0)\cos(\alpha t). \tag{3.9}$$

Since $\mathbf{v}$ is non-linear with respect to the state components, we must use the Extended version of the Kalman-Bucy filter, giving a linearized observation operator of

$$\mathbf{H}(t) = \frac{\partial \mathbf{h}(\hat{\mathbf{x}})}{\partial \hat{\mathbf{x}}} = \begin{bmatrix} y - y_0 & 0 & -\alpha \\ -x + x_0 & \alpha & 0 \end{bmatrix}. \tag{3.10}$$

For this problem, we will assume that

$$\mathbf{R}^0 = \sigma^2\mathbf{I} \qquad \text{and} \qquad \mathbf{P}(0) = \mathbf{I}.$$

Plugging into Equation (3.1), we find that

$$\frac{d\mathbf{P}}{dt} = -\frac{1}{\sigma^2}\mathbf{P}\begin{bmatrix} (y - y_0)^2 + (x - x_0)^2 & -\alpha(x - x_0) & -\alpha(y - y_0) \\ -\alpha(x - x_0) & \alpha^2 & 0 \\ -\alpha(y - y_0) & 0 & \alpha^2 \end{bmatrix}\mathbf{P}. \tag{3.11}$$

Using Equation (3.3), we find that

$$\mathbf{P}(t) = \tilde{\mathbf{P}}(t)^{-1}, \tag{3.12}$$

where

$$\tilde{\mathbf{P}}(t) = \mathbf{I} + \frac{1}{\sigma^2}\int_0^t \mathbf{H}(s)^T\mathbf{H}(s)ds, \tag{3.13}$$

which has components

$$\tilde{P}_{11} = \frac{F^2 t^3}{3\sigma^2} + \frac{Ft^2}{\sigma^2}\Big[(x_s - x_0)\cos(\theta_0) + (y_s - y_0)\sin(\theta_0)\Big] + \frac{t}{\sigma^2}\Big[(x_s - x_0)^2 + (y_s - y_0)^2\Big] + 1,$$

$$\tilde{P}_{12} = \tilde{P}_{21} = \frac{Ft\sin(\theta_0 - \alpha t)}{\sigma^2} - \frac{F\big(\cos(\theta_0 - \alpha t) - \cos(\theta_0)\big)}{\alpha\sigma^2} + \frac{\big(y_s - y_0\big)\big(\cos(\alpha t) - 1\big)}{\sigma^2} - \frac{\big(x_s - x_0\big)\sin(\alpha t)}{\sigma^2},$$

$$\tilde{P}_{13} = \tilde{P}_{31} = \frac{-Ft\cos(\theta_0 - \alpha t)}{\sigma^2} - \frac{F\big(\sin(\theta_0 - \alpha t) - \sin(\theta_0)\big)}{\alpha\sigma^2} - \frac{\big(x_s - x_0\big)\big(\cos(\alpha t) - 1\big)}{\sigma^2} - \frac{\big(y_s - y_0\big)\sin(\alpha t)}{\sigma^2},$$

$$\tilde{P}_{22} = \tilde{P}_{33} = \frac{\alpha^2 t}{\sigma^2} + 1,$$

$$\tilde{P}_{23} = \tilde{P}_{32} = 0.$$

Inverting $\tilde{\mathbf{P}}(t)$ and simplifying, we find that

$$\mathrm{tr}[\mathbf{P}(t)] = \frac{1}{\tilde{P}_{22}} + \frac{\tilde{P}_{11} + \tilde{P}_{22}}{\tilde{P}_{11}\tilde{P}_{22} - \tilde{P}_{12}^2 - \tilde{P}_{13}^2}. \tag{3.14}$$

In order to find the optimal solution for this problem, the following parameters are chosen

$$F = 1,$$

$$\sigma^2 = 0.1,$$

$$(x_s, y_s) = (0, 0),$$

$$t_f = 3,$$

and

$$\hat{\mathbf{x}}_0 = \hat{\mathbf{x}}_{true} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

By symbolically differentiating Equation (3.14) with respect to θ_0 in MATLAB, and numerically solving $\frac{d}{d\theta_0}(\mathrm{tr}[\mathbf{P}(t_f)]) = 0$, we find that a minimum of the expected covariance trace occurs with an initial control angle of $\theta_0 \approx 2.01426$. Figure 3.1 shows a comparison of the true evolution of $\mathrm{tr}[\mathbf{P}(t)]$, to the numerical approximation. Note that while it is only necessary to find the expected covariance trace at time t_f,

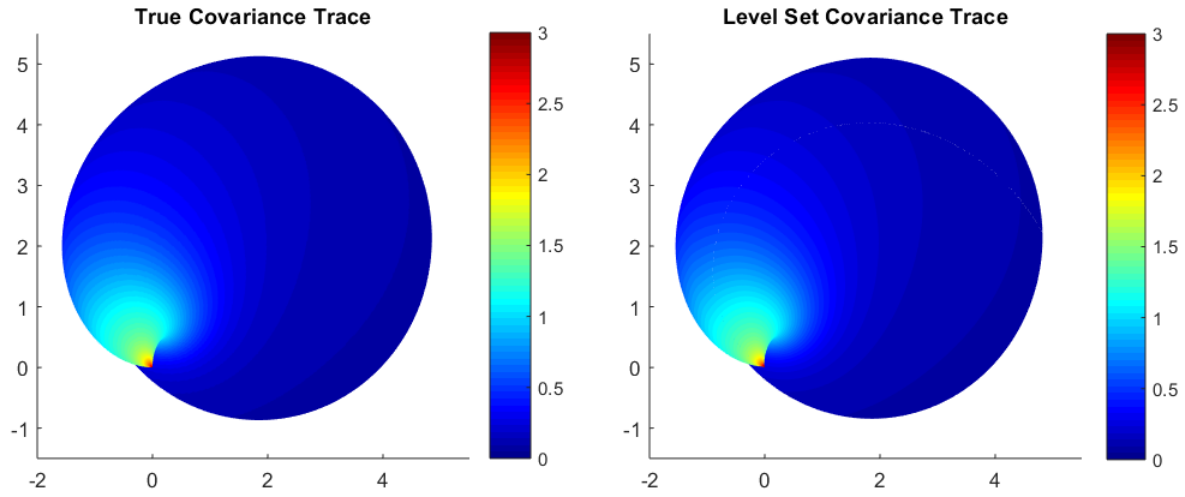

Figure 3.1 Evolution of the true covariance trace (left) and the numerical approximation (right) for Example 1.

we calculated it at each time step in order to show the accuracy of our numerical approximation.

Figure 3.2 shows a comparison of the true optimal trajectory to that found using our covariance-tracking algorithm. We can see that this algorithm very accurately reproduces the optimal solution, and moreover, we find first-order convergence in space, as we expect with the level set method.

3.3.2 Example 2

As a second example, consider a velocity of the form

$$\mathbf{v} = \sum_{i=1}^{4} \frac{A\beta}{r_i} e^{-\beta r_i} \begin{bmatrix} (x - x_{c_i}) - (-1)^i B(y - y_{c_i}) \\ (y - y_{c_i}) + (-1)^i B(x - x_{c_i}) \end{bmatrix}, \tag{3.15}$$

confined to the domain $[0, 3] \times [0, 3]$, with

$$(x_{c_1}, y_{c_1}) = (1, 1),$$

$$(x_{c_2}, y_{c_2}) = (1, 2),$$

$$(x_{c_3}, y_{c_3}) = (2, 2),$$

$$(x_{c_4}, y_{c_4}) = (2, 1),$$

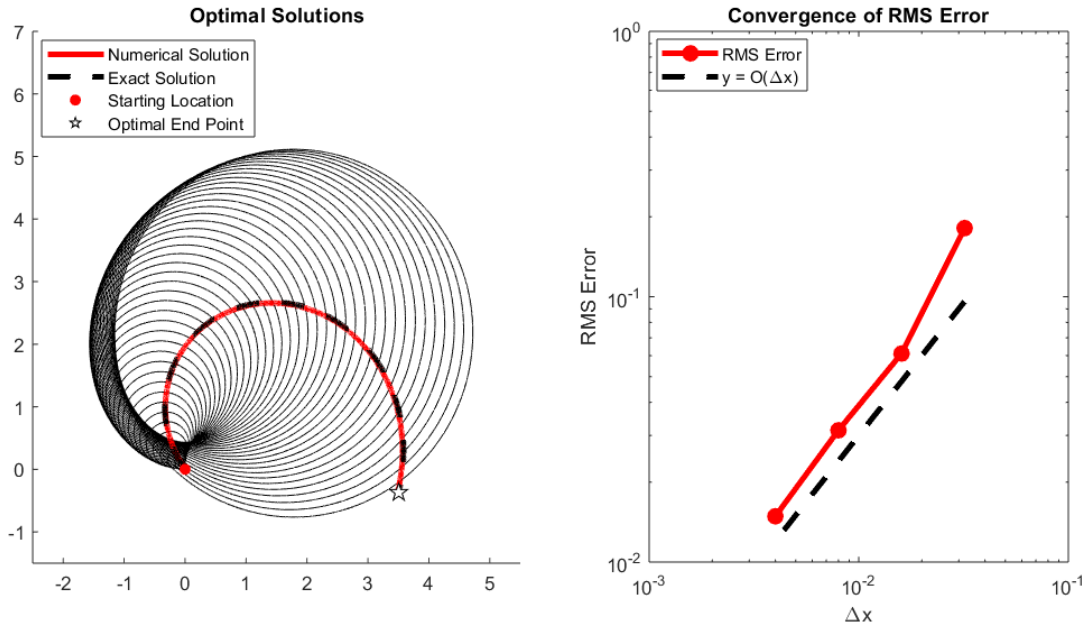

Figure 3.2 Left: Reachable set and comparison of optimal paths. Right: Convergence of numerical solution to exact solution.

and

$$r_i = \sqrt{(x - x_{c_i})^2 + (y - y_{c_i})^2}.$$

For this problem, we are interested in reducing the uncertainty of

$$\hat{\mathbf{x}} = \begin{bmatrix} A \\ \beta \\ B \end{bmatrix}, \tag{3.16}$$

where

$$\hat{\mathbf{x}}_{true} = \begin{bmatrix} 0.0244 \\ 3 \\ 0.6 \end{bmatrix}, \tag{3.17}$$

subject to the following parameters

$$\mathbf{P}(t_0) = \mathbf{I},$$

$$\mathbf{R}^0 = \sigma^2\mathbf{I},$$

$$\sigma^2 = 0.01,$$

$$F = 0.04,$$

$$(x_s, y_s) = (1.5, 0.5),$$

$$t_0 = 0,$$

$$t_f = 90,$$

and

$$\hat{\mathbf{x}}_0 = \begin{bmatrix} 0.05 \\ 2.5 \\ 1 \end{bmatrix}.$$

We have chosen 18 path updates (i.e., $M = 18$), which means the control is updated every 5 time-units. Furthermore, the trajectory found using the covariance-tracking methodology is compared to a gradient-defined control. Note that neither control is capable of using new information until the next path update. The trajectories corresponding to each control are shown in Figure 3.3, whereas the error reductions for the two controls can be found in Figure 3.4.

From Figure 3.3, we see that both controls lead the AV to the same gyre. However, using the covariance-tracking methodology, the AV is brought to the gyre faster, and as such, reduces the uncertainty faster. This is reflected in both the RMS error, as well as the covariance trace (see Figure 3.4). Although the RMS errors for both controls are ultimately about the same, the better rate of convergence for the level set control is exactly what we would expect, and verifies that this method works as intended.

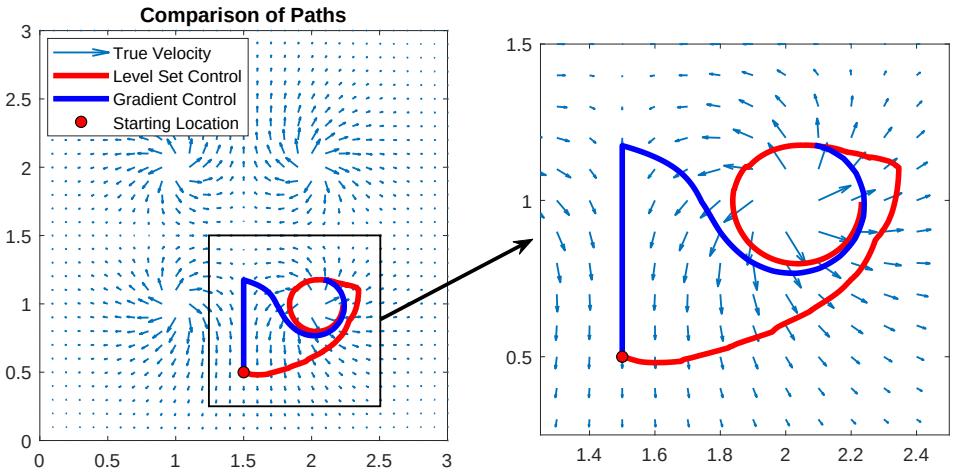

Figure 3.3 Comparison of level set control to gradient-defined control.

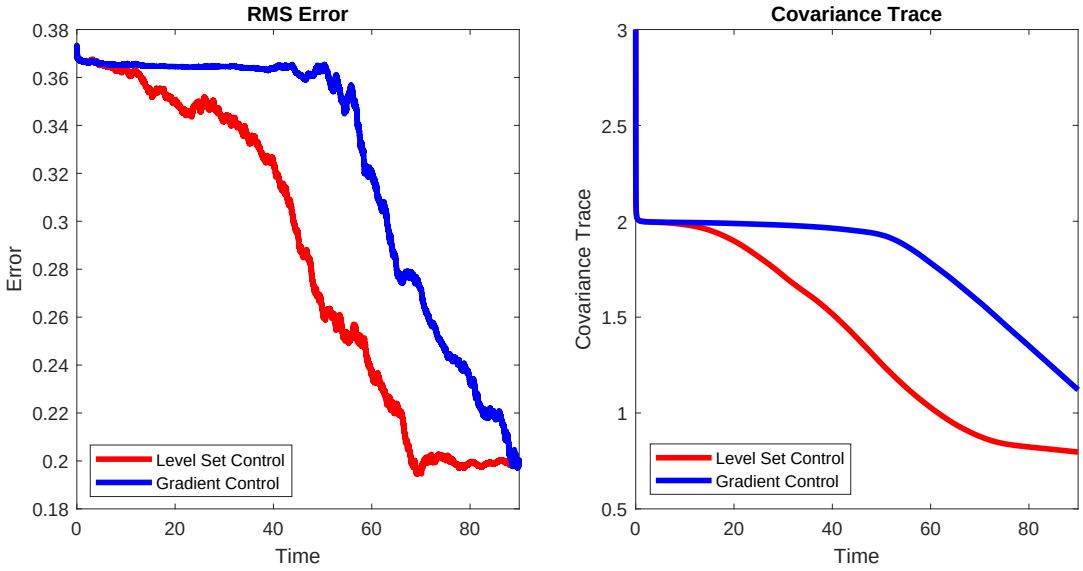

Figure 3.4 Left: Convergence of the RMS errors. Right: Convergence of the covariance trace.

3.4 Time-Dependent Systems

For problems with time-dependent states, and problems where system noise is present, it becomes more computationally expensive to calculate the expected covariance trace within the reachable set. This is due to the fact that Equation (2.33) is no longer separable, and cannot be simplified or solved explicitly. It is, therefore, necessary to

compute each component of $\mathbf{P}(t)$ at each time step using a finite-difference scheme, which subjects the solution to CFL conditions and corresponding stability issues.

CHAPTER 4

COORDINATED CONTROL OF MULTIPLE VEHICLES

In this chapter, we extend the covariance-tracking methodology of Chapter 3 to find optimal paths for fleets of autonomous vehicles. To do this, we make use of an iterative procedure which perturbs the optimal trajectory of each AV into a minimum of the expected covariance trace.

4.1 Problem Statement

Consider a fleet of K autonomous surface vehicles initially located at the points $\mathbf{x}_s^k \in D \subseteq \mathbb{R}^2$, $k = 1, 2, \ldots, K$, at time t_0. Assume that these AVs are within a body of water with surface current of the form $\mathbf{v}(\mathbf{x}, t; \hat{\mathbf{x}})$, where $\hat{\mathbf{x}}$ is a vector of uncertain parameters which define $\mathbf{v}$. These vehicles travel with constant speeds of F^k with respect to $\mathbf{v}$ and are constrained by a finite deployment time, $t_0 \leq t \leq t_f$. As in Chapter 3, we assume that each vehicle is continually taking noisy measurements of the current, and therefore make use of the Kalman-Bucy filter for state estimation and uncertainty quantification. We are seeking optimal controls $\theta_{opt}^k(t)$, $t_0 \leq t \leq t_f$, for each member of the fleet such that the uncertainty of $\hat{\mathbf{x}}$ is minimized at the end of deployment t_f.

For multiple vehicles, we can no longer minimize the uncertainty of $\hat{\mathbf{x}}$ by looking at each vehicle independently. This is due to the fact that the trajectory of each AV is now coupled through the observation operator, $\mathbf{H}(t)$, and as such, is also coupled through the expected covariance, $\mathbf{P}(t)$, as well. Thus, our optimal solutions will be time-minimizing paths which *collectively* minimize $\mathrm{tr}[\mathbf{P}(t_f)]$.

4.2 Finding Optimal Trajectories

As explained in Chapter 3, we begin by finding the reachable set $S^k(t)$ for each of the K autonomous vehicles, as well as a set of paths for each AV which adequately samples $S^k(t_f)$. Having this information, the initial trajectories, $\mathbf{x}_0^k$, are chosen for $2 \leq k \leq K$. The algorithm then proceeds as follows for AVs $1 \leq k \leq K$, and iteration number $1 \leq n \leq N$:

- For $k = 1$, set $\mathbf{x}_{n-1}^2, \mathbf{x}_{n-1}^3, \ldots, \mathbf{x}_{n-1}^K$ to be fixed, and find the path $\boldsymbol{\xi}$ which minimizes $\text{tr}[\mathbf{P}\big(\boldsymbol{\xi}, \mathbf{x}_{n-1}^2, \mathbf{x}_{n-1}^3, \ldots, \mathbf{x}_{n-1}^K, t_f\big)]$. Set $\mathbf{x}_n^1 = \boldsymbol{\xi}$.

- For $2 \leq k \leq K-1$, set $\mathbf{x}_n^1, \ldots, \mathbf{x}_n^{k-1}, \mathbf{x}_{n-1}^{k+1}, \ldots, \mathbf{x}_{n-1}^K$ to be fixed, and find the path $\boldsymbol{\xi}$ which minimizes $\text{tr}[\mathbf{P}\big(\mathbf{x}_n^1, \ldots, \mathbf{x}_n^{k-1}, \boldsymbol{\xi}, \mathbf{x}_{n-1}^{k+1}, \ldots, \mathbf{x}_{n-1}^K, t_f\big)]$. Set $\mathbf{x}_n^k = \boldsymbol{\xi}$.

- For $k = K$, set $\mathbf{x}_n^1, \ldots, \mathbf{x}_n^{K-1}$ to be fixed, and find the path $\boldsymbol{\xi}$ which minimizes $\text{tr}[\mathbf{P}\big(\mathbf{x}_n^1, \ldots, \mathbf{x}_n^{K-1}, \boldsymbol{\xi}, t_f\big)]$. Set $\mathbf{x}_n^K = \boldsymbol{\xi}$.

- If $n = N$, stop algorithm. Otherwise, $n \to n+1$, repeat steps.

4.3 Convergence of Optimal Trajectories

To prove this iterative technique works, we assume that we have already found a minimizing set of controls, and perturb them away from that minimum. We show that as we iterate through the optimal paths of the AVs, this perturbation becomes smaller and ultimately tends to zero.

We first note that for time-minimizing trajectories, it can be shown from the solution of Zermelo's navigation problem, that for $\mathbf{v}(\mathbf{x}, t) \in C^1$, $\mathbf{x}_{opt}^k$ is defined uniquely by it's initial control heading θ_0^k. Thus, perturbations made to an entire optimizing trajectory can be thought of as a perturbation to the control's initial heading.

Now, consider K autonomous vehicles and their corresponding optimal trajectories, $\mathbf{x}_{opt}^k(t; \theta_0^k)$, $k = 1, 2, \ldots, K$. By our definition of optimal, these trajectories are

such that $\text{tr}[\mathbf{P}(t_f)] = f(\boldsymbol{\theta}_0)$ is minimized, where

$$\boldsymbol{\theta}_0 = \begin{bmatrix} \theta_0^1 \\ \theta_0^2 \\ \vdots \\ \theta_0^K \end{bmatrix}, \tag{4.1}$$

are the initial headings for each AV. As these trajectories correspond to a minimum, we know that

$$f_{\boldsymbol{\theta}_0} = \mathbf{0}, \text{ and} \tag{4.2}$$

$$f_{\boldsymbol{\theta}_0\boldsymbol{\theta}_0} > \mathbf{0}, \tag{4.3}$$

where $f_{\boldsymbol{\theta}_0}$ and $f_{\boldsymbol{\theta}_0\boldsymbol{\theta}_0}$ are the gradient and Hessian of f with respect to $\boldsymbol{\theta}_0$, respectively. We perturb our optimal control vector $\boldsymbol{\theta}_0$ by vector $\boldsymbol{\delta}_0$, and aim to show that as we iterate through the vehicles, as was explained in Section 4.2, this perturbation term will tend to zero, forcing our controls back into the minimum.

Following the algorithm in Section 4.2, we begin by initializing the paths for vehicles $2 \leq k \leq K$, which are defined here by the initial headings $\theta_0^k + \delta_0^k$. Beginning the first iteration, we search for the optimal control for vehicle 1, defined by the initial heading $\theta_0^1 + \delta_1^1$. Since θ_0^1 is fixed, and $\theta_0^1 + \delta_1^1$ is a local minimizer, we seek the value δ_1^1 such that

$$f_{\theta_0^1 + \delta_1^1} = 0. \tag{4.4}$$

To accomplish this, we first assume that each $\delta_0^k \ll 1$. Under this assumption we can linearize each component of our gradient $f_{\boldsymbol{\theta}_0 + \boldsymbol{\delta}_0}$ to yield

$$f_{\theta_0^k + \delta_0^k} \approx f_{\theta_0^k} + \sum_{j=1}^{K} f_{\theta_0^k \theta_0^j} \delta_0^j = \sum_{j=1}^{K} f_{\theta_0^k \theta_0^j} \delta_0^j. \tag{4.5}$$

Combining Equations (4.4) and (4.5), we find that the optimal value for δ_1^1 is given by the equation

$$f_{\theta_0^1 \theta_0^1} \delta_1^1 = -\sum_{j=2}^{K} f_{\theta_0^1 \theta_0^j} \delta_0^j. \tag{4.6}$$

Moving onto vehicle 2, we now seek the value for δ_1^2 such that

$$f_{\theta_0^2+\delta_1^2} = 0. \tag{4.7}$$

Again making use of Equations (4.4) and (4.5), we have that the optimal value for δ_1^2 is given by the equation

$$f_{\theta_0^2\theta_0^1}\delta_1^1 + f_{\theta_0^2\theta_0^2}\delta_1^2 = -\sum_{j=3}^{K} f_{\theta_0^2\theta_0^j}\delta_0^j. \tag{4.8}$$

Continuing this for all vehicles $1 \leq k \leq K$, we see that, in general, the optimal value for δ_1^k is given by the formula

$$\sum_{j=1}^{k-1} f_{\theta_0^k\theta_0^j}\delta_1^j + f_{\theta_0^k\theta_0^k}\delta_1^k = -\sum_{j=k+1}^{K} f_{\theta_0^k\theta_0^j}\delta_0^j. \tag{4.9}$$

Using the fact that $f_{\theta_0^i\theta_0^j} = f_{\theta_0^j\theta_0^i}$, we can write this in matrix form as

$$\big(\mathbf{D}+\mathbf{L}\big)\boldsymbol{\delta}_1 = -\mathbf{L}^T\boldsymbol{\delta}_0, \tag{4.10}$$

or

$$\boldsymbol{\delta}_1 = -\big(\mathbf{D}+\mathbf{L}\big)^{-1}\mathbf{L}^T\boldsymbol{\delta}_0, \tag{4.11}$$

where

$$\mathbf{D} = \begin{bmatrix} f_{\theta_0^1\theta_0^1} & 0 & 0 & \dots & 0 \\ 0 & f_{\theta_0^2\theta_0^2} & 0 & \dots & 0 \\ 0 & 0 & f_{\theta_0^3\theta_0^3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & f_{\theta_0^K\theta_0^K} \end{bmatrix}, \tag{4.12}$$

and

$$\mathbf{L} = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ f_{\theta_0^1\theta_0^2} & 0 & \dots & 0 & 0 \\ f_{\theta_0^1\theta_0^3} & f_{\theta_0^2\theta_0^3} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ f_{\theta_0^1\theta_0^K} & f_{\theta_0^2\theta_0^K} & \dots & f_{\theta_0^{K-1}\theta_0^K} & 0 \end{bmatrix}. \tag{4.13}$$

From Equation (4.11), it is easy to see that extending this process over multiple iterations will give optimal controls defined by $\boldsymbol{\theta}_0 + \boldsymbol{\delta}_n$, where

$$\boldsymbol{\delta}_n = \Big[-\big(\mathbf{D} + \mathbf{L}\big)^{-1}\mathbf{L}^T \Big]^n \boldsymbol{\delta}_0, \tag{4.14}$$

and n is the iteration number.

For this approach to converge back to the minimum, it must be the case that Equation (4.14) is a convergent operator. That is,

$$\lim_{n\to\infty} \boldsymbol{\delta}_n \to \mathbf{0}. \tag{4.15}$$

This can be shown by first noting that Equations (4.12) and (4.13) consist only of the components of the Hessian matrix of $f(\boldsymbol{\theta}_0)$. That is,

$$f_{\boldsymbol{\theta}_0\boldsymbol{\theta}_0} = \mathbf{L} + \mathbf{D} + \mathbf{L}^T. \tag{4.16}$$

With this in mind, it is clear to see that Equation (4.14) is the Gauss-Seidel iteration for solving $f_{\boldsymbol{\theta}_0\boldsymbol{\theta}_0}\boldsymbol{\delta} = \mathbf{0}$. Since $f_{\boldsymbol{\theta}_0\boldsymbol{\theta}_0}$ is symmetric positive-definite, it is a known result [13] that the Gauss-Seidel iteration is convergent, and as such, so is this iterative technique. It has therefore been demonstrated that the algorithm explained in Section 4.2 will yield optimal controls which minimize $\mathrm{tr}[\mathbf{P}(t_f)]$, assuming such a minimum exists.

4.4 Examples

4.4.1 Example 1

Consider the case of two autonomous vehicles within a body of water with a surface current of the form

$$\mathbf{v} = \begin{bmatrix} \alpha(y-\beta) \\ -\alpha(x-\beta) \end{bmatrix}. \tag{4.17}$$

Here, we wish to minimize the uncertainty of

$$\hat{\mathbf{x}} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{4.18}$$

using noisy observations taken by the two AVs. Since $\mathbf{v}$ is non-linear with respect to the state components, we must use the Extended version of the Kalman-Bucy filter, giving a linearized observation operator of

$$\mathbf{H}(t) = \frac{\partial \mathbf{h}(\hat{\mathbf{x}})}{\partial \hat{\mathbf{x}}} = \begin{bmatrix} y^1 - \beta & -\alpha \\ -(x^1 - \beta) & \alpha \\ y^2 - \beta & -\alpha \\ -(x^2 - \beta) & \alpha \end{bmatrix}, \tag{4.19}$$

where y^i and x^i define the trajectory of the i^{th} vehicle. These trajectories are found from the equations of motion

$$\dot{\mathbf{x}}^i = \begin{bmatrix} \alpha(y^i - \beta) + F^i \cos(\theta^i) \\ -\alpha(x^i - \beta) + F^i \sin(\theta^i) \end{bmatrix}, \tag{4.20}$$

which, for $t_0 = 0$, has solution

$$x^i = F^i t \cos(\theta_0^i - \alpha t) + (y_s^i - \beta)\sin(\alpha t) + (x_s^i - \beta)\cos(\alpha t) + \beta, \tag{4.21}$$

$$y^i = F^i t \sin(\theta_0^i - \alpha t) - (x_s^i - \beta)\sin(\alpha t) + (y_s^i - \beta)\cos(\alpha t) + \beta. \tag{4.22}$$

Note that (x_s^i, y_s^i) is the initial position of the i^{th} AV, and θ_0^i is the initial heading. Assuming that

$$\mathbf{R}^0 = \sigma^2 \mathbf{I} \qquad \text{and} \qquad \mathbf{P}(t_0) = \mathbf{I},$$

plugging into Equation (3.3) and solving, we find the expected covariance trace as a function of θ_0^1 and θ_0^2

$$\text{tr}[\mathbf{P}(t)] = \frac{\sigma^2\big(G(\theta_0^1) + G(\theta_0^2) + C\big)}{C\big(G(\theta_0^1) + G(\theta_0^2)\big) - \big(H(\theta_0^1) + H(\theta_0^2)\big)^2}, \tag{4.23}$$

where

$$C = 4\alpha^2 t + \sigma^2,$$

$$G(\theta_0^i) = \frac{(F^i)^2 t^3}{3} + F^i t^2\Big((x_s^i - \beta)\cos(\theta_0^i) + (y_s^i - \beta)\sin(\theta_0^i)\Big) + \Big((x_s^i - \beta)^2 + (y_s^i - \beta)^2\Big)t + \frac{\sigma^2}{2},$$

$$\begin{aligned} H(\theta_0^i) = &-\frac{F^i}{\alpha}\Big((1 - \alpha t)\sin(\theta_0^i - \alpha t) + (1 + \alpha t)\cos(\theta_0^i - \alpha t) - \sin(\theta_0^i) - \cos(\theta_0^i)\Big) \\ &- x_s^i\Big(\sin(\alpha t) + \cos(\alpha t)\Big) - y_s^i\Big(\sin(\alpha t) - \cos(\alpha t)\Big) + 2\beta\sin(\alpha t) + x_s^i - y_s^i. \end{aligned}$$

In order to find the optimal trajectories for this problem, the following parameters are chosen

$$(x_s^1, y_s^1) = (0, 1),$$
$$(x_s^2, y_s^2) = (1, 0),$$
$$F^1 = F^2 = 1,$$
$$\sigma^2 = 0.1,$$
$$t_0 = 0,$$
$$t_f = 5,$$

and

$$\hat{\mathbf{x}}_0 = \hat{\mathbf{x}}_{true} = \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix}.$$

After running the algorithm for 100 iterations, we find that the optimal initial headings for the two AVs are $\theta_0^1 \approx \pi$ and $\theta_0^2 \approx \frac{3\pi}{2}$. Therefore, the optimal trajectories are given approximately by

$$x_{opt}^1(t) = -(1+t)\cos\left(\frac{t}{2}\right) + 1,$$
$$y_{opt}^1(t) = (1+t)\sin\left(\frac{t}{2}\right) + 1,$$
$$x_{opt}^2(t) = -(1+t)\sin\left(\frac{t}{2}\right) + 1,$$
$$y_{opt}^2(t) = -(1+t)\cos\left(\frac{t}{2}\right) + 1.$$

To further demonstrate that these initial headings correspond to a minimum, Figure 4.1 shows the expected covariance trace at $t_f = 5$ for all initial headings. Again, we find that $\theta_0^1 \approx \pi$ and $\theta_0^2 \approx \frac{3\pi}{2}$ corresponds to the minimum of the expected covariance trace. Figure 4.2 shows a comparison of the numerical solutions obtained using the covariance-tracking algorithm, to the exact solutions $\mathbf{x}_{opt}^1$ and $\mathbf{x}_{opt}^2$. The right panel of Figure 4.2 shows the RMS error convergence for the two AV trajectories

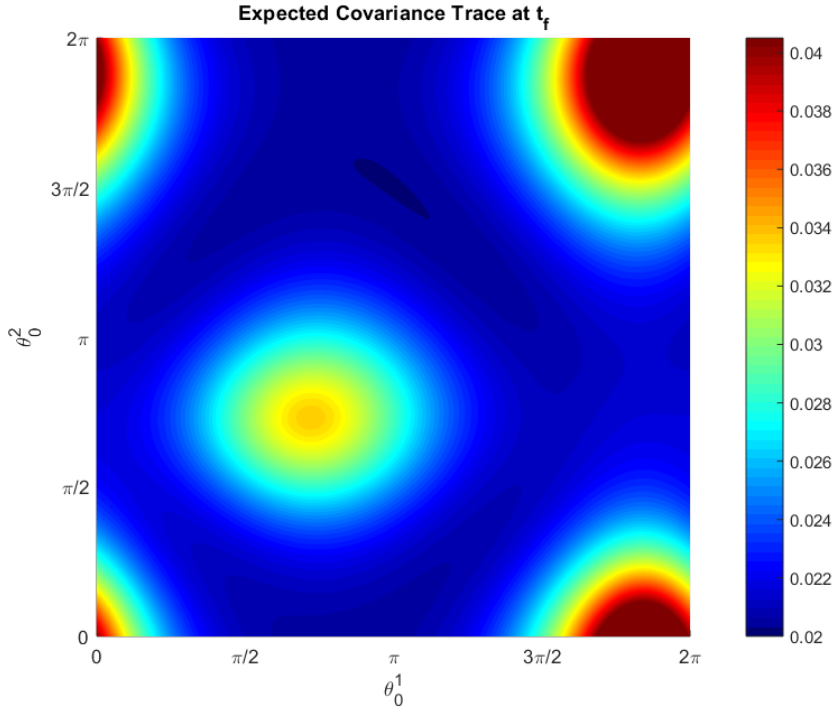

Figure 4.1 Expected covariance trace at $t_f = 5$ for all initial headings.

against the iteration number, as well as

$$\delta_n = \left(\frac{\text{tr}[\mathbf{P}]^2_{\theta_0^1 \theta_0^2}}{\text{tr}[\mathbf{P}]_{\theta_0^1 \theta_0^1} \ \text{tr}[\mathbf{P}]_{\theta_0^2 \theta_0^2}} \right)^n, \tag{4.24}$$

which is the n^{th} power of the largest eigenvalue, evaluated at $\theta_0^1 = \pi$ and $\theta_0^2 = \frac{3\pi}{2}$.

We see from this figure that the numerical and exact errors correspond fairly well for $n < 40$, after which, the numerical error plateaus due to the resolution of Δx in the level set method. Furthermore, we find that the rate of convergence is of the same order as δ_n, as expected. This is due to the fact that the iterative method should converge with a rate corresponding to the largest (i.e., slowest) eigenvalue.

4.4.2 Example 2

As a second example, consider a velocity field of the form

$$\mathbf{v} = \sum_{i=1}^{4} \frac{A\beta}{r_i} e^{-\beta r_i} \begin{bmatrix} (x - x_{c_i}) - (-1)^i B(y - y_{c_i}) \\ (y - y_{c_i}) + (-1)^i B(x - x_{c_i}) \end{bmatrix}, \tag{4.25}$$

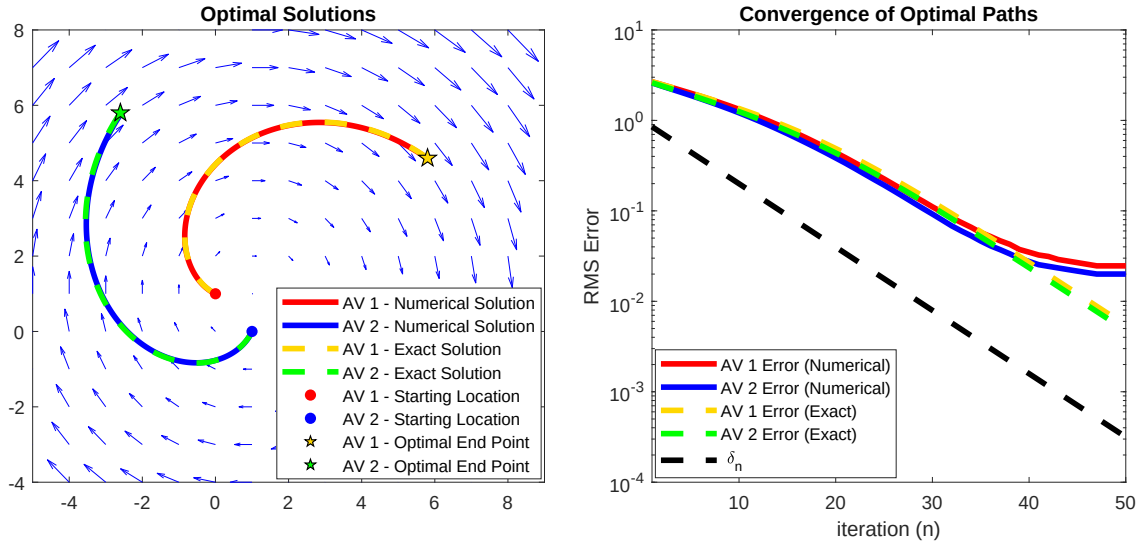

Figure 4.2 Left: Comparison of optimal trajectories. Right: Convergence of solutions to optimal trajectories against the number of iterations.

confined to the domain $[0, 3] \times [0, 3]$, with

$$A = 0.0244,$$

$$\beta = 3,$$

$$B = 0.6,$$

and

$$r_i = \sqrt{(x - x_{c_i})^2 + (y - y_{c_i})^2}. \tag{4.26}$$

For this problem, we are interested in reducing the uncertainty of

$$\hat{\mathbf{x}} = \begin{bmatrix} x_{c_1} & x_{c_2} & x_{c_3} & x_{c_4} & y_{c_1} & y_{c_2} & y_{c_3} & y_{c_4} \end{bmatrix}^T, \tag{4.27}$$

where

$$\hat{\mathbf{x}}_{true} = \begin{bmatrix} 1 & 1 & 2 & 2 & 1 & 2 & 2 & 1 \end{bmatrix}^T, \tag{4.28}$$

subject to the following parameters

$$\mathbf{P}(t_0) = \mathbf{I},$$

$$\mathbf{R}^0 = \sigma^2\mathbf{I},$$

$$\sigma^2 = 0.01,$$

$$F^1 = F^2 = F^3 = 0.03,$$

$$(x_s^1, y_s^1) = (2.7, 0.35),$$

$$(x_s^2, y_s^2) = (2.75, 0.35),$$

$$(x_s^3, y_s^3) = (2.75, 0.4),$$

$$t_0 = 0,$$

$$t_f = 120,$$

and initial condition

$$\hat{\mathbf{x}}_0 = \begin{bmatrix} 0.5 & 1.5 & 1.5 & 2.8 & 0.5 & 2.5 & 1.8 & 1.5 \end{bmatrix}^T.$$

For this problem, 12 path updates (i.e., $M = 12$) are chosen. As a comparison, the covariance-tracking methodology is tested against a gradient-defined control. Note that neither control is capable of using new information regarding the flow until the next path update. The trajectories corresponding to each control are shown in Figure 4.3, whereas the error reductions for the two controls can be found in Figure 4.4. Figure 4.5 shows a comparison of the true velocity field, the initial guess, and the inferred velocity fields for both controls at time t_f.

From Figure 4.3 it can be seen that the three AVs following the coordinated level set control divide to sample different gyres, whereas the three gradient-controlled AVs fall into the same minimum, and as such travel along the same trajectory. We see in Figure 4.4 that this level set control ultimately outperforms the gradient-defined control in reducing both the RMS error and the covariance trace. These results demonstrate the advantages of the covariance-tracking algorithm over the less computationally intensive gradient-following methods.

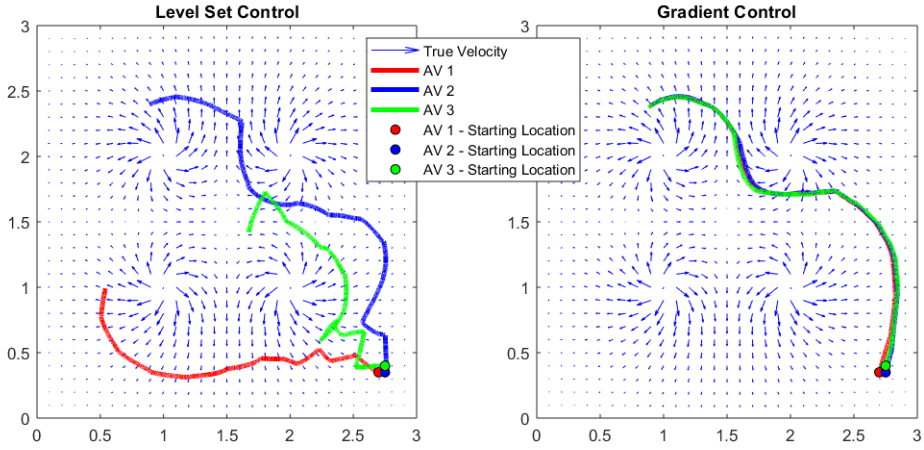

Figure 4.3 Comparison of level set control to gradient-defined control.

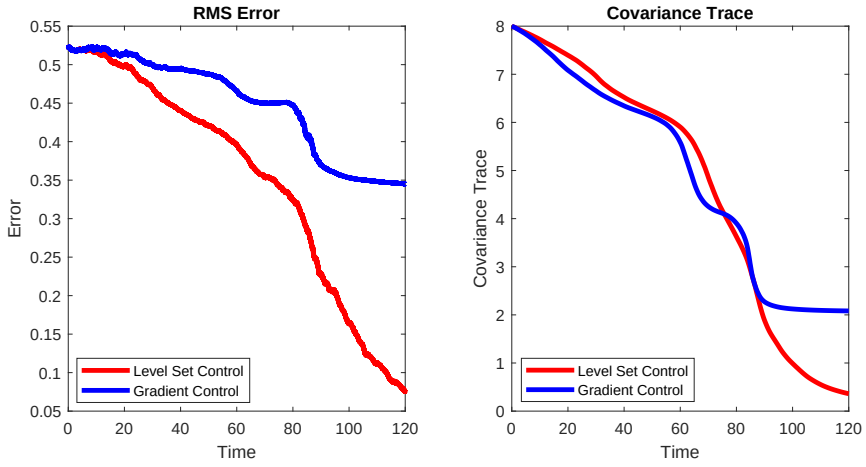

Figure 4.4 Left: Convergence of the RMS error. Right: Convergence of the covariance trace.

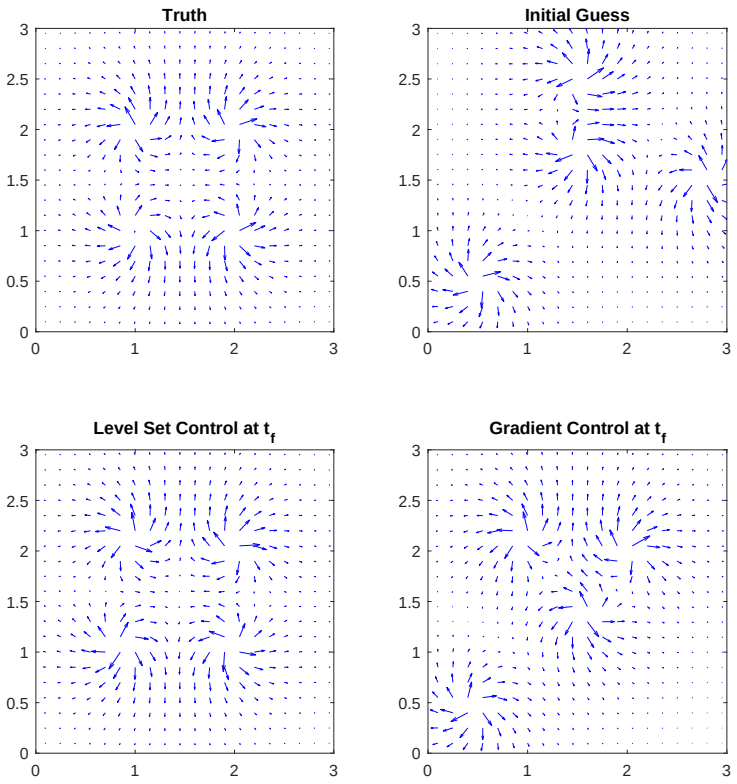

Figure 4.5 A comparison of the true velocity field (top left), the initial guess of the velocity field (top right), and the inferred velocity fields at time t_f using the level set control (bottom left) and the gradient control (bottom right).

CHAPTER 5

TIME-EVOLVING FLOWS

In Chapter 3, it is mentioned that determining accurate ocean models is not a straightforward procedure. This is due to the fact that ocean currents are massively high-dimensional, and can vary greatly on a number of different spatial and temporal scales, particularly for the meso- to macro-scale oceanic dynamics that this work focuses on. As such, we seek a model that is robust enough to capture these variations in both space and time, and simple enough that it can be implemented within the current framework of our methodology. Due to the relatively large scale of our domain, an appropriate approximation for these types of ocean dynamics can be found by using the shallow water equations.

5.1 The Shallow Water Equations

The shallow water equations (SWE) are given by the following set of coupled PDEs

$$\frac{\partial h}{\partial t} + \frac{\partial (hu)}{\partial x} + \frac{\partial (hv)}{\partial y} = 0, \tag{5.1}$$

$$\frac{\partial (hu)}{\partial t} + \frac{\partial}{\partial x}\Big(hu^2 + \frac{gh^2}{2}\Big) + \frac{\partial (huv)}{\partial y} = 0, \tag{5.2}$$

$$\frac{\partial (hv)}{\partial t} + \frac{\partial (huv)}{\partial x} + \frac{\partial}{\partial y}\Big(hv^2 + \frac{gh^2}{2}\Big) = 0, \tag{5.3}$$

where $h = h(\mathbf{x}, t)$ is the (strictly-positive) height of the water, $u = u(\mathbf{x}, t)$ is the water velocity in the x-direction, and $v = v(\mathbf{x}, t)$ is the water velocity in the y-direction. Note here that the Coriolis terms that sometimes appear on the right hand side of these equations are neglected for simplicity. Since it is our goal to use this model within the scope of the Kalman-Bucy filter, we rewrite these equations more concisely

as

$$\frac{\partial h}{\partial t} = A(h, u, v; t), \tag{5.4}$$

$$\frac{\partial (hu)}{\partial t} = B(h, u, v; t), \tag{5.5}$$

$$\frac{\partial (hv)}{\partial t} = C(h, u, v; t). \tag{5.6}$$

For this inference problem, we are interested in finding the horizontal components of the ocean flow, $u(\mathbf{x}, t)$ and $v(\mathbf{x}, t)$, from the above equations. To accomplish this numerically, we choose to discretize our oceanic domain of interest onto an $I \times J$ inference grid, where we seek $u(t)_{i,j} = u(\mathbf{x}_{i,j}, t)$ and $v(t)_{i,j} = v(\mathbf{x}_{i,j}, t)$ at each of these inference points. While we are strictly interested in u and v, it is important to note that these variables are strongly coupled to the water height, h. As such, it is also necessary to try to infer $h(t)_{i,j} = h(\mathbf{x}_{i,j}, t)$, despite the fact that no observations of the water height are being made by the vehicles. Thus, to formulate this problem in terms of the Kalman-Bucy filter, we define a $3IJ$ state-vector of the form

$$\hat{\mathbf{x}}(t) = \begin{bmatrix} \mathbf{u}(t) & \mathbf{v}(t) & \mathbf{h}(t) \end{bmatrix}^T, \tag{5.7}$$

where

$$\mathbf{u}(t) = \begin{bmatrix} u(t)_{1,1} & \dots & u(t)_{I,1} & u(t)_{1,2} & \dots & u(t)_{I,2} & \dots & u(t)_{I,J} \end{bmatrix}^T, \tag{5.8}$$

$$\mathbf{v}(t) = \begin{bmatrix} v(t)_{1,1} & \dots & v(t)_{I,1} & v(t)_{1,2} & \dots & v(t)_{I,2} & \dots & v(t)_{I,J} \end{bmatrix}^T, \tag{5.9}$$

$$\mathbf{h}(t) = \begin{bmatrix} h(t)_{1,1} & \dots & h(t)_{I,1} & h(t)_{1,2} & \dots & h(t)_{I,2} & \dots & h(t)_{I,J} \end{bmatrix}^T. \tag{5.10}$$

Since the shallow water equations are nonlinear, the Extended version of the Kalman-Bucy filter must be used. The evolution of the expected state-vector and covariance matrix are therefore given by the nonlinear versions of Equations (2.32) and (2.33), respectively. To determine the system-model, which describes the evolution

of our state-vector, we make use of Equations (5.4) - (5.6), as well as the chain rule, to find that

$$\frac{\partial u_{i,j}}{\partial t} = \frac{1}{h_{i,j}}\left[\frac{\partial (hu)_{i,j}}{\partial t} - u_{i,j}\frac{\partial h_{i,j}}{\partial t}\right] = \frac{1}{h_{i,j}}\Big[B_{i,j} - u_{i,j}A_{i,j}\Big], \tag{5.11}$$

$$\frac{\partial v_{i,j}}{\partial t} = \frac{1}{h_{i,j}}\left[\frac{\partial (hv)_{i,j}}{\partial t} - v_{i,j}\frac{\partial h_{i,j}}{\partial t}\right] = \frac{1}{h_{i,j}}\Big[C_{i,j} - v_{i,j}A_{i,j}\Big], \tag{5.12}$$

$$\frac{\partial h_{i,j}}{\partial t} = A_{i,j}. \tag{5.13}$$

Utilizing Equations (2.32), (5.11), (5.12), and (5.13), we find that our state-vector evolves as follows

$$\begin{aligned}\dot{\hat{\mathbf{x}}} = &\begin{bmatrix} \frac{B_{1,1}}{h_{1,1}} & \dots & \frac{B_{I,J}}{h_{I,J}} & \frac{C_{1,1}}{h_{1,1}} & \dots & \frac{C_{I,J}}{h_{I,J}} & A_{1,1} & \dots & A_{I,J}\end{bmatrix}^T \\ &- \begin{bmatrix} \frac{u_{1,1}A_{1,1}}{h_{1,1}} & \dots & \frac{u_{I,J}A_{I,J}}{h_{I,J}} & \frac{v_{1,1}A_{1,1}}{h_{1,1}} & \dots & \frac{v_{I,J}A_{I,J}}{h_{I,J}} & 0 & \dots & 0\end{bmatrix}^T \\ &+ \mathbf{P}(t)\mathbf{H}(t)^T\mathbf{R}^0(t)^{-1}\Big(\mathbf{z}(t) - \mathbf{H}(t)\hat{\mathbf{x}}(t)\Big).\end{aligned} \tag{5.14}$$

Since Equation (5.14) cannot be solved explicitly, we will discretize in time and solve this equation numerically. Of course, to do this will require numerical approximations of $A_{i,j}$, $B_{i,j}$ and $C_{i,j}$. These are found using the high-resolution, finite-volume scheme developed by Leveque in [33] for the shallow water equations (Equations (5.1) - (5.3)). Thus, we have an approximation for the evolution of our state-vector given by

$$\begin{aligned}\frac{\hat{\mathbf{x}}^{n+1} - \hat{\mathbf{x}}^n}{\Delta t} = &\begin{bmatrix} \frac{B^n_{1,1}}{h^n_{1,1}} & \dots & \frac{B^n_{I,J}}{h^n_{I,J}} & \frac{C^n_{1,1}}{h^n_{1,1}} & \dots & \frac{C^n_{I,J}}{h^n_{I,J}} & A^n_{1,1} & \dots & A^n_{I,J}\end{bmatrix}^T \\ &- \begin{bmatrix} \frac{u^n_{1,1}A^n_{1,1}}{h^n_{1,1}} & \dots & \frac{u^n_{I,J}A^n_{I,J}}{h^n_{I,J}} & \frac{v^n_{1,1}A^n_{1,1}}{h^n_{1,1}} & \dots & \frac{v^n_{I,J}A^n_{I,J}}{h^n_{I,J}} & 0 & \dots & 0\end{bmatrix}^T \\ &+ \mathbf{P}^n(\mathbf{H}^n)^T\big(\mathbf{R}^{0^n}\big)^{-1}\Big(\mathbf{z}^n - \mathbf{H}^n\hat{\mathbf{x}}^n\Big),\end{aligned} \tag{5.15}$$

where the superscript n denotes the n^{th} time-step.

Determining the evolution of the covariance matrix requires finding the Jacobian of the system-model with respect to the state-vector. In terms of Equation (2.33),

this means replacing $\mathcal{F}$ with $\left.\frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}}\right|_{\hat{\mathbf{x}}}$, where

$$\begin{aligned}\mathbf{f}(\hat{\mathbf{x}}) = &\begin{bmatrix} \frac{B_{1,1}}{h_{1,1}} & \dots & \frac{B_{I,J}}{h_{I,J}} & \frac{C_{1,1}}{h_{1,1}} & \dots & \frac{C_{I,J}}{h_{I,J}} & A_{1,1} & \dots & A_{I,J} \end{bmatrix}^T \\ &- \begin{bmatrix} \frac{u_{1,1}A_{1,1}}{h_{1,1}} & \dots & \frac{u_{I,J}A_{I,J}}{h_{I,J}} & \frac{v_{1,1}A_{1,1}}{h_{1,1}} & \dots & \frac{v_{I,J}A_{I,J}}{h_{I,J}} & 0 & \dots & 0 \end{bmatrix}^T.\end{aligned} \tag{5.16}$$

Again, discretizing in time and solving numerically, we find an approximation for the evolution of our covariance matrix, given by

$$\frac{\mathbf{P}^{n+1}-\mathbf{P}^n}{\Delta t} = \left(\left.\frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}^n}\right|_{\hat{\mathbf{x}}^n}\right)\mathbf{P}^n + \mathbf{P}^n\left(\left.\frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}^n}\right|_{\hat{\mathbf{x}}^n}\right)^T + \mathbf{Q}^n - \mathbf{P}^n(\mathbf{H}^n)^T(\mathbf{R}^{0^n})^{-1}\mathbf{H}^n\mathbf{P}^n. \tag{5.17}$$

Thus, Equations (5.15) and (5.17) represent the Extended Kalman-Bucy filter when the shallow water equations are used as a system-model.

Finally, we must determine our observation operator $\mathbf{H}$ in order to make use of Equations (5.15) and (5.17). Since our state-vector is just the components of the underlying flow at the inference points, our observation operator will simply be a sparse matrix which picks out the corresponding state components, depending on the vehicle location within the given domain. Since it is expected that most of the time our vehicle's location will be between inference grid points, we have chosen to make $\mathbf{H}$ a weighted average of the closest ℓ inference points, where $1 \leq \ell \leq IJ$.

To demonstrate this, suppose we are interested in using a 4-point weighted average (i.e., $\ell = 4$) to define $\mathbf{H}$. Referring to Figure 5.1 as an example, we determine the four closest inference points to our vehicle (the red star), as well as their corresponding distances (labeled d_1, d_2, d_3 and d_4). In this example, each of those four points would be weighted as

$$w_i = \frac{1}{Cd_i}, \tag{5.18}$$

where $1 \leq i \leq 4$, and C is a normalizing constant, defined as

$$C = \sum_{j=1}^{4} \frac{1}{d_j}. \tag{5.19}$$

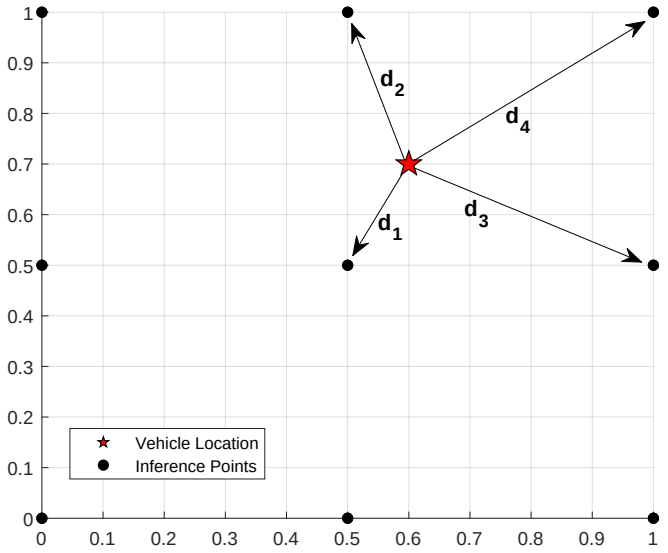

Figure 5.1 Demonstration using a 4-point weighted average for the observation operator. The red star is the vehicle's current location, the black dots are the inference points, and the arrows represent the distances between the vehicle and the four closest inference points.

It is important to note here that in the circumstance that the vehicle's location lies exactly on one of the inference-grid points, then the weight for that point would simply be set to 1, and all others set to 0.

Having all necessary information, we find that for this example with a 3×3 inference grid and only a single vehicle, $\mathbf{H}$ is a 2×27 sparse matrix with 8 non-zero terms defined by

$$\mathbf{H}_{1,4} = \mathbf{H}_{2,13} = w_2,$$

$$\mathbf{H}_{1,5} = \mathbf{H}_{2,14} = w_1,$$

$$\mathbf{H}_{1,7} = \mathbf{H}_{2,16} = w_4,$$

$$\mathbf{H}_{1,8} = \mathbf{H}_{2,17} = w_3.$$

Note that this method for defining $\mathbf{H}$ easily extends to utilize any number of points in our weighted average, so long as $1 \leq \ell \leq IJ$. For the remainder of this chapter, however, we use a 4-point weighted average to define our observation operator.

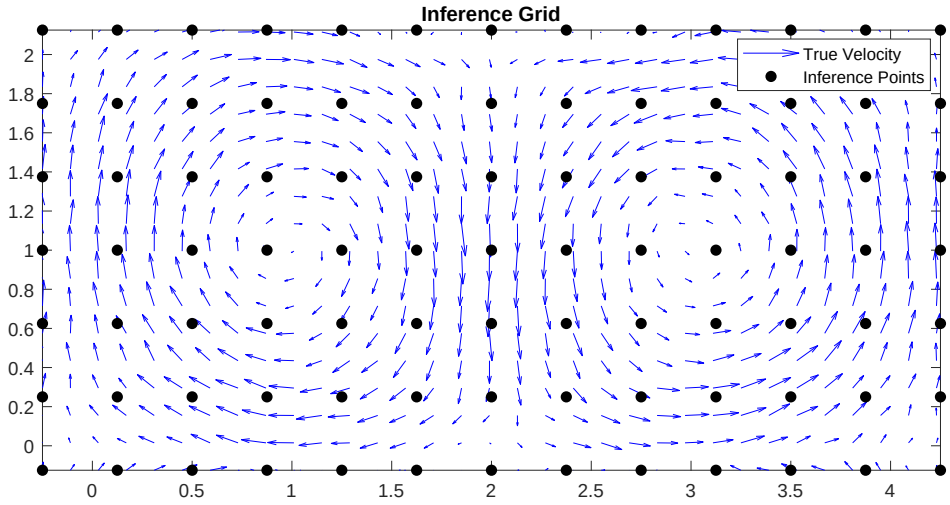

Figure 5.2 The 7×13 inference grid used for this example, overlaying the true velocity.

5.2 Example

Consider four autonomous surface vehicles confined to an oceanic domain (arbitrarily chosen to have size $[-0.25, 4.25] \times [-0.125, 2.125]$), which has a noisy surface current representing a double-gyre system, and evolves according to Equations (5.1) - (5.3). For this problem, we are interested in determining the values of u, v, and h on a 7×13 inference grid, which is shown in Figure 5.2, overlaying the domain and true velocity. Thus, our state-vector for this problem has 273 components, and takes the form of Equation (5.7).

In initializing this problem, we assume no prior information regarding u or v, and as such, set these values to zero at each inference point. Since the height components must be greater than zero, we initialize these to be the arbitrarily chosen value of 1.5. Thus, our initial state-vector is given by

$$\hat{\mathbf{x}}(t_0) = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{1.5} \end{bmatrix}^T, \tag{5.20}$$

where $\mathbf{0}$ is the zero-vector with 91 components, and $\mathbf{1.5}$ is a vector whose 91 components are all 1.5. The rest of the parameters for this problem are chosen as follows

$$\mathbf{P}(t_0) = \mathbf{I},$$

$$\mathbf{R}^0 = \sigma^2\mathbf{I},$$

$$\mathbf{Q} = \omega^2\mathbf{I},$$

$$\sigma = \omega = 0.001,$$

$$F^1 = F^2 = F^3 = F^4 = 0.1,$$

$$(x_s^1, y_s^1) = (0, 0),$$

$$(x_s^2, y_s^2) = (0, 2),$$

$$(x_s^3, y_s^3) = (4, 0),$$

$$(x_s^4, y_s^4) = (4, 2),$$

$$t_0 = 0,$$

$$t_f = 20,$$

and finally, there will be four path updates (i.e., $M = 4$), which will occur at times $t = 4$, 8, 12, and 16. As a comparison, the covariance-tracking methodology is tested against a gradient-defined control. Note that neither control is capable of using new information regarding the flow until the next path update. The trajectories for each vehicle, corresponding to both control methods, are shown in Figure 5.3. As anticipated, we see here that using the paths defined by the level set methodology allow the vehicles to explore more of their given domain compared to the vehicles following a gradient-based approach.

Figure 5.4 shows how our estimated (using the covariance-tracking method) and true velocity field evolve at each of the update times, as well as at the final time. We note here that periodic boundary conditions are employed for this problem, which appears to have given the true velocity field a fairly steady flow (see the right-side of Figure 5.4). Additionally, we note that the true value of $h(\mathbf{x}, t)$ is approximately equal to 1 for the entirety of this simulation, at least up to a noise term. With this

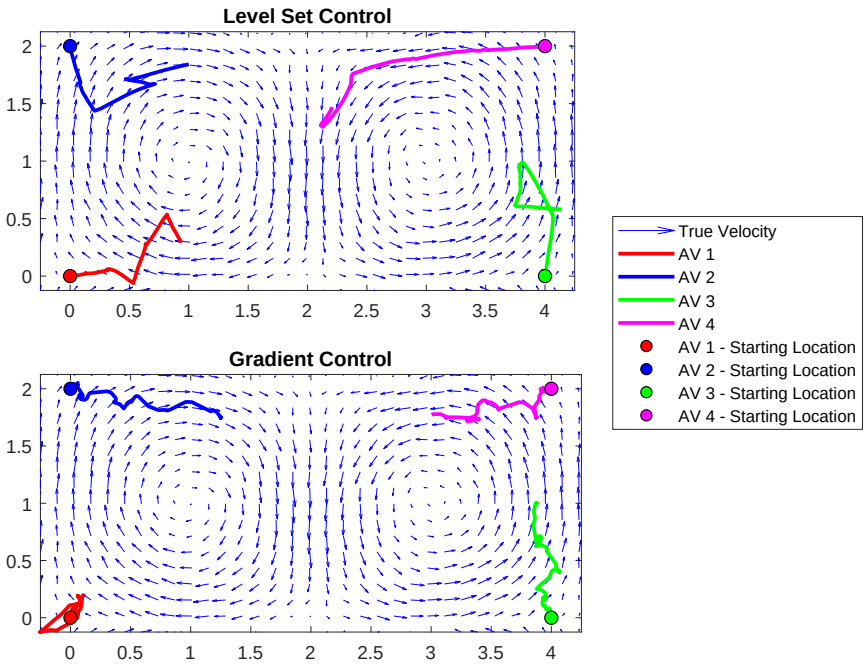

Figure 5.3 Comparison of level set control (top) to gradient-defined control (bottom).

knowledge of the true flow and water height, we can calculate the RMS error between our estimates and the truth at each time-step. This information is plotted in Figure 5.5, along with the evolution of the covariance-trace (i.e., tr[$\mathbf{P}(t)$]), for both control methods.

By comparing the level set trajectories in Figure 5.3 with the evolution of the estimate in Figure 5.4, we can see that the accuracy of the estimate is highly dependent on the location of the vehicles. More specifically, the regions of the flow with the least deviation from the truth are those which have been sampled by one of the four vehicles. Indeed, this is expected given our definition of the state-vector and observation operator.

From Figure 5.5 we find that the trace of the covariance matrices for both methods are comparable, and not much deviation is found between the two. Looking at the RMS error though, we do find that the level set control does in fact out-perform the gradient-control. Interestingly, however, the RMS error ultimately increases,

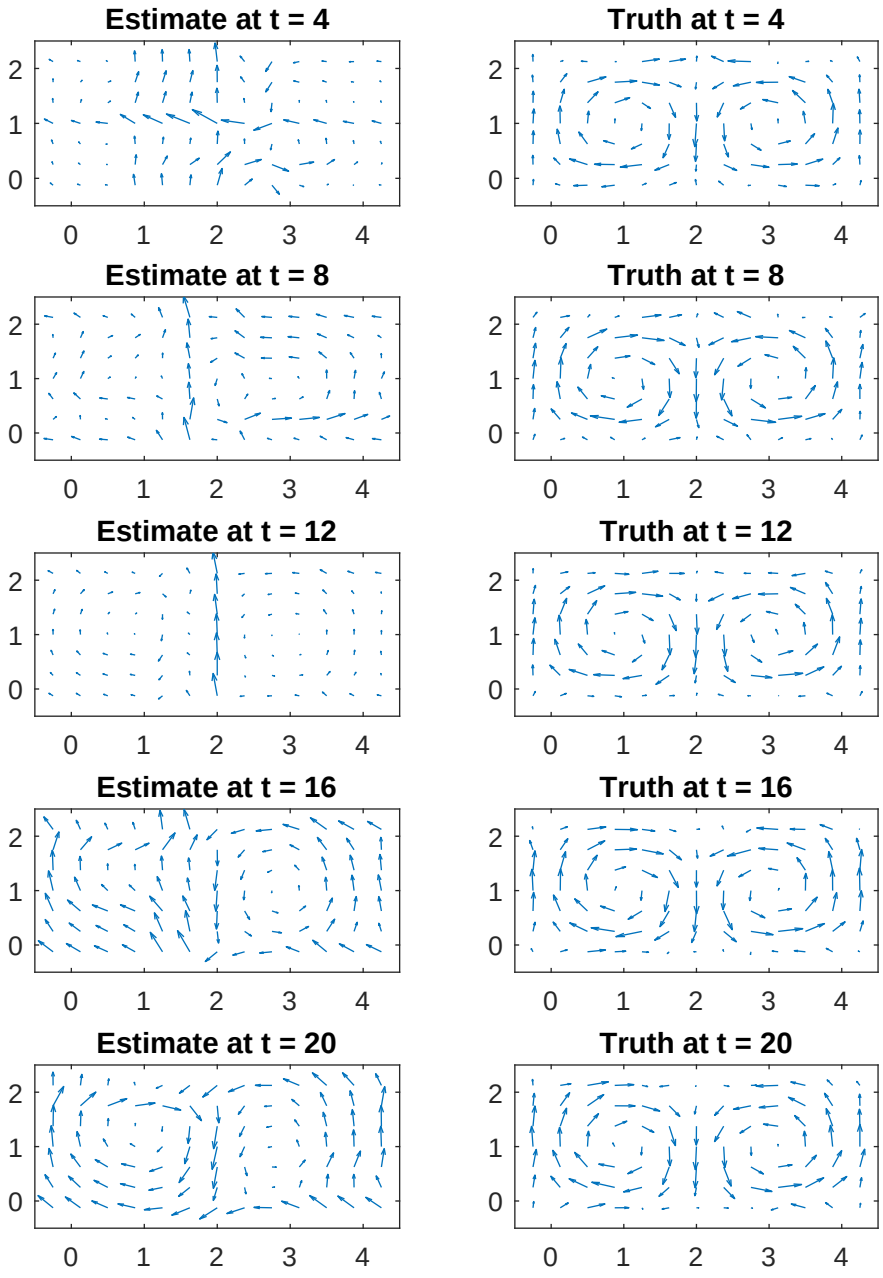

Figure 5.4 Left: The estimate of the velocity field at each update time using the level set control. Right: The true velocity field at each update time.

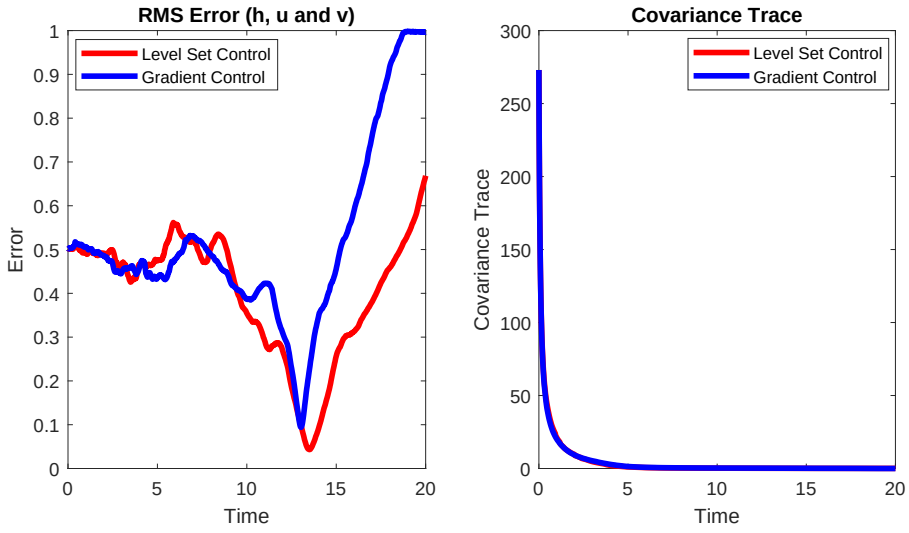

Figure 5.5 Left: Convergence of the RMS errors. Right: Convergence of the covariance trace.

rather than decreases. Careful analysis of this situation determined that the RMS error in Figure 5.5, which takes into account the errors in h, u, and v, is actually dominated by the error in h. This can be seen in Figure 5.6, which looks at the error in h independently of the errors in u and v.

The right panel of Figure 5.6 shows the RMS convergence for the simulation in only u and v. As desired, we find that the level set control ultimately out-performs the gradient-control in learning the flow. This is further demonstrated in Figure 5.7, which plots the inferred velocity fields at time t_f for both the level set control (top) and the gradient-control (bottom), as well as the true velocity field at that time (middle).

In the left panel of Figure 5.6, we find that for both controls, the RMS error for h drops to zero before spiking upwards. This is due to the fact that the EKF being implemented here is driving the values of h at each inference point to zero. Thus, since we initialize with $h(\mathbf{x}, t_0) = 1.5$, the RMS error decreases as h approaches one, and increases thereafter. The EKF forcing h to zero may be due to the fact that we are attempting to infer an approximately constant value of h using a system

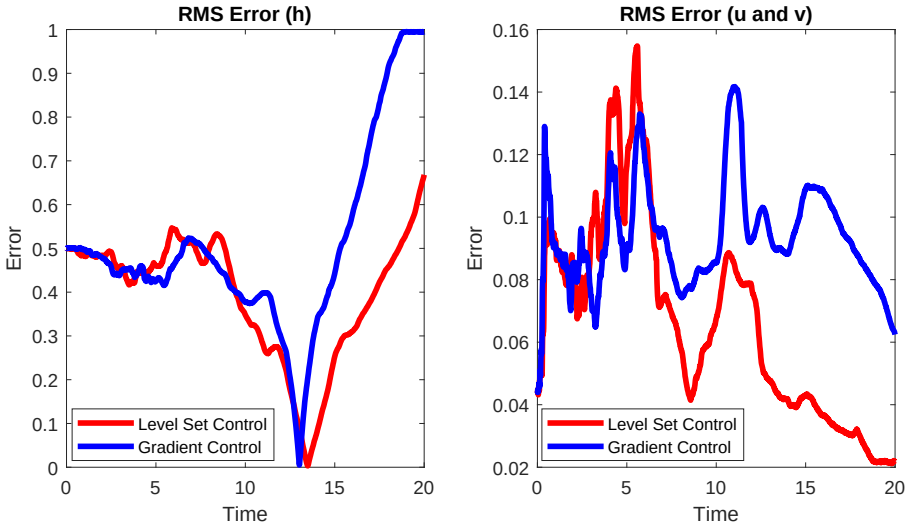

Figure 5.6 Left: RMS error in h. Right: RMS error in u and v.

of PDEs. While the system successfully determines that h is essentially a constant function, it seems unable to determine the value of that constant without having any direct information regarding the true water height. As such, since the Kalman filter is designed to minimize the covariance trace, it would appear that without having more observational data, smaller values of h are optimal for minimizing $\text{tr}[\mathbf{P}(t)]$.

The results found from this example not only further demonstrate the capabilities of the covariance-tracking algorithms discussed in Chapters 3 and 4, but also demonstrates the capabilities of using the shallow water equations as a model within the Extended Kalman-Bucy filter. Although this system was unable to reliably estimate the water height, it is again noted that we are only truly interested in estimating $u(\mathbf{x}, t)$ and $v(\mathbf{x}, t)$, and that the values of $h(\mathbf{x}, t)$ were only included in the state-vector due to its strong coupling with u and v.

5.3 Convergence

This work has thus far demonstrated the convergence of the level set method, as well as the convergence of the iterative scheme used in the covariance-tracking algorithm. Additionally, it is shown in [33] that the high-resolution, finite-volume

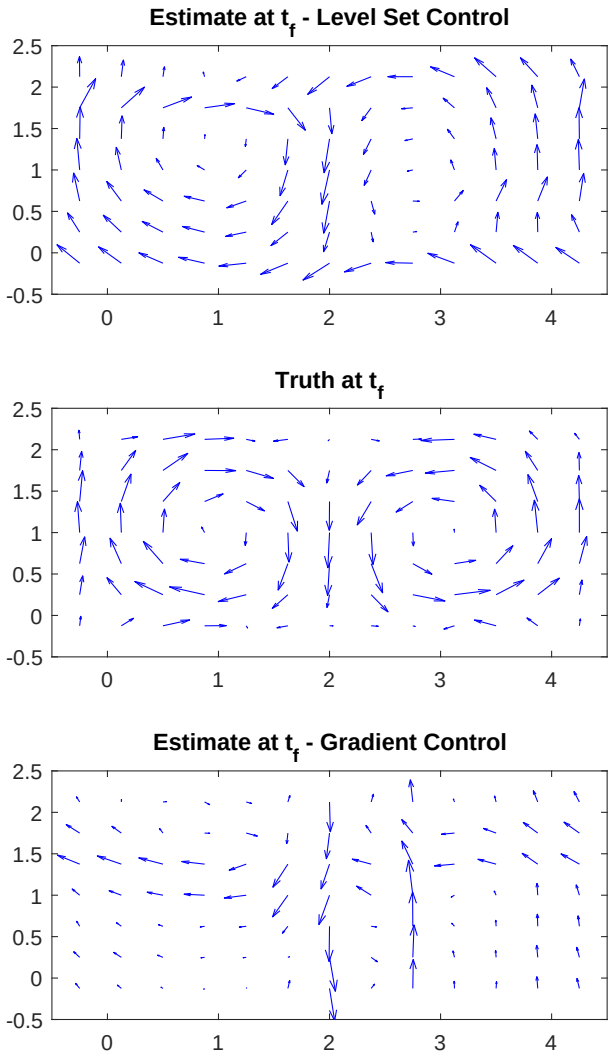

Figure 5.7 Top: The inferred velocity field at time t_f using the level set control. Middle: The true velocity field at time t_f. Bottom: The inferred velocity field at time t_f using the gradient control.

method employed to solve the shallow water equations is also convergent. While each of these components may converge individually, we ultimately desire to demonstrate the convergence of this method as a whole (i.e., show that the estimated velocities at t_f converge as Δx and Δt approach 0). Unfortunately, this is not currently possible due to computational constraints. More specifically, the memory required to run a simple example to the necessary resolution in order to find convergence is outside of our current capabilities.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

Our approach uses a level set method for time-optimal path planning in order to find a family of candidate trajectories for the autonomous vehicles. We then use a variant of the Kalman filter to evolve a metric of the uncertainty (i.e., $\text{tr}[\mathbf{P}(t)]$) along each of these trajectories to determine which will culminate in the lowest expected uncertainty at the end of deployment. While implementation of this method is straightforward for the single-vehicle problem, a modification must be made for multi-vehicle problems in order to find controls which will collectively minimize the expected uncertainty. As such, an iterative procedure is introduced, which is shown to converge to a minimum where one exists.

This methodology is demonstrated for static and time-evolving flow models. Chapters 1-4 focus on low-dimensional, explicit equations for static flows to more clearly show the utility of this method. Chapter 5 then generalizes our approach to the shallow water equations (SWE) as a model for velocity fields more relevant to the ocean. We show that while our algorithm is capable of accurately estimating the horizontal components of the underlying flow, it struggles to accurately determine the water height when using this SWE model. This is due to that fact that there is no information regarding water height in our observations, leading to difficulties in determining the appropriate amplitude scaling for inferred height.

Many of the examples in this work compare the results obtained using the proposed methodology to those found using a gradient-based approach. These examples truly demonstrate the efficacy of the proposed algorithm and, in particular, show that by optimizing over long-duration, continuous trajectories, superior results

can be obtained. Specifically, we find that our method allows for these vehicles to avoid ineffective local minima, which is where their gradient-following counterparts typically get caught. However, it is noted that these results come with a higher computational cost than many alternative methods. As such, this algorithm would be better suited for oceanographic studies with deployment times on the order of days, weeks, or months.

6.2 Future Work

6.2.1 Expanding to Three Dimensions

Thus far, we have only considered the two-dimensional optimal control problem. Specifically, we have been considering autonomous vehicles constrained to the surface of an oceanic domain. In truth, AUVs such as Spray and Slocum gliders dive to depths of approximately a kilometer as they travel, extending below the surface layer or "sunlight zone". This requires consideration of either three-dimensional or coupled two-dimensional velocity field models, which is planned for future work.

We can also use a functional form for the motion of the gliders in this additional spatial dimension. Specifically, we can assume that they trace out sinusoidal or sawtooth paths as they travel. Such an assumption would allow us to somewhat reduce the dimensionality of the optimization problem, while still retaining some of the more complex structures of ocean currents beyond the surface model. Clearly, the computational expense of data assimilation in this case would increase with increased complexity of the velocity model.

While the level set method can easily be extended to take into account additional dimensions, it comes at the price of increased computational complexity. A partial fix to this problem is the implementation of a narrow-band method for our level set algorithm. First introduced by Chopp [9], this method constrains calculations of $\phi(\mathbf{x}, t)$ to only a narrow-band around the zero-level set, as opposed to performing

calculations over the entire domain. For a three-dimensional problem this drops the total number of calculations from $O(N^3)$ to $O(kN^2)$, where N is the number of grid points along one side, and k is the number of cells within the band [44]. It is noted that this narrow-band methodology has already been implemented in the two-dimensional problem.

6.2.2 Efficiently Modeling Velocity Fields

As discussed in Chapter 5, realistic oceanic currents require sophisticated techniques in order to be accurately modeled. However, these models can result in very high-dimensional systems, leaving the problem computationally intractable for real-time applications. This is indeed the case in Chapter 5, where we have taken the naive approach of discretizing the entire domain, and assumed a simple velocity within each grid box (e.g., a constant flow). Note, however, that for an $I \times J$ discretization of a two-dimensional domain, this results in a state-vector with $3IJ$ components. This implies considerable computational expense, and is unnecessary in highly structured flows.

An alternative method to help determine ocean dynamics is to look for transport barriers, or Lagrangian Coherent Structures (LCS), within our domain. It is known that LCSs contain extensive amounts of information regarding large scale ocean currents, and it was demonstrated in [37] that these structures heavily affect posterior information. Thus, we can use LCSs for both determining an approximate functional form of the ocean currents, as well as reducing the overall uncertainty of it. Additionally, there already exists methodologies for tracking coherent structures using groups of coordinated robots [19], as well as using LCSs to quickly determine near-optimal vehicle trajectories [20]. Both of these methods may be of interest moving forward.

In [45], Shadden et al. define LCSs as ridges of finite-time Lyapunov exponent (FTLE) fields, and use this definition to numerically extract LCSs from existing data sets. In [17], Haller and Yuan provide a method for finding coherent structure boundaries by searching for stable and unstable material lines where "hyperbolicity times" are locally maximized or minimized. These numerical techniques can be implemented in conjunction with our current methodology in an attempt to limit the dimensionality of the state-vector, and thus make the algorithm computationally feasible.

6.2.3 Experimental Verification

While our numerical simulations have shown promising results so far, we ultimately wish to see how they would work for actual AVs in real currents. This leads us to pursue experimental verification of our methodology. Specifically, we would like to know if the uncertainty reduction seen numerically matches the uncertainty reduction we would see in practice. Fortunately, this can be done with collaborators at the University of Pennsylvania who are in possession of the Multi-Robot Coherent Structure Testbed (mCoSTe).

The mCoSte is an experimental test-bed capable of producing controllable currents, and supports the deployment of multiple micro-autonomous surface vehicles (mASVs). The currents are created using a grid of motorized flow-driving cylinders, which can simulate a variety of different time-independent and time-varying flow fields. The information regarding these flows can be determined using either particle imaging velocimetry (PIV) and/or particle tracking velocimetry (PTV). This will provide us with real velocity data and a corresponding uncertainty to test our proposed methods.

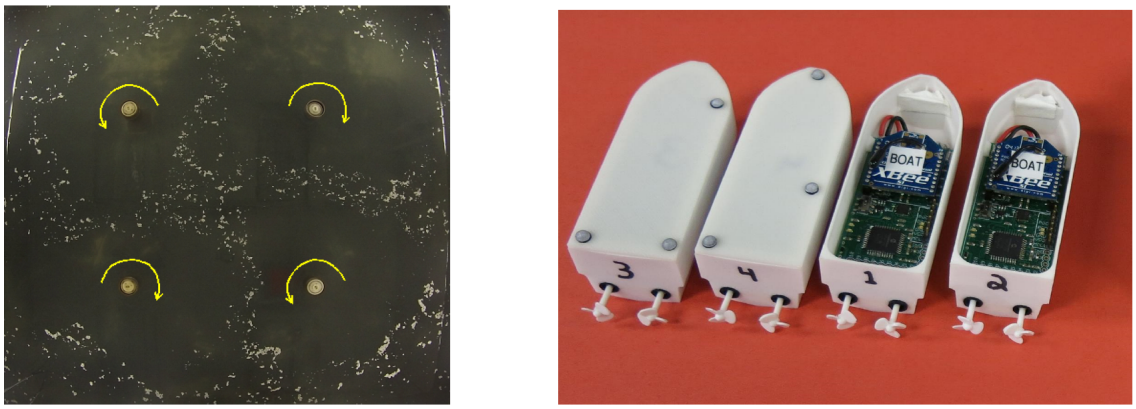

Figure 6.1 Left: The mCoSTe experimental testbed simulating a double gyre flow. Right: micro-autonomous surface vehicles (mASVs).

Figure 6.1, which is taken from [26], shows the mCoSTe experimental test-bed (left) as well as the mASVs (right) that we wish to use in order to demonstrate the capabilities of our algorithm.

BIBLIOGRAPHY

[1] History of the Odyssey-Class of Autonomous Underwater Vehicles. `https://auvlab.mit.edu/history.html`. (accessed on 7/3/2019), AUV Laboratory, MIT, Cambridge, MA.

[2] National Aeronautics and Space Administration Earth Science website: Currents. `https://science.nasa.gov/earth-science/oceanography/physical-ocean/currents`. (accessed on 7/3/2019), Washington, D.C.

[3] National Oceanic and Atmospheric Administration Marine Debris Program. 2016 Report on Modeling Oceanic Transport of Floating Marine Debris. Silver Spring, MD.

[4] National Oceanic and Atmospheric Administration website: What is an ocean glider? `https://oceanservice.noaa.gov/facts/ocean-gliders.html`. (accessed on 7/3/2019), Silver Spring, MD.

[5] Woods Hole Oceanographic Institution website: AUVs. `https://www.whoi.edu/what-we-do/explore/underwater-vehicles/auvs/`. (accessed on 7/3/2019), Woods Hole, MA.

[6] National Oceanic and Atmospheric Administration website: Ocean currents. `http://www.noaa.gov/resource-collections/ocean-currents`, 2011. (accessed on 7/3/2019), Silver Spring, MD.

[7] National Oceanic and Atmospheric Administration Website: Gulf oil spill. `http://www.noaa.gov/resource-collections/gulf-oil-spill`, 2013. (accessed on 7/3/2019), Silver Spring, MD.

[8] James G. Bellingham. New Oceanographic Uses of Autonomous Underwater Vehicles. *Marine Technology Society Journal*, 31(3):34–47, 1997.

[9] David L. Chopp. Computing Minimal Surfaces via Level Set Curvature Flow. *Journal of Computational Physics*, 106:77–91, 1993.

[10] Edsger W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[11] Edward Fiorelli, Naomi E. Leonard, Pradeep Bhatta, Derek A. Paley, Ralf Bachmayer, and David M. Fratantoni. Multi-AUV Control and Adaptive Sampling in Monterey Bay. *IEEE Journal of Oceanic Engineering*, 31(4):935–948, 2006.

[12] Bartolome Garau, Alberto Alvarez, and Gabriel Oliver. Path Planning of Autonomous Underwater Vehicles in Current Fields with Complex Spatial Variability: an A$*$ Approach. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 1996.

[14] Annalisa Griffa, A. D. Jr. Kirwan, Arthur J. Mariano, Tamay Ozgokmen, and H. Thomas Rossby, editors. *Lagrangian Analysis and Prediction of Coastal and Ocean Dynamics*. Cambridge, UK: Cambridge University Press, 2007.

[15] Axel Hackbarth, Edwin Kreuzer, and Thorben Schröder. CFD in the Loop: Ensemble Kalman Filtering With Underwater Mobile Sensor Networks. In *Proceedings of the ASME 33rd International Conference on Ocean, Offshore and Arctic Engineering*, volume 2: CFD and VIV, 2014.

[16] William W. Hager. Updating the Inverse of a Matrix. *SIAM Review*, 31(2):221–239, 1989.

[17] George Haller and Guocheng Yuan. Lagrangian Coherent Structures and Mixing in Two-Dimensional Turbulence. *Physica D: Nonlinear Phenomena*, 147:352–370, 2000.

[18] Wilfred K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970.

[19] M. Ani Hsieh, Eric Forgoston, T. William Mather, and Ira B. Schwartz. Robotic Manifold Tracking of Coherent Structures in Flows. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2012.

[20] Tamer Inanc, Shawn C. Shadden, and Jerrold E. Marsden. Optimal Trajectory Generation in Ocean Flows. In *Proceedings of the IEEE American Control Conference*, 2005.

[21] Arthur E. Bryson Jr. and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. New York, NY: Taylor & Francis Group, 1975.

[22] Simon J. Julier and Jeffrey K. Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers: Signal Processing, Sensor Fusion, and Target Recognition VI*, 1997.

[23] Rudolf E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[24] Rudolf E. Kalman and Richard S. Bucy. New Results in Linear Filtering and Prediction Theory. *Journal of Basic Engineering*, 83(1):95–108, 1961.

[25] Sertac Karaman and Emilio Frazzoli. Sampling-Based Algorithms for Optimal Motion Planning. *The Internation Journal of Robotics Research*, 30:846–894, 2011.

[26] Dhanushka Kularatne, Subhrajit Bhattacharya, and M. Ani Hsieh. Time and Energy Optimal Path Planning in General Flows. In *Proceedings of the Robotics: Science and Systems Conference*, 2016.

[27] Ajeet Kumar and Alexander Vladimirsky. An Efficient Method for Multiobjective Optimal Control and Optimal Control Subject to Integral Constraints. *Journal of Computational Mathematics*, 28(4):517–551, 2010.

[28] Steven M. LaValle. *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 2006.

[29] Naomi E. Leonard and Edward Fiorelli. Virtual Leaders, Artificial Potentials and Coordinated Control of Groups. In *Proceedings of the 40th IEEE International Conference on Decision and Control*, 2001.

[30] Naomi E. Leonard, Derek A. Paley, Russ E. Davis, David M. Fratantoni, Francois Lekien, and Fuming Zhang. Coordinated Control of an Underwater Glider Fleet in an Adaptive Ocean Sampling Field Experiment in Monterey Bay. *Journal of Field Robotics*, 27:718–740, 2010.

[31] Naomi E. Leonard, Derek A. Paley, Francois Lekien, Rodolphe Sepulchre, David M. Fratantoni, and Russ E. Davis. Collective Motion, Sensor Networks, and Ocean Sampling. *Proceedings of the IEEE*, 95(1):48–74, 2007.

[32] Pierre F.J. Lermusiaux, Sri Venkata Tapovan Lolla, Patrick J. Haley Jr., Konuralp Yigit, Mattheus P. Ueckermann, Thomas Sondergaard, and Wayne G. Leslie. *Science of Autonomy: Time-Optimal Path Planning and Adaptive Sampling for Swarms of Ocean Vehicles*, pages 481–498. New York, NY: Springer, 2016.

[33] Randall J. Leveque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge, UK: Cambridge University Press, 2002.

[34] Sri Venkata Tapovan Lolla. Path Planning in Time Dependent Flows using Level Set Methods. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2012.

[35] Sri Venkata Tapovan Lolla, Mattheus P. Ueckermann, Konuralp Yigit, Patrick J. Haley Jr., and Pierre F.J. Lermusiaux. Path Planning in Time Dependent Flow Fields using Level Set Methods. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2012.

[36] R. Timothy Marler and Jasbir Singh Arora. Survey of Multi-Objective Optimization Methods for Engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[37] Damon McDougall and Chris K. R. T. Jones. *Decreasing Flow Uncertainty in Bayesian Inverse Problems Through Lagrangian Drifter Control*, pages 215–228. New York, NY: Springer, 2016.

[38] Damon McDougall and Richard O. Moore. Optimal Strategies for the Control of Autonomous Vehicles in Data Assimilation. *Physica D: Nonlinear Phenomena*, 351-352:42–52, 2017.

[39] Ian M. Mitchell. *Application of Level Set Methods to Control and Reachability Problems in Continuous and Hybrid Systems*. PhD thesis, Stanford University, Stanford, CA, 2002.

[40] Ian M. Mitchell and Shankar Sastry. Continuous Path Planning with Multiple Constraints. In *Proceedings of the 42nd IEEE International Conference on Decision and Control*, 2003.

[41] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. New York, NY: Springer, 2003.

[42] Stanley Osher and James A. Sethian. Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[43] Blane Rhoads, Igor Mezic, and Andrew Poje. Minimum Time Feedback Control of Autonomous Underwater Vehicles. In *Proceedings of the 49th IEEE International Conference on Decision and Control*, 2010.

[44] James A. Sethian. *Level Set Methods and Fast Marching Methods.* Cambridge, UK: Cambridge University Press, 1999.

[45] 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*, 212:271–304, 2005.

[46] Robert F. Stengel. *Optimal Control and Estimation.* Mineola, NY: Dover Publications, 1994.

[47] Deepak N. Subramani and Pierre F.J. Lermusiaux. Energy-Optimal Path Planning by Stochastic Dynamically Orthogonal Level-Set Optimization. *Ocean Modelling*, 100:57–77, 2016.

[48] Mark Sussman and Emad Fatemi. An Efficient, Interface-Preserving Level Set Redistancing Algorithm and Its Application to Interfacial Incompressible Fluid Flow. *SIAM Journal on Scientific Computing*, 20(4):1165–1191, 1999.

[49] Mark Sussman, Peter Smereka, and Stanley Osher. A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *Journal of Computational Physics*, 114(1):146–159, 1994.

[50] Chiew Seon Tan, Robert Sutton, and John Chudley. An Incremental Stochastic Motion Planning Technique for Autonomous Underwater Vehicles. *Interational Federation of Automatic Control Proceedings Volumes*, 37(10):483–488, 2004.

[51] Claire J. Tomlin. *Hybrid Control of Air Traffic Management Systems.* PhD thesis, University of California, Berkeley, CA, 1998.

[52] John N. Tsitsiklis. Efficient Algorithms for Globally Optimal Trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[53] Robert L. Wernli. AUV's – The Maturity of the Technology. In *Proceedings of the MTS/IEEE Oceans '99 Conference: Riding the Crest into the 21st Century*, 2000.

[54] Bin Xu, Daniel J. Stilwell, and Andrew Kurdila. Efficient Computation of Level Sets for Path Planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[55] Konuralp Yigit. Path Planning Methods for Autonomous Underwater Vehicles. Master's thesis, Massachusetts Intistitute of Technology, Cambridge, MA, 2011.