UNIVERSITY OF CALIFORNIA SAN DIEGO

**Ensemble-Based Adaptive Observation**

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Engineering Sciences: Mechanical Engineering

by

Mark Wong

Committee in charge:

Professor Thomas Bewley, Chair
Professor Robert Bitmead
Professor Mauricio de Oliveira

2018

The thesis of Mark Wong is approved, and it is acceptable in quality and form for publication on microfilm:

_____

_____

_____
Chair

University of California San Diego

2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thank you to Professor Bewley for his continuous help putting this research together.

This thesis, in part, is currently being prepared for submission for publication of the material. Wong, Mark; Bewley, Thomas. The thesis author was the primary investigator and author of this material.

ABSTRACT OF THE THESIS

**Ensemble-Based Adaptive Observation**

by

Mark Wong
Master of Science in Engineering Sciences: Mechanical Engineering
University of California San Diego, 2018
Professor Thomas Bewley, Chair

Adaptive observation seeks to move sensor vehicles in order to accurately estimate and forecast the state of a system. This thesis seeks to formulate an adaptive observation algorithm around the Ensemble Kalman Filter. The Monte Carlo approach of the Ensemble Kalman filter allows for the approximation of the variance of the system estimate, which can be used to move the vehicles in a manner that minimizes this variance. After an introduction to the problem, this thesis gives a brief history of the Ensemble Kalman filter before describing the formulation of the adaptive observation algorithm. It then goes on to describe the numerical simulation setup that is used to perform experiments that test the algorithm's performance. The results show the success of my adaptive observation algorithm in reducing the variance of the system estimate and therefore the ability of my algorithm to produce an accurate estimate of a model two-dimensional convective flow.

# Chapter 1

# Introduction

Forecasting is an important tool in predicting future events from current knowledge. On a large scale, forecasting is used to predict atmospheric conditions and give insight into future weather and natural disaster possibilities. On a smaller scale, we can use forecasting to predict fluid flow in order to track something that is moving in that flow (i.e. radioactive plume in air, or oil spill in water). To do this there needs to be some kind of sensors that can measure the current state of a fluid flow and the presence of something in that flow. Traditionally, sensors are placed somewhere in the region that you are trying to measure. Although there is work that can describe optimal placement of these sensors, it is often the case that moving the sensors throughout the domain will help achieve a more accurate forecast with fewer sensors.

Adaptive observation is an important tool in the field of forecasting. As opposed to traditional observation where sensors are placed statically in a given space, adaptive observation uses moving sensors to collect information at different locations over time. Sensors are typically placed on some sort of vehicle that is fit for moving in the environment that they exist. The goal in adaptive observation is to control these sensor vehicles in a manner that minimizes the forecast error.

## 1.1 Motivation

The Coordinated Robotics Laboratory began research into the estimation and forecasting of environmental plumes several years ago. The original problem involved calculating unmanned aerial vehicle (UAV) flight trajectories to improve the estimation and forecast of airborne biological or chemical agents released in an urban area. Since then, the research scope has broadened to incorporate the dynamics of any type of sensor vehicle and the dynamics of any plume moving convectively in a chaotic flow field. Such plume dynamics can be seen in different types of environmental disasters such as a radioactive plume from nuclear reactor failure, oil spills, and biological weapon attacks. Similar dynamics can even be seen in the spread of wildfires, which has had increased interest as the western United States sees an increased risk of fire damage. All of these situations present a unique challenge because forecasting a fluid flow often requires complex dynamic modeling with relatively large uncertainties.

The scale at which environmental plumes tend to occur presents its own issues. Unlike global scale atmospheric or ocean patterns that may change over the course of a day or two, the time-scale of the change in fluid flow at a smaller scale is much faster and therefore on the same time-scale as sensor vehicles measuring the environment. This means that instead of planning way points that the sensor vehicles should get to at a certain hour or on a certain day, the trajectories of the vehicles should be planned for only the upcoming short period of time because the flow field may change.

## 1.2 Background

There are a number of theories on how to control sensor vehicles in adaptive observation, but they can generally be divided into two distinct methods: centralized and decentralized. The goal of many decentralized algorithms is to achieve the maximum coverage over an area. Algorithms do this by formulating an error function and moving the sensor vehicles in a negative gradient to optimally cover the planar area. Much work has been done to make decentralized algorithms applicable to real world environments by incorporating vehicle dynamics and sensor constraints [2].

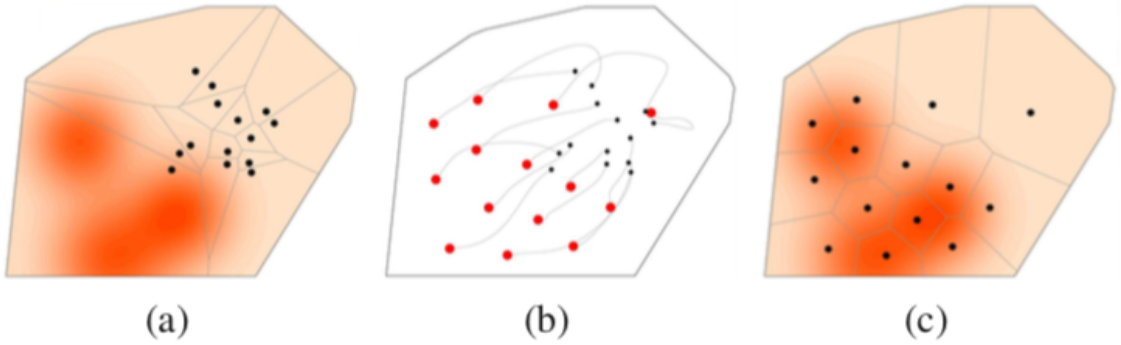


Figure 1.1: Example of how a coverage control algorithm works. a) Initial sensor vehicle positions, b) Sensor vehicle trajectories, c) Final sensor vehicle positions

The problem, however, is that decentralized methods do their computing on each individual sensor vehicle, which limits the complexity of the calculations that can be done and, therefore, has obvious limitations for large-scale complex systems such as environmental flows.

Centralized methods use a central computer that has access to all sensor information and can broadcast commands to the entire fleet of sensor vehicles. This allows for a more complex computing platform such as a cluster of supercomputers that are fit to handle high complexity, large-scale systems. Algorithms such as those described in [3] [4] and [5] use the higher capacity computing in order to properly position sensor vehicles when forecasting ocean environments. The algorithm in [3] uses a centralized adaptive observation algorithm to find the source of a chemical plume. The algorithm in [5] attempts to tackle the plume tracking problem; however, it relies on satellite data to determine the initial conditions of the plume and external sensors to develop the ocean forecast model. While these algorithms are promising for positioning vehicles over a longer horizon such as a full day, they do not consider situations in which the flow is evolving at a time scale similar to the vehicle movement.

This thesis is based on initial research done by David Zhang. In [1], Zhang outlines an algorithm that was developed to accomplish adaptive observation by seeking to minimize the variance of the system estimate for complex convective systems

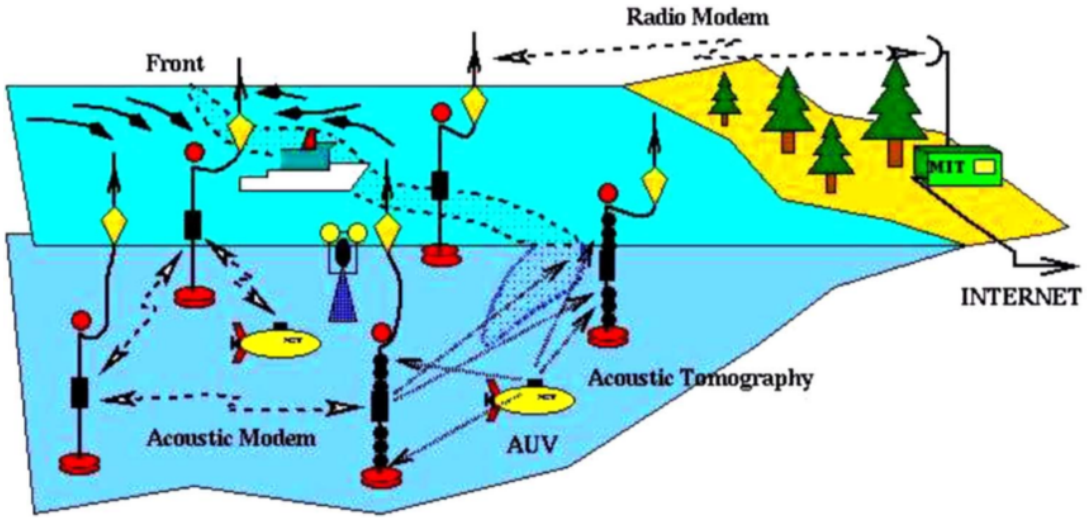


Figure 1.2: Example of how a network can be set up for use of centralized algorithms to control multiple vehicles.

with vehicles subject to specific vehicle dynamics. Zhang's previous work simplified the process of propagating the covariance matrix by simply growing the covariance linearly in time during the prediction horizon. The new method I am proposing is to implicitly propagate the covariance matrix using the Ensemble Kalman Filter based on the ensemble spread at each discrete time step. This new method more accurately describes the propagation of the covariance matrix by taking advantage of the information obtained by the measurement vehicles.

## 1.3 The Ensemble Kalman Filter

The Ensemble Kalman Filter (EnKF) has become a very useful tool since its introduction by Evensen in 1994 [6]. It has become a staple for data assimilation in the oceanic and atmospheric sciences communities. The EnKF was introduced as a computationally efficient alternative to the Extended Kalman Filter (EKF) for high dimensional systems. The EnKF takes a Monte Carlo approach by propagating an ensemble of perturbed trajectories of the system. Using this ensemble of trajectories,

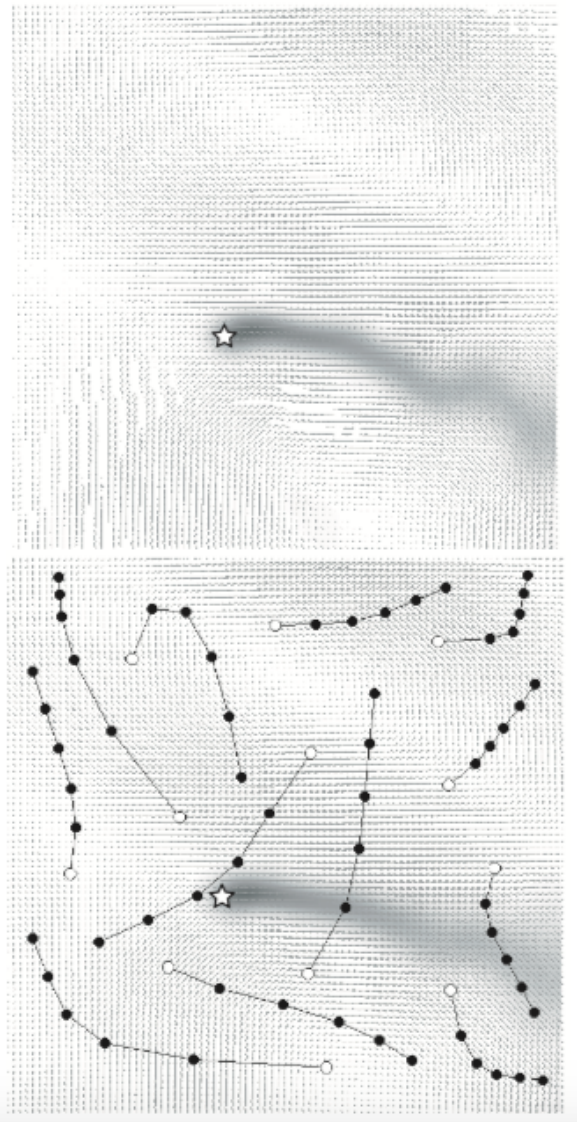

Figure 1.3: Simulation produced in previous work [1]. Top) Truth simulation of plume driven by a vector field. Bottom) Vehicle trajectories plotted on estimation of vector field and plume.

a low rank approximation of the error covariance can be calculated. This approximation approaches the exact error covariance as the number of ensemble members approaches infinity. The approximation of the error covariance is then used in an update similar to an update of a standard Kalman Filter (KF).

This thesis uses the EnKF in two ways. First it is used as it was intended (and as David Zhang used it in his work [1]), as a data assimilation technique to produce a system estimate from multiple sensor vehicles. Second, it is used in the formulation of the adaptive observation algorithm so that an approximation of the covariance matrix is available whenever it is needed during the computation of the gradient of the cost function. This new ensemble formulation of an adaptive observation algorithm is the innovation of this thesis.

As mentioned previously, past work [1] used the KF for discrete updates to the system during the prediction horizon of the adaptive observation algorithm. The problem with this method is that, to compute the gradient to optimize vehicle control, the full covariance matrix would need to be propagated in time during the prediction horizon. This is not feasible in real-world applications. The work-around used in previous work was to propagate the covariance matrix linearly through the prediction horizon.

### 1.3.1 Formulation

The EnKF is initialized with a state estimate $\hat{\mathbf{x}}$. An ensemble, $\hat{\mathbf{x}}^n$, is created with mean $\hat{\mathbf{x}}$ and covariance consistant with the error covariance of the estimate. The $N$ ensemble members are then propagated using a nonlinear model equation with the addition of random forcing $\mathbf{w}^n(t)$ that is statistically consistent with real state disturbances.

$$\frac{d\hat{\mathbf{x}}^n(t)}{dt} = \mathbf{f}(\hat{\mathbf{x}}(t),^n \mathbf{w}(t)^n) \tag{1.1}$$

At discrete times $t_k$, an observation $\mathbf{y}_k$ is taken. Each ensemble member is updated using this observation with the addition of random forcing $\mathbf{v}_k^n$ that is

statistically consistent with those of the actual measurement noise, $\mathbf{v}_k$ .

$$\hat{\mathbf{y}}_k^n = \mathbf{y}_k + \mathbf{v}_k^n \tag{1.2}$$

$$\hat{\mathbf{x}}_{k^+}^n = \hat{\mathbf{x}}_{k^-}^n + P_{k^-}^e H_k^T (H_k P_{k^-}^e H_k^T + R_k)^{-1} (\hat{\mathbf{y}}_k^n - H_k \hat{\mathbf{x}}_{k^-}^n) \tag{1.3}$$

where $H$ is the linearization of the output operator $h()$ which depends on the measurement sensor. This update is very similar to a KF or EKF update with two key differences. The first difference is the additional random forcing applied to the measurement of each ensemble member. The second is that the matrix $P^e$ is an estimate of the covariance matrix. Unlike the EKF, which uses a Riccati equation to propagate the covariance matrix $P$, the EnKF computes an estimate, $P^e$, based on the second moment of the ensemble members from the ensemble mean.

$$\bar{\mathbf{x}}(t) = \frac{1}{N} \sum \hat{\mathbf{x}}^n(t), \;\; \delta\hat{\mathbf{x}}^n(t) = \hat{\mathbf{x}}^n(t) - \bar{\mathbf{x}}(t), \;\; \delta\mathbf{X} = [\delta\hat{\mathbf{x}}^1 \;\; \delta\hat{\mathbf{x}}^2 \;\; ... \;\; \delta\hat{\mathbf{x}}^N] \tag{1.4}$$

$$P^e(t) = \frac{(\delta\mathbf{X})(\delta\mathbf{X})^T}{N-1} \tag{1.5}$$

In summary, the EnKF works very similarly to the KF and the EKF. The ensemble members $\hat{\mathbf{x}}^n(t)$ are propagated forward in time using the nonlinear model equation with state disturbances $\mathbf{w}^n(t)$ until a new measurement $\mathbf{y}_k$ is obtained. Then each ensemble member is updated to include this new information with measurement noise $\mathbf{v}_k^n$. The covariance matrix is not propagated as it is in the EKF. It is instead calculated implicitly using the evolution of the ensemble.

### 1.3.2 Practical Considerations

As mentioned, the EnKF can outperform the KF and EKF in high-dimensional systems. While this can be attributed to a few different aspects of the EnKF, it is important to note the computational benefit from the structure of the covariance estimate, as seen in equation 1.5. This structure allows for the separation of the $n$ x $n$ matrix $P^e(t)$ into two smaller matrices of size $n$ x $l$ and $m$ x $l$, where $n$ is the number of ensemble members and $l$ is the size of the state of the system. Typically in

estimation problems involving the EnKF systems, state size can be 10,000 or more, while the size of the ensemble is 2 to 3 orders of magnitude less. This can make a big difference in both computation time and storage requirements.

Another aspect to consider when implementing an EnKF is covariance localization. As mentioned earlier, the number of ensemble members is often 2-3 orders of magnitude or more smaller than the size of the state of the system. This results in an undersampling of the system and can lead to spurious correlations. Essentially, spurious correlations arise because there are not measurements in a grid close enough to comensate for the errors that arise in the covariance estimate. To counteract this problem, covaricance localization can be used to ensure that only grid points physically close to each other are correlated. Covariance localization is carried out by a scaling matrix $\rho$ as seen in equation 1.6.

$$\tilde{P}^e(t) = \rho \circ P^e(t) \tag{1.6}$$

(where $\circ$ represents the Schur product)

While there are multiple interpretations of how to calculate $\rho$, the idea is to create a function that results in the scaling matrix having values between 1 and 0. When grid points are physically close to each other, the corresponding element in $\rho$ will approach 1; and vice versa. The work in [7] presents a piecewise function to compute $\rho$, as seen in equation 1.7.

$$\rho = \begin{cases} -\frac{1}{4}(\frac{z}{c})^5 + \frac{1}{2}(\frac{z}{c})^4 + \frac{5}{8}(\frac{z}{c})^3 - \frac{5}{3}(\frac{z}{c})^2 + 1, & \text{if } 0 \leq z \leq c \\ \frac{1}{12}(\frac{z}{c})^5 - \frac{1}{2}(\frac{z}{c})^4 + \frac{5}{8}(\frac{z}{c})^3 + \frac{5}{3}(\frac{z}{c})^2 - 5(\frac{z}{c}) + 4 - \frac{2}{3}(\frac{c}{z}), & \text{if } c \leq z \leq 2c \\ 0, & \text{if } 2c \leq z \end{cases} \tag{1.7}$$

(where z is the distance between grid points and c is the desired localization variable)

As you can see from equation 1.6, covariance localization can be computa-

tionally inefficient because it involves the Schur product of two large matrices. Work such as in [8] has been done to formulate the EnKF in a new way in order to reduce the computational inefficiencies that occur when applying localization. In the experimentation done in this thesis, these methods were not used to increase efficiency because the test case we are using has a relatively small system state size (256).

### 1.3.3 Applications

As previously mentioned, the EnKF was created as an alternative to the EKF for high-dimensional systems. To compare the EnKF with the standard EKF there was initial testing done on highly nonlinear model equations such as the Lorenz Equation [9]. Soon after, researchers started applying the EnKF to problems in atmospheric and oceanic data assimilation [10] [11]. In [10], Mitchell and Houtekamer determined from simulations of a global atmospheric model that atmospheric flow could be estimated quite accurately from an EnKF with a relatively small number of ensemble members and 80,000 observations a day. The work in [12] shows that work has continued to progress in the atmospheric sciences, and the EnKF has become an integral tool in global forecasting.

More recent and novel work has taken the success of the EnKF in oceanic and atmospheric sciences and applied it to new fields. The work in [13] shows that the EnKF can be a useful data assimilation tool for crop yield estimation. The work in [14] uses the EnKF to estimate the state of a ballistic target during re-entry. Results from that paper show that the EnKF with an ensemble size larger than 25 significantly outperforms the EKF in estimating the position, velocity, and ballistic coefficient of an airborne ballistic missile.

These examples show that, much like the traditional KF, the EnKF can be used in a variety of ways. Creative uses can be found by understanding the advantages of the EnKF over other types of Kalman filters. In the case of this thesis, we take advantage of the error covariance approximation that naturally arises from computing an ensemble of system trajectories.

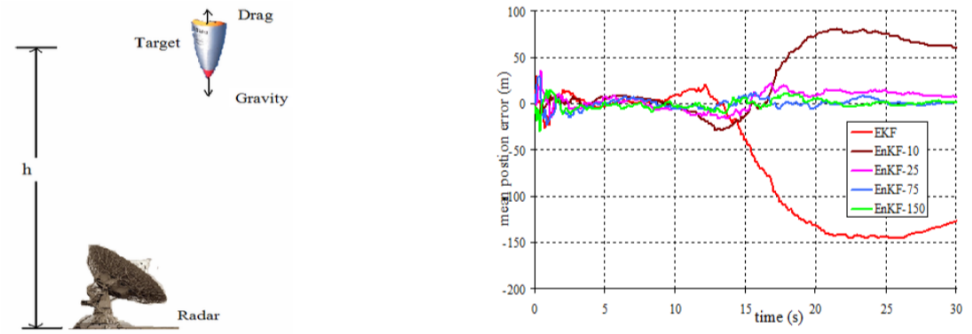


Figure 1.4: Ballistic target tracking using the Ensemble Kalman Filter. Left) Real world setup. Right) Filter position estimate error comparing the Extended Kalman Filter to the Ensemble Kalman Filter with various numbers of ensemble members.

# Chapter 2

# Ensemble Adaptive Observation

The Ensemble Adaptive Observation (EnAO) algorithm presented in this work is an algorithm that solves the problem of sensor vehicles estimating and forecasting the state of a plume moving convectively in a fluid. I begin this chapter by formulating the algorithm. I then introduce my experimental setup including the model equation used, the data assimilation process, and the implementation of the algorithm to compute vehicle trajectories. The chapter ends with details of the results from this experimentation.

## 2.1 Formulation

Assume a continuous time (CT) discretization $\mathbf{x}(t)$ of an infinite-dimensional system describing, e.g., a flow of air and smoke over a domain $\Omega$, excited by state disturbances $\mathbf{w}(t)$ [CT, white, zero mean, and spectral density $Q(t)$], governed by:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}(t), \mathbf{w}(t)) \tag{2.1}$$

Assume also that there are $M$ sensor vehicles ($m = 1, \ . \ . \ . \ , M$ ) moving (in CT) throughout $\Omega$, with positions ${}^m\mathbf{q}(t)$ and control inputs ${}^m\mathbf{u}(t)$, and taking measurements ${}^m\mathbf{y}_k$ (in discrete time (DT) at times $t_k = kh$ with $h = T/K$ for $k =$ 1,...,$K$) corrupted by measurement noise ${}^m\mathbf{v}_k$ [DT, white, Gaussian, zero mean, and

covariance ${}^{m}R_k({}^{m}\mathbf{q}_k)$], according to:

for $m$ = 1,...,$M$:

$$\frac{d^{m}\mathbf{q}(t)}{dt} = \mathbf{g}({}^{m}\mathbf{q}(t), {}^{m}\,\mathbf{u}(t)) \tag{2.2}$$

$${}^{m}\mathbf{y}_k = {}^{m}\mathbf{h}_k(\mathbf{x}_k, {}^{m}\mathbf{q}_k) + {}^{m}\mathbf{v}_k = {}^{m}H_k({}^{m}\mathbf{q}_k)\mathbf{x}_k + {}^{m}\mathbf{v}_k \tag{2.3}$$

That is, we assume the (DT) measurements ${}^{m}\mathbf{h}_k(\mathbf{x}_k, {}^{m}\mathbf{q}_k)$ are linear functions of the state $\mathbf{x}(t_k) = \mathbf{x}_k$, but both the measurement matrix ${}^{m}H_k({}^{m}\mathbf{q}_k)$ and the covariance matrix ${}^{m}R_k({}^{m}\mathbf{q}_k)$ corresponding to the $m$th sensor vehicle are functions of its configuration (position, heading, velocity) at time $t_k$, denoted ${}^{m}\mathbf{q}(t_k) = {}^{m}\mathbf{q}_k$ [which, in turn, can be changed by modifying the control inputs ${}^{m}\mathbf{u}(t)$ in (2.2)]. It is assumed for the purpose of this derivation that the CT trajectories of the sensor vehicles can be modeled accurately, and thus a disturbance input to (2.2) is not included in the model, and an estimator for the vehicle positions ${}^{m}\mathbf{q}(t)$ is not needed.

The Ensemble Kalman filter assumes there are $N$ ensemble members ($n$ = 1,...,$N$), with individual ensemble members $\hat{\mathbf{x}}^n(t)$ evolving according to the modeled state equation (2.1) with excitation by $\hat{\mathbf{w}}^n(t)$, and individual ensemble measurements ${}^{m}\hat{\mathbf{y}}_k^n$ are taken according to (2.2)-(2.3) with additional excitation by ${}^{m}\hat{\mathbf{v}}_k^n$ in each update, where $\hat{\mathbf{w}}^n(t)$ and ${}^{m}\hat{\mathbf{v}}_k^n$ are generated in a manner that is statistically consistent with the models of the state disturbances $\mathbf{w}(t)$ and measurement noise ${}^{m}\mathbf{v}_k$ (i.e., with spectral density $Q(t)$ and covariance ${}^{m}R_k({}^{m}\mathbf{q}_k)$, and zero mean):

for $n$ = 1,...,$N$:

$$\frac{d\mathbf{x}^n(t)}{dt} = \mathbf{f}(\hat{\mathbf{x}}^n(t), \hat{\mathbf{w}}^n(t)), {}^{m}\hat{\mathbf{y}}_k^n = {}^{m}\mathbf{y}_k + {}^{m}\hat{\mathbf{v}}_k^n \tag{2.4}$$

(between measurement times $t_k$)

$$\hat{\mathbf{y}}_k^n = [{}^{1}\hat{\mathbf{y}}_k^n; {}^{2}\,\hat{\mathbf{y}}_k^n; ...; {}^{M}\hat{\mathbf{y}}_k^n], \hat{\mathbf{v}}_k^n = [{}^{1}\hat{\mathbf{v}}_k^n; {}^{2}\,\hat{\mathbf{v}}_k^n; ...; {}^{M}\hat{\mathbf{v}}_k^n] \tag{2.5}$$

$$H_k = [{}^{1}H_k; {}^{2}H_k; ...; {}^{M}H_k], R_k = diag[{}^{1}R_k; {}^{2}R_k; ...; {}^{M}R_k], \tag{2.6}$$

$$\hat{\mathbf{x}}_{k^+}^n = \hat{\mathbf{x}}_{k^-}^n + P_{k^-}^e H_k^T (H_k P_{k^-}^e H_k^T + R_k)^{-1} (\hat{\mathbf{y}}_k^n - H_k \hat{\mathbf{x}}_{k^-}^n) \tag{2.7}$$

(at measurement times $t_k$)

where $\hat{\mathbf{x}}^n_{k^-}$ denotes the value of $\hat{\mathbf{x}}^n$ at time $t$ before its DT update during its forward march, $\hat{\mathbf{x}}^n_{k^+}$ is its value after this update, and the low-rank ensemble approximation $P^e(t)$ of the covariance $P(t) = \mathcal{E}[[\mathbf{x}(t) - \bar{\mathbf{x}}(t)][\mathbf{x}(t) - \bar{\mathbf{x}}(t)]^T]$ is

$$\bar{\mathbf{x}}(t) = \frac{1}{N}\sum \hat{\mathbf{x}}^n(t), \;\; \delta\hat{\mathbf{x}}^n(t) = \hat{\mathbf{x}}^n(t) - \bar{\mathbf{x}}(t), \;\; \delta\mathbf{X} = [\hat{\mathbf{x}}^1 \;\; \delta\hat{\mathbf{x}}^2 \;\; ... \;\; \delta\hat{\mathbf{x}}^N] \tag{2.8}$$

$$P^e(t) = \frac{(\delta\mathbf{X})(\delta\mathbf{X})^T}{N-1} \tag{2.9}$$

The indices have been put in distinct locations in order to keep the notation clear (as much as possible); for example, the vector ${}^m\hat{\mathbf{y}}^n_k$ represents the value of $\hat{\mathbf{y}}$ for the $m$th vehicle, the $n$th ensemble member, and the $k$th timestep. Starting from the current time (taken as $t = 0$ in the discussion that follows), [1], which assumed a different model for the evolution of P and used a slightly different notation, developed an iterative framework to optimize the trajectory of the sensor vehicles to minimize the uncertainty of the state estimate at time $T$ . We follow the general approach of that paper here, modifying this formulation to apply to the Ensemble Kalman formulation (2.4)-(2.7) for the evolution of $P^e(t)$. To accomplish this, consider the problem of minimizing a cost function $J$ with respect to the control inputs ${}^m\mathbf{u}(t)$ over the time interval $t \;\epsilon\; (0, T)$, which, taking $Z > 0$, we write here in the form

$$J = \frac{1}{2} trace[P^e(T)] + \frac{1}{2}\sum_{m=1}^{M}\int_0^T {}^m\mathbf{u}^T(t) Z\, {}^m\mathbf{u}(t) dt \tag{2.10}$$

Note that $J$ quantifies the forecast uncertainty (i.e., the trace of the ensemble approximation of the covariance matrix at time $T$). We now lay out the equations to optimize the control inputs ${}^m\mathbf{u}(t)$ over the interval $(0, T)$ in order to minimize $J$ via an iterative adjoint-based updating strategy (a.k.a. model predictive control).

Applying perturbations ${}^m\mathbf{u}'$ to the set of control inputs ${}^m\mathbf{u}$ over $t \;\epsilon\; [0, T]$ causes the following chain reaction:

1. perturbations ${}^m\mathbf{q}'_k$ to the $M$ vehicle state vectors ${}^m\mathbf{q}_k$,

2. perturbations ${}^mH'_k = (d^mH_k/d^m\mathbf{q}_k){}^m\mathbf{q}'_k$ to the measurement operators ${}^mH_k$,

3. perturbations ${}^mR'_k = (d^mR_k/d^m\mathbf{q}_k){}^m\mathbf{q}'_k$ to the covariance of the measurement noise ${}^mR_k$,

4. perturbations $\hat{\mathbf{x}}'_n$ to the $N$ ensemble members $\hat{\mathbf{x}}_n$,

5. perturbations $P^{e\prime}$ to the ensemble approximation $P^e$ of the covariance, and, ultimately,

6. perturbations $J'$ to the cost function $J$.

Adjoint arithmetic may now be used to trace back through this cascade, in order to write

$$J' = \sum_{m=1}^{M} \int_0^T [{}^m\mathbf{g}]^T {}^m\mathbf{u}' dt \tag{2.11}$$

thus identifying the gradient, ${}^m\mathbf{g}(t)$, of the cost $J$ with respect to the control inputs ${}^m\mathbf{u}(t)$ over $t \ \epsilon \ (0,T)$; gradient-based optimization may then be used to minimize $J$ with respect to ${}^m\mathbf{u}(t)$ over this interval. To accomplish this, we first need to write the first-order perturbations of (2.2)-(2.10). Applying the chain rule for differentiation gives the following relations quantifying the chain reaction mentioned above, all of which are linear in the primed quantities:

$$\frac{d^m\mathbf{q}'(t)}{dt} = {}^mA\,{}^m\mathbf{q}'(t) + {}^mB\,{}^m\mathbf{u}'(t), \ \ {}^m\mathbf{q}'(0) = 0, \ \ {}^mA = \frac{\partial \mathbf{g}}{\partial {}^m\mathbf{q}}, \ \ {}^mB = \frac{\partial \mathbf{g}}{\partial {}^m\mathbf{u}}, \tag{2.12}$$

$$ {}^m\mathbf{y}'_k = {}^mH'_k\mathbf{x}_k, \ \ {}^mH'_k = \frac{d^mH_k}{d^m\mathbf{q}_k}{}^m\mathbf{q}'_k,$$

$$ {}^mR' = \frac{d^mR_k}{d^m\mathbf{q}_k}{}^m\mathbf{q}'_k, \ \ R'_k = diag[{}^1R'_k; {}^2R'_k; ...; {}^MR'_k], \tag{2.13}$$

$$\frac{d\hat{\mathbf{x}}^{n\prime}(t)}{dt} = \hat{A}^n\hat{\mathbf{x}}^{n\prime}(t), \ \ \hat{\mathbf{x}}^{n\prime}(t^+_{k-1}) = \hat{\mathbf{x}}^{n\prime}{}_{(k-1)^+}, \ \ \hat{A}^n = \frac{\partial \mathbf{f}}{\partial \hat{\mathbf{x}}^n}, \tag{2.14}$$

(between times $t^+_{k-1}$ and $t^-_k$ for $k$=1,...,$K$)

$$\bar{\mathbf{x}}' = \frac{1}{N}\sum \hat{\mathbf{x}}^{n\prime}, \ \ \delta\hat{\mathbf{x}}^{n\prime} = \hat{\mathbf{x}}^{n\prime} - \bar{\mathbf{x}}', \ \ \delta\mathbf{X}' = [\delta\hat{\mathbf{x}}^{1\prime} \ \ \delta\hat{\mathbf{x}}^{2\prime} \ \ ... \ \ \delta\hat{\mathbf{x}}^{N\prime}] \tag{2.15}$$

$$P^{e\prime} = \frac{(\delta\mathbf{X}')(\delta\mathbf{X})^T + (\delta\mathbf{X})(\delta\mathbf{X}')^T}{N-1} \tag{2.16}$$

$$\hat{\mathbf{x}}^n_{k^+}{}' = L^{1,n}_{k^-}(\hat{\mathbf{x}}_{k^-}{}') + L^{2,n}_{k^-}(\mathbf{y}'_k) + L^{3,n}_{k^-}(P^e_{k^-}) + L^{4,n}_{k^-}(H'_k) + L^{5,n}_{k^-}(R'_k), \tag{2.17}$$

(at times $t_k$ for $k$=1,...,$K$)

$$J' = \frac{1}{2} trace(P^e_{K^+}{}') + \sum_{m=1}^{M} \int_0^T {}^m\mathbf{u}^T(t) Z^m \mathbf{u}(t) dt \tag{2.18}$$

where the five linear operators $L^i_k(.)$ in (2.17) may be derived as follows: noting (2.7) and applying the chain rule together with the fact that
$\partial A^{-1}/\partial\alpha = -A^{-1}(\partial A/\partial\alpha)A^{-1}$, and thus $(A^{-1})' = -A^{-1}A'A^{-1}$, we may write

$$\hat{\mathbf{x}}^n_{k^+}{}' = \hat{\mathbf{x}}^n_{k^-}{}' + P^e_{k^-}{}' H^T_k D_{k^-} \mathbf{e}^n_k + P^e_{k^-}{}' (H'_k)^T D_{k^-} \mathbf{e}^n_k + C_{k^-} D_{k^-} (\mathbf{y}'_k - H_k \hat{\mathbf{x}}^n_{k^-}{}' - H'_k \hat{\mathbf{x}}^n_{k^-})$$

$$-C_{k^-} D_{k^-} [H'_k P^e_{k^-} H^T_k + H_k P^e_{k^-}{}' H^T_k + H_k P^e_{k^-} (H'_k)^T + R'_k] D_{k^-} \mathbf{e}^n_k$$

where $C_{k^-} = P^e_{k^-} H^T_k$, $D_{k^-} = (H_k P^e_{k^-} H^T_k + R_k)^{-1}$, and $\mathbf{e}^n_{k^-} = \hat{\mathbf{y}}^n_k - H_k \hat{\mathbf{x}}^n_{k^-}$; defining $E_{k^-} = I - C_{k^-} D_{k^-} H_k$ and $\mathbf{f}^n_{k^-} = P^e_{k^-} H^T_k D_{k^-} \mathbf{e}^n_{k^-} + \hat{\mathbf{x}}^n_{k^-}$, the linear operators $L^{i,n}_{k^-}(.)$ in (2.17) may thus be written

$$L^{1,n}_{k^-}(\hat{\mathbf{x}}_{k^-}{}') = E_{k^-} \hat{\mathbf{x}}^n_{k^-}{}', \;\; L^{2,n}_{k^-}(\mathbf{y}'_k) = C_{k^-} D_{k^-} \mathbf{y}'_k, \;\; L^{3,n}_{k^-}(P^e_{k^-}{}') = E_{k^-} P^e_{k^-}{}' H^T_k D_{k^-} \mathbf{e}^n_{k^-}, \tag{2.19}$$

$$L^{4,n}_{k^-}(H'_k) = E_{k^-} P^e_{k^-} (H'_k)^T D_{k^-} \mathbf{e}^n_{k^-} - C_{k^-} D_{k^-} H'_k \mathbf{f}^n_{k^-}, \;\; L^{5,n}_{k^-}(R'_k) = C_{k^-} D_{k^-} R'_k D_{k^-} \mathbf{e}^n_{k^-} \tag{2.20}$$

Noting (2.15)-(2.16) allows us to write the first term of (2.18) as

$$\frac{1}{2} trace(P^e_{K^+}{}') = \sum_{n=1}^{N} (\mathbf{s}^n)^T \hat{\mathbf{x}}^n_{K^+}{}' \tag{2.21}$$

where $\mathbf{s}^n = \frac{1}{N-1}(\hat{\mathbf{x}}^n_{K^+} - \bar{\mathbf{x}}_{K^+})$

Noting (2.13), (2.15)-(2.16), and (2.19)-(2.20) and taking $\mathbf{q} = [{}^1\mathbf{q}; {}^2\mathbf{q}; ...; {}^M\mathbf{q}]$

allows us to rewrite (2.17), leveraging the rank-3 tensors $\Delta_q H$ and $\Delta_q R$, as

$$\hat{\mathbf{x}}^n_{k^+}{}' = E_{k^-}\hat{\mathbf{x}}^n_{k^-}{}' + C_{k^-}D_{k^-}H'_k(\mathbf{x}_k - \mathbf{f}^n_{k^-}) + E_{k^-}P^e_{K^-}(H'_k)^T D_{k^-}\mathbf{e}^n_{k^-}$$

$$+E_{k^-}P^e_{K^-}{}'H^T_k D_{k^-}\mathbf{e}^n_{k^-} + C_{k^-}D_{k^-}R'_k D_{k^-}\mathbf{e}^n_{k^-}$$

$$= L^{6,n}_{k^-}(\hat{\mathbf{x}}'_{k^-}) + L^{7,n}_{k^-}(\mathbf{q}'_{k^-})$$

where

$$P^{e\prime} = \sum_{l=1}^{N}(\hat{\mathbf{x}}^{l\prime}[\delta\hat{\mathbf{x}}^l]^T + \delta\hat{\mathbf{x}}^l[\hat{\mathbf{x}}^{l\prime}]^T), \;\; \hat{\mathbf{x}}'_{k^-} = [\hat{\mathbf{x}}^1_{k^-}{}'; \hat{\mathbf{x}}^2_{k^-}{}'; ...; \hat{\mathbf{x}}^N_{k^-}{}']$$

$$H' = \frac{dH}{d\mathbf{q}} \cdot \mathbf{q}' = \Delta_{\mathbf{q}} H \cdot \mathbf{q}'$$

$$R' = \frac{dR}{d\mathbf{q}} \cdot \mathbf{q}' = \Delta_{\mathbf{q}} R \cdot \mathbf{q}'$$

thus,

$$L^{6,n}_{k^-}(\hat{\mathbf{x}}_{k^-}{}') = E_{k^-}\hat{\mathbf{x}}^n_{k^-}{}' + E_{k^-}\sum_{l=1}^{N}(\hat{\mathbf{x}}^{l\prime}[\delta\hat{\mathbf{x}}^l]^T + \delta\hat{\mathbf{x}}^l[\hat{\mathbf{x}}^{l\prime}]^T)H^T_k D_{k^-}\mathbf{e}^n_{k^-} \tag{2.22}$$

$$L^{7,n}_{k^-}(\hat{\mathbf{q}}'_k) = C_{k^-}D_{k^-}(\Delta_{\mathbf{q}}H_k \cdot \mathbf{q}'_k)(\mathbf{x}_k - \mathbf{f}^n_{k^-}) + E_{k^-}P^e_{k^-}(\Delta_{\mathbf{q}}H_k \cdot \mathbf{q}'_k)^T D_{k^-}\mathbf{e}^n_{k^-}$$

$$-C_{k^-}D_{k^-}(\Delta_{\mathbf{q}}R_k \cdot \mathbf{q}'_k)D_{k^-}\mathbf{e}^n_{k^-} \tag{2.23}$$

Taking $\hat{\mathbf{z}}_{k^+} = [\hat{\mathbf{z}}^1_{k^+}; \hat{\mathbf{z}}^2_{k^+}; ...; \hat{\mathbf{z}}^N_{k^+}]$, we will also make use below of the adjoints of the linear operators $L^{6,n}_{k^-}(\cdot)$and $L^{7,n}_{k^-}(\cdot)$, defined as follows

$$\sum_{n=1}^{N}[\hat{\mathbf{z}}^n_{k^+}]^T L^{6,n}_{k^-}(\hat{\mathbf{x}}'_{k^-}) = \sum_{n=1}^{N}[L^{6,n*}_{k^-}(\hat{\mathbf{z}}_{k^+})]^T\hat{\mathbf{x}}^n_{k^-}{}', \;\; \sum_{n=1}^{N}[\hat{\mathbf{z}}^n_{k^+}]^T L^{7,n}_{k^-}(\hat{\mathbf{q}}'_k) = [L^{7*}_{k^-}(\hat{\mathbf{z}}_{k^+})]^T\hat{\mathbf{q}}'_k, \tag{2.24}$$

thus,

$$L^{6,n*}_{k^-}(\hat{\mathbf{z}}_{k^+}) = E^T_{k^-}\hat{\mathbf{z}}^n_{k^+} + E^T_{k^-}\sum_{l=1}^{N}\hat{\mathbf{z}}^l_{k^+}[\delta\hat{\mathbf{x}}^n_{k^-}]^T H^T_k D_{k^-}\mathbf{e}^n_{k^-} + E^T_{k^-}\sum_{l=1}^{N}\delta\hat{\mathbf{x}}^n_{k^-}[\hat{\mathbf{z}}^l_{k^+}]^T H^T_k D_{k^-}\mathbf{e}^n_{k^-} \tag{2.25}$$

$$L^{7*}_{k^-}(\hat{\mathbf{z}}_{k^+}) = \sum_{n=1}^{N}[(\mathbf{x}_k - \mathbf{f}^n_{k^-})^T(\Delta_{\mathbf{q}}H_k\cdot)^T D^T_{k^-}C^T_{k^-}\hat{\mathbf{z}}_{k^+} + [\mathbf{e}^n_{k^-}]^T D^T_{k^-}(\Delta_{\mathbf{q}}H_k\cdot)[P^e_{k^-}]^T[E_{k^-}]^T\hat{\mathbf{z}}_{k^+}$$

$$-[\mathbf{e}^n_{k^-}]^T D^T_{k^-}(\Delta_{\mathbf{q}}R_k\cdot)^T D^T_{k^-}C^T_{k^-}\hat{\mathbf{z}}_{k^+}] \tag{2.26}$$

where we have introduced the notation $[\mathbf{a}^T(\Delta B\cdot)^T\mathbf{c}]^T\mathbf{q}' = \mathbf{c}^T(\Delta B\cdot\mathbf{q}')\mathbf{a}$
and $[\mathbf{a}^T(\Delta B\cdot)\mathbf{c}]^T\mathbf{q}' = \mathbf{c}^T(\Delta B\cdot\mathbf{q}')^T\mathbf{a}$

The perturbation problem (2.12)-(2.18) now reduces to significantly simplified form. The perturbations of the states of the sensor vehicles, $\mathbf{q}'(t)$, evolve in CT from $t = 0$ to $t = T$, forced by $\mathbf{u}'(t)$ according to the CT system

$$\frac{d^m\mathbf{q}'(t)}{dt} = {}^mA(t){}^m\mathbf{q}'(t) + {}^mB{}^m\mathbf{u}'(t),\ \ {}^m\mathbf{q}'(0) = 0,\ \Rightarrow {}^m\mathcal{L}(t){}^m\mathbf{q}'(t) = {}^mB(t){}^m\mathbf{u}'(t),$$

$${}^m\mathcal{L}(t) = \frac{d}{dt} - {}^mA(t) \tag{2.27}$$

while the perturbations of the ensemble members, $\hat{\mathbf{x}}^{n\prime}(t)$, evolve in a mixed CT/DT framework over the same time interval, forced by the ${}^m\mathbf{q}'(t)$: defining $\hat{\mathbf{x}}^n_{0|0}{}' = 0$, they evolve between $t^+_{k-1}$ and $t^-_k$(for $k = 1, ..., K$) according to

$$\frac{\hat{\mathbf{x}}^{n\prime}(t)}{dt} = \hat{A}^n(t)\hat{\mathbf{x}}^{n\prime}(t),\ \ \hat{\mathbf{x}}^{n\prime}(t^+_{k-1}) = \hat{\mathbf{x}}^n_{(k-1)^+}{}',\ \Rightarrow \hat{\mathcal{L}}^n(t)\hat{\mathbf{x}}^{n\prime}(t) = 0,$$

$$\hat{\mathcal{L}}^n(t) = \frac{d}{dt} - \hat{A}^n(t) \tag{2.28}$$

and, upon completion of the $k$'th march of (2.28), defining $\hat{\mathbf{x}}^n_{k^-}{}' = \hat{\mathbf{x}}^{n\prime}(t_k)$, are updated at time $t_k$ according to

$$\hat{\mathbf{x}}^n_{k^+}{}' = L^{6,n}_{k^-}(\hat{\mathbf{x}}_{k^-}{}') + L^{7,n}_{k^-}(\hat{\mathbf{q}}'_k) \tag{2.29}$$

finally, the corresponding perturbation to the cost function is given by

$$J' = \sum_{n=1}^{N}(\mathbf{s}^n)^T\hat{\mathbf{x}}^n_{K^+}{}' + \sum_{m=1}^{M}\int_0^T {}^m\mathbf{u}^T(t)Z^m\mathbf{u}'(t)dt \tag{2.30}$$

We have thus identified a simplified cascade of linear relations,(2.27)-(2.30),

that relate any perturbation of the controls ${}^m\mathbf{u}'$ to the corresponding cost perturbation $J'$. What remains is to pose the appropriate adjoint identities to turn these four relations around, thereby expressing $J'$ in the form given in (2.11) to identify the gradient ${}^m\mathbf{g}(t)$. To proceed, denote ${}^m\mathbf{r}(t)$ as the adjoint of ${}^m\mathbf{q}'(t)$ and $\hat{\mathbf{z}}^n(t)$ as the adjoint of $\hat{\mathbf{x}}^{n\prime}(t)$, take $\mathbf{r} = [{}^1\mathbf{r};{}^2\mathbf{r};...;{}^M\mathbf{r}]$, and define the necessary duality pairings and adjoint identities piecewise on each time interval from $t_{k-1}^+$ to $t_k^-$ as follows:

$$\langle {}^m\mathbf{r}(t), {}^m\mathbf{q}'(t)\rangle_k = \int_{t_{k-1}}^{t_k} [{}^m\mathbf{r}(t)]^T {}^m\mathbf{q}'(t)dt,$$

$$\langle {}^m\mathbf{r}(t), {}^m\mathcal{L}(t){}^m\mathbf{q}'(t)\rangle_k = \langle {}^m\mathcal{L}^*(t){}^m\mathbf{r}(t), {}^m\mathbf{q}'(t)\rangle_k + {}^m b_k \tag{2.31}$$

$$\Rightarrow {}^m\mathcal{L}^*(t) = -\frac{d}{dt} - [{}^mA(t)]^T, \;\; {}^m b_k = [{}^m\mathbf{r}_{k^-}]^T {}^m\mathbf{q}'_k - [{}^m\mathbf{r}_{(k-1)^+}]^T {}^m\mathbf{q}'_{k-1} \tag{2.32}$$

$$\langle \hat{\mathbf{z}}^n(t), \hat{\mathbf{x}}^{n\prime}(t)\rangle_k = \int_{t_{k-1}}^{t_k} [\hat{\mathbf{z}}^n(t)]^T \hat{\mathbf{x}}^{n\prime}(t)dt,$$

$$\langle \hat{\mathbf{z}}^n(t), \hat{\mathcal{L}}^n(t)\hat{\mathbf{x}}^{n\prime}(t)\rangle_k = \langle \hat{\mathcal{L}}^{n*}(t)\hat{\mathbf{z}}^n(t), \hat{\mathbf{x}}^{n\prime}(t)\rangle_k + \hat{b}_k^n \tag{2.33}$$

$$\Rightarrow \hat{\mathcal{L}}^{n*}(t) = -\frac{d}{dt} - [\hat{A}^n(t)]^T, \;\; \hat{b}_k^n = [\hat{\mathbf{z}}_{k^-}^n]^T \hat{\mathbf{x}}_{k_-}^{n\,\prime} - [\hat{\mathbf{z}}_{(k-1)^+}^n]^T \hat{\mathbf{x}}_{(k-1)^+}^{n\,\prime} \tag{2.34}$$

Leveraging the adjoint operators ${}^m\mathcal{L}^*(t)$ and $\mathcal{L}^{n*}(t)$ and identities above, we now perform the necessary adjoint analysis. To begin, initialize $\mathbf{r}_{K^+} = 0$ and $\hat{\mathbf{z}}_{K^+}^n = \mathbf{s}$. Then, for $k = K, K-1, ..., 1$ consider the DT updates

$$\mathbf{r}_{k^-} = \mathbf{r}_{k^+} + L_{k^-}^{7*}(\hat{\mathbf{z}}_{k^+}), \;\; \hat{\mathbf{z}}_{k^-} = L_{k^-}^{6,n*}(\hat{\mathbf{z}}_{k^+}) \tag{2.35}$$

coupled with the appropriate CT marches of ${}^m\mathbf{r}(t)$ and $\hat{\mathbf{z}}^n(t)$ in reverse time on each interval, from $t = t_k^-$ to $t = t_{k-1}^+$ for $k = K, K-1, ..., 1$, as follows:

$${}^m\mathcal{L}^*(t){}^m\mathbf{r}(t) = 0 \Rightarrow -\frac{d{}^m\mathbf{r}(t)}{dt} = [{}^mA(t)]^T {}^m\mathbf{r}(t), \;\; {}^m\mathbf{r}(t_k^-) = {}^m\mathbf{r}_{k^-} \tag{2.36}$$

$$\hat{\mathcal{L}}^{n*}(t)\hat{\mathbf{z}}^n(t) = 0 \Rightarrow -\frac{d\hat{\mathbf{z}}^n(t)}{dt} = [\hat{A}^n(t)]^T \hat{\mathbf{z}}^n(t), \;\; \hat{\mathbf{z}}^n(t_k^-) = \hat{\mathbf{z}}_{k^-}^n \tag{2.37}$$

where ${}^m\mathbf{r}_{k^+} = {}^m\mathbf{r}(t_k^+)$ and $\hat{\mathbf{z}}_{k^+}^n = \hat{\mathbf{z}}(t_k^+)$ indicate the values of ${}^m\mathbf{r}$ and $\hat{\mathbf{z}}^n$ at time $t_k^+$,

just before their $k$'th DT updates on this backward march, and ${}^m\mathbf{r}_{k^-} = {}^m\mathbf{r}(t_{k^-})$ and $\hat{\mathbf{z}}^n_{k^-} = \hat{\mathbf{z}}^n(t_{k^-})$ indicate their values at $t_k^-$, just after these updates.

Note that, by (2.29), (2.35), and (2.24) it follows that

$$\sum_{m=1}^{M}[{}^m\mathbf{r}_{k^+}]^T({}^m\mathbf{q}'_k) + \sum_{n=1}^{N}[\hat{\mathbf{z}}^n_{k^+}]^T(\hat{\mathbf{x}}^n_{k^+}{}')$$

$$= \sum_{m=1}^{M}[{}^m\mathbf{r}_{k^-}]^T({}^m\mathbf{q}'_k) + \sum_{n=1}^{N}[\hat{\mathbf{z}}_{k^-}]^T\hat{\mathbf{x}}^n_{k^-}{}' \tag{2.38}$$

This fact, together with the initial conditions ${}^m\mathbf{q}'_0 = 0$ and $\hat{\mathbf{x}}^n_{0^+}{}' = 0$, the terminal conditions ${}^m\mathbf{r}_{K^+} = 0$ and $\hat{\mathbf{z}}^n_{K^+} = \mathbf{s}^n$, the perturbation equations (2.27)-(2.29), and the adjoint equations (2.35)-(2.37) allow us to leverage the identities in (2.31)-(2.34) to express the first term in the cost perturbation $J'$ in (2.11) in the desired form, thus identifying the gradient:

$$\sum_{n=1}^{N}[\mathbf{s}^n]^T\hat{\mathbf{x}}^n_{K^+}{}' = \sum_{m=1}^{M}\int_0^T [{}^m\mathbf{r}(t)]^T{}^mB(t){}^m\mathbf{u}'(t)dt$$

$$\Rightarrow J' = \sum_{m=1}^{M}\int_0^T [{}^mB^T(t){}^m\mathbf{r}(t) + Z^m\mathbf{u}(t)]^T{}^m\mathbf{u}'(t)dt \tag{2.39}$$

$$\Rightarrow {}^m\mathbf{g}(t) = {}^mB^T(t){}^m\mathbf{r}(t) + Z^m\mathbf{u}(t) \tag{2.40}$$

where the adjoint $({}^m\mathbf{r}(t), \hat{\mathbf{z}}^n(t))$ is determined from the piecewise-continuous backward-in-time march given in (2.35)-(2.37), the operators of which are functions of the result of the piecewise-continuous forward-in-time march of the system $({}^m\mathbf{q}(t), \hat{\mathbf{x}}^n(t))$ defined by (2.2)-(2.10).

## 2.2 Experimentation

To test this EnAO algorithm, a simulation was set up using MATLAB. Although MATLAB lacks the speed and portability of languages like C or Fortran, it was chosen for ease of use and was found to be sufficient for this small proof of

concept experiment.

The simulation structure is as follows:

1. A truth simulation of a model equation is initialized and propagated in time using tested numerical methods. This experiment uses the two-dimensional Convective Cahn-Hilliard Equation as a model equation and simulates it using appropriate spectral methods.

2. Sensor vehicles are placed in the truth simulation with the ability to measure the scalar value of the model equation at the vehicle's current location in the grid. The measurements taken have additional zero mean white Gaussian measurement noise $v_k$ with covariance $R_k$. These vehicles have the ability to move with point mass dynamics with damping and their control inputs come from the EnAO algorithm.

3. An EnKF is used to create an estimate of the model using the measurement data from the sensor vehicles. All ensembles are initialized as zero and are updated according to the measurement data and propagated according to the model equation and additional zero mean white disturbances $w(t)$ with spectral density $Q(t)$. In practice, this is done in discrete time with disturbances $w_k$ at each time step.

4. The EnAO algorithm runs to optimize the vehicle trajectory over a specified time horizon. The optimization attempts to minimize the variance of the ensemble that is used to estimate the model equation. Gradient-based minimization methods are used to minimize the cost function.

5. When the EnAO algorithm is finished, it sends the control inputs to the sensor vehicles.

6. The model equation marches forward in time using the chosen numerical methods and the vehicles march forward in time with the new control inputs and specified vehicle dynamics.

7. The simulation continues in a similar manner with the EnAO computing control inputs, the truth simulation marching in time, and the EnKF estimating the truth simulation.

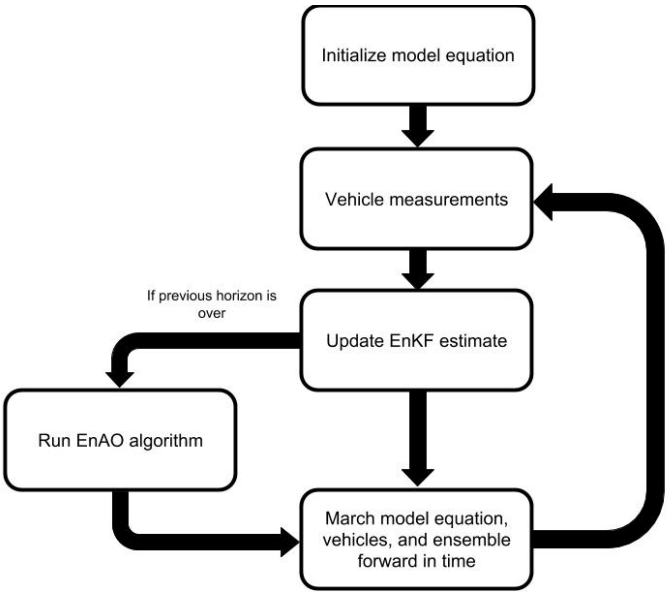


Figure 2.1: A figure describing the steps of the simulation process.

You can see that this experimental simulation proceeds linearly without any parallelization. This is the downside of using the standard MATLAB package where coarse-grain parallelization is not available. Future experiments should take advantage of many coarse-grain parallelizable tasks. However this experimental setup was found to be sufficient for a proof of concept of the algorithm.

There are many different parameters associated with this experimental setup, including the size of the grid, the number of ensemble members, the variance of the measurement noise and ensemble perturbations, and the optimization horizon. All of these parameters were chosen to accomplish two main goals: 1) to create an experiment that is consistent with potential real world uses; and 2) to create an experiment that can be performed with my current computational resources.

### 2.2.1 Convective Cahn-Hilliard Equation As A Model Equation

This work uses the two-dimensional Convective Cahn-Hilliard Equation as a model equation to test the effectiveness of the newly formulated EnAO algorithm. The two-dimensional Convective Cahn-Hilliard equation is typically used as a model equation to represent several physical phenomena, including specific types of crystal growth and phase separation. This equation is a useful model because, with a large enough driving force, the two-dimensional Convective Cahn-Hilliard Equation exhibits chaotic convective behavior [15]. This aligns well with problems that involve plumes driven by environmental forces (e.g. radioactive clouds in air or oil in water) where the evolution of the plume is convective rather than diffusive and typically exhibits chaotic behavior.

The specific mathematical form of the two-dimensional Convective Cahn-Hilliard Equation that was used in this work is:

$$h_t = \frac{1}{2}D|\nabla h|^2 - |\nabla^2 h| - |\nabla^4 h| + 3(h_x^2 + \alpha h_y^2)h_{xx} + 3(\alpha h_x^2 + h_y^2)h_{yy} + \beta h_x h_y h_{xy} \quad (2.41)$$

Where the coefficients $\alpha$, $\beta$, and $D$ were chosen to be 0, 0, and 10 to exhibit the chaotic behavior seen in [15].

Appropriate numerical methods were used to simulate the two-dimensional Convective Cahn-Hilliard Equation. The spatial discretization was done pseudospectrally on a periodic domain, meaning all derivatives were done in Fourier space and all nonlinear products were done in physical space. The system was time stepped using a mixed implicit/explicit scheme where Crank-Nicholson was used for the linear terms and Runga-Kutta 3 was used for the non-linear terms.

A contour plot was used to plot the two-dimensional Convective Cahn-Hilliard equation so that it would be easier to see vehicles moving in the same two dimensional grid.

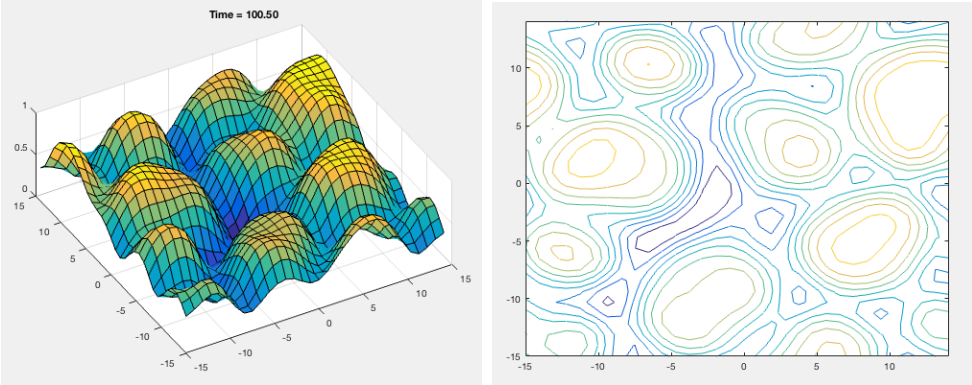


Figure 2.2: Example of the solution to the two-dimensional Convective Cahn-Hilliard Equation. This snapshot is representative of the peaks and valleys that move around the grid as the equation marches in time. Left) Surface plot to better illustrate the peaks and valleys of the equation. Right) Contour plot which will be used in the demonstration of the adaptive observation algorithm.

### 2.2.2 Estimating Truth Simulation

An EnKF is used to create an estimate of the model using the measurement data from the sensor vehicles. The EnKF works as described in section 1.2. At the beginning of each simulation, the number of ensemble members, $N$, is specified and all ensemble members are initialized at zero with no initial variance. The ensemble members are propagated in the same manner as the model equation with an additional disturbance $w_k^n$ applied to each ensemble member. $w_k^n$ is a zero mean Gaussian disturbance with a covariance specified at the beginning of each simulation. At specified measurement times, the ensemble members are updated according to the measurement data with additional measurement noise $v_k^n$. The noise, $v_k^n$, is a zero mean Gaussian white noise with covariance $R$ that is specified at the beginning of the simulation.

The estimate of the truth simulation is then calculated as the average of the ensemble. Because the ensemble is initialized at zero, the estimate takes time to approach the truth model. The speed at which the estimation approaches the truth model and the overall estimation error is dependent on the control technique used to move the vehicles.

### 2.2.3 Controlling Sensor Vehicles

Control of the sensor vehicles is calculated using the EnAO algorithm described in section 2.1. The number of vehicles is specified at the beginning of each simulation and the vehicles are initialized evenly in the middle of the domain. It is assumed that the vehicle states can be accurately tracked so there is no uncertainty in the vehicle states. A finite prediction time horizon is specified at the beginning of each simulation and the EnAO optimizes a control output for the horizon according to the cost function described in section 2.1. At the beginning of each horizon, the EnAO is initialized with the best current estimate of the model equation. As said before, the best current estimate of the model equation is the average of the ensemble taken from the EnKF running in parallel with the truth simulation.

Numerically, the EnAO algorithm is fairly simple to implement. The EnAO starts with a prediction in which the vehicles are propagated forward in time using point mass dynamics with damping, and an ensemble of flow predictions is propagated forward in time using the same technique as in the truth simulation with additional state disturbances for each ensemble member. Discrete updates to the flow are made according to the EnKF at specified update intervals. Once the vehicles and flow are propagated to the end of the prediction horizon, then their adjoints are propagated backwards in time from the end of the horizon to the beginning. The adjoint of the linear operator for the vehicle movement is straightforward; however, the model equation for the flow must be linearized and discretized in order to find its adjoint.

To do this the two-dimensional Convective Cahn-Hilliard Equation (2.41) was first linearized. This equation was then discretized in space using finite difference techniques. Central difference was used for the first derivatives, second order central difference for the second derivatives, and the classic 13-point approximation for the bi-harmonic oscillator equation. All discretizations assumed periodic boundary conditions.

Once the gradient is identified as described in the EnAO algorithm in section 2.1 then the minimization of the cost function is achieved through gradient descent

methods. Our results were produced using the Polak-Ribiere conjugate gradient method.

## 2.3 Results

All simulations were run as outlined in the previous section. To test the algorithm, each simulation was run, not only with the vehicles moving according to the EnAO algorithm, but also with vehicles sweeping across the entire domain and vehicles moving randomly in the domain. All flow estimates were produced in exactly the same way using an EnKF with 20 ensemble members running in parallel with the truth simulation of the flow. The measurement noise and ensemble disturbances remained consistent between the three vehicle movement cases. The simulation was run with two vehicles in a 16x16 grid. This was chosen to be an adequate size for a proof of concept while still able to be carried out on a standard computer. In each movement case, the vehicles were initialized at the same positions and control inputs were normalized so that the magnitude of control is consistent between all movement cases.

As stated before, the goal of this algorithm is to move vehicles in a given domain in order to achieve an accurate estimate and forecast of a convective flow that the vehicles are measuring. This algorithm attempts to achieve the goal by minimizing the estimate error variance through the minimization of a corresponding cost function. To show that the vehicles are attempting to minimize the error variance, figure 2.3 shows the trajectory of two vehicles during one horizon plotted over a color map of the error variance at the beginning of the prediction horizon. The figure shows that, as expected, the optimization of the vehicle paths over a given horizon results in the vehicles moving to areas of high variance.

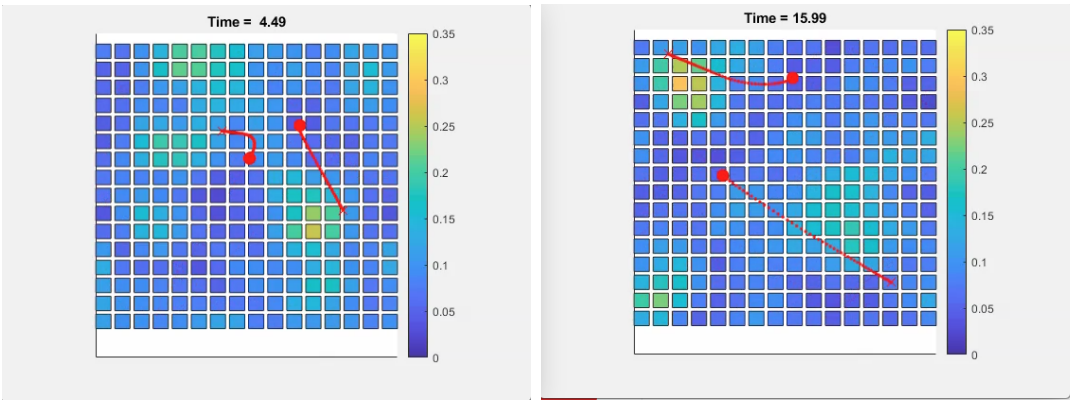


Figure 2.3: Plot of vehicle trajectories over a horizon plotted over the variance at the beginning of the horizon

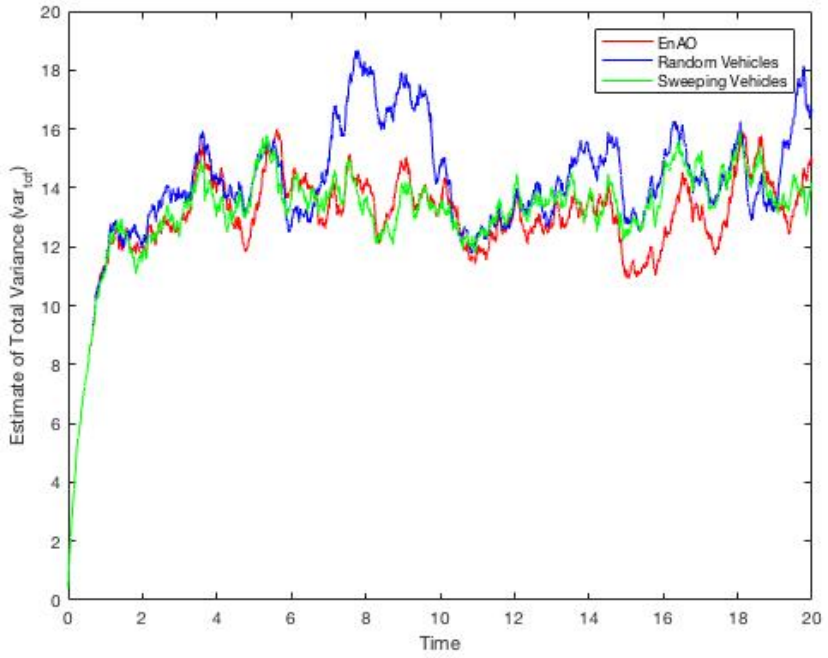


Figure 2.4: Plot of the ensemble variance approximation over time produced from the average of 6 simulations

Table 2.1: The average total variance and standard deviation averaged from time t=2 to t=20 of 7 simulations

| | Average Total Variance | Standard Deviation |
|---|---|---|
| Random Vehicles | 14.5680 | 1.5187 |
| Sweeping Vehicles | 13.5925 | 0.8135 |
| EnAO Vehicles | 13.3104 | 1.0427 |

While figure 2.3 provides evidence that vehicles move toward areas of high variance over a single horizon, the goal is to keep the variance to a minimum for longer periods of time. To this end, figure 2.4 shows a line graph of the total error variance approximation over a specified time. Total error variance is calculated as the sum of the variance at each grid point, (i.e. the trace of the covariance matrix).

$$var_{tot} = trace(P^e) \tag{2.42}$$

These results were produced from an average of 7 trials. You can see that

the EnAO algorithm along with the other vehicle movement strategies keep the total variance from growing in time. However, the line graph does not clearly show that one strategy is better than another. To better interpret the data, it can be seen that the variance remains relatively constant from time $t = 2$ to $t = 20$. The average and standard deviation of this period is shown in table 2.1. The table shows that the EnAO has the lowest variance, but has a higher standard deviation than the sweeping vehicles.

These results are promising because, although sweeping vehicles seem to keep the variance more constant in time, it is not feasible to have sweeping vehicles in a larger domain. This means that sweeping vehicles can be thought of as a best case scenario and the EnAO nearly matches or even slightly outperforms this idealized situation.

In a similar way to figure 2.4, figure 2.5 shows the estimation error over time of the flow estimates from the three different sensor vehicle movement strategies. The estimation error is calculated as the normalized 2-norm of the difference between the truth state and the state estimate.

$$E = \frac{|\hat{\mathbf{x}} - \mathbf{x}|_2}{|\mathbf{x}|_2} \tag{2.43}$$

Figure 2.5 shows that all three cases produce a fairly accurate estimate of the flow after some time. To assess this more visually, image c in figure 2.6 shows what the estimates look like towards the end of the simulation. Based only on looking at figure 2.6, one could say that all estimates are of high quality. But figure 2.5 indicates that the EnAO does in fact produce the best estimate of the flow in the end.

Although results are promising, it is obvious that the limited size of the experiment is holding back more definitive conclusions. For example, a larger domain would make it such that vehicles could not sweep across the entire domain like the ones in the test simulations. To test this, a much larger grid must be used so that the vehicles cannot cover the entire grid in any reasonable period of time. This should favor the EnAO algorithm where the vehicles are trying to find a local area that they

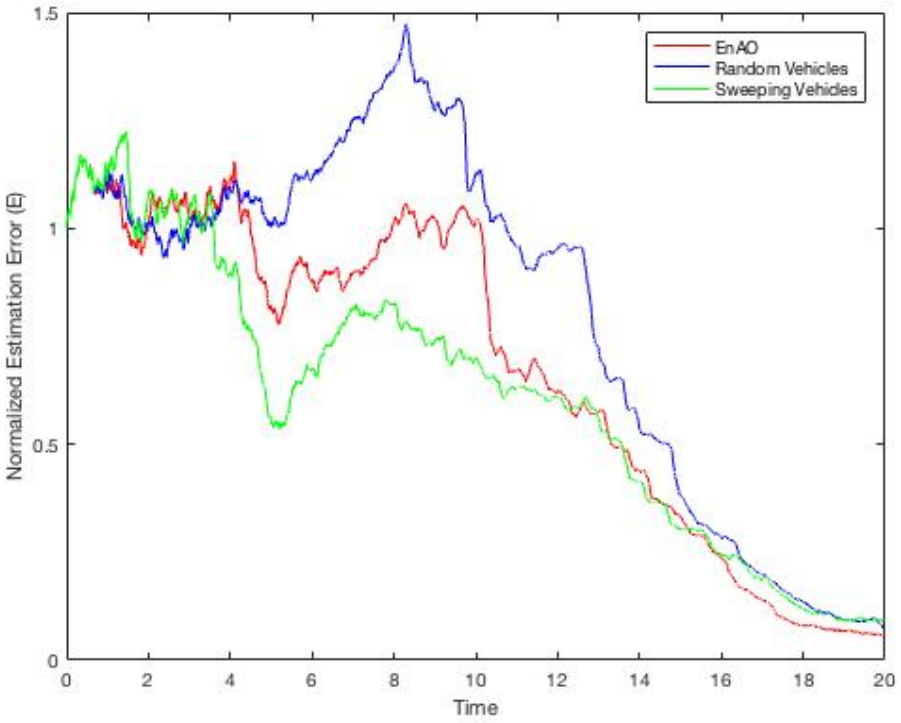


Figure 2.5: Plot of estimation error over time produced from the average of 7 simulations

can move to in order to minimize the variance as opposed to sweeping vehicles that are attempting to cover the entire domain.

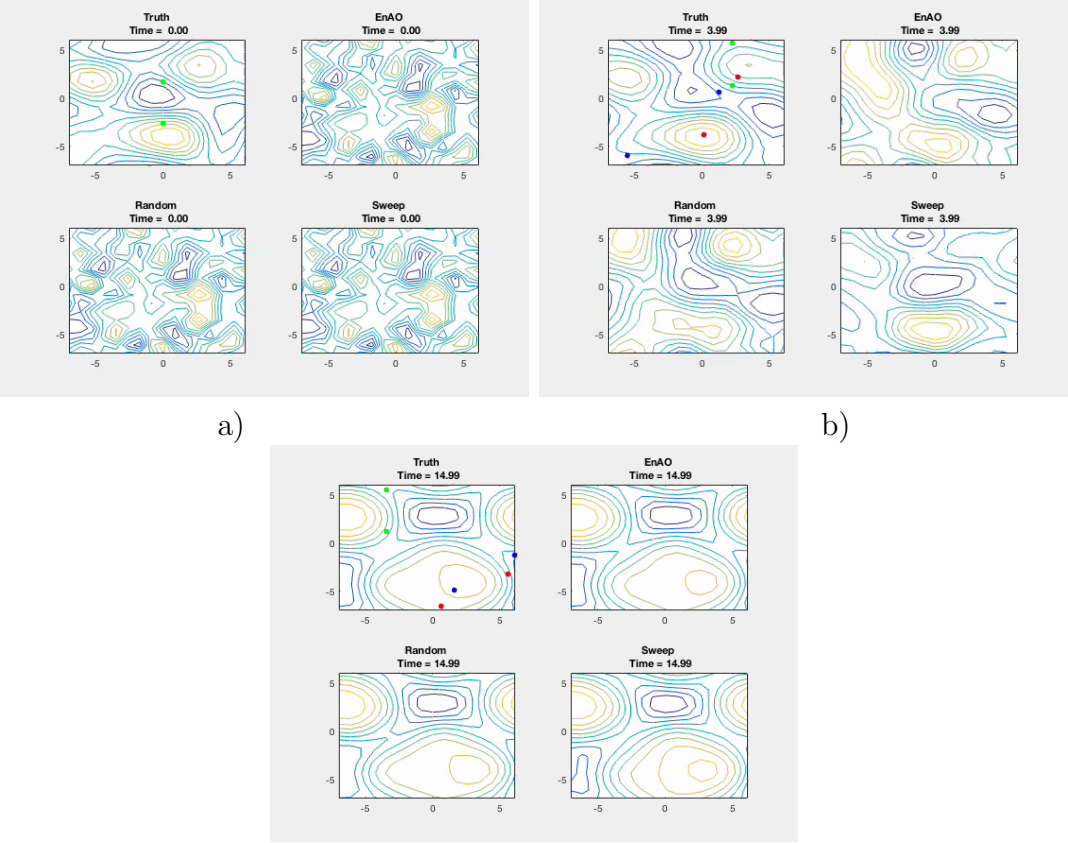


Figure 2.6: Screenshots of the simulation a) Beginning: all vehicles start in the same place and estimates are not good. b) Middle: vehicles move around the domain and estimates start to approach the truth. c) End: estimates from all vehicles look very accurate

### 2.3.1 Covariance Localization

All previous experimental results were produced without covariance localization. To test the effect of covariance localization on my experimental setup, simulations were run where static sensors were used along with an EnKF to estimate the same CCH equation on the same size grid. Using the function from equation 1.7 to compute $\rho$, figure 2.7 shows how reducing c decreases the overall steady state error of the flow estimate. What cannot be seen is that c values lower than 6 resulted in divergence of the covariance matrix and therefore could not estimate the system accurately.

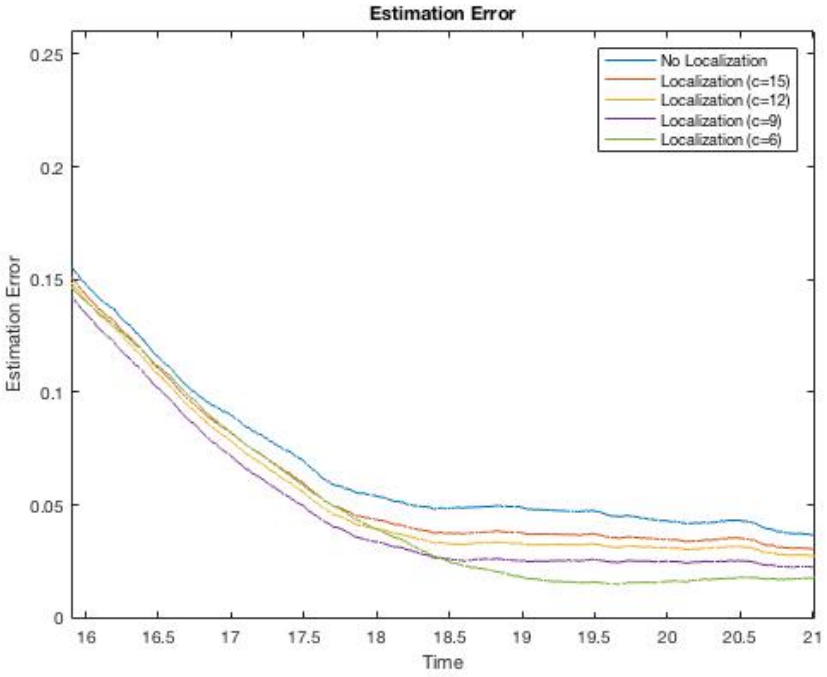


Figure 2.7: Estimation error when estimating flow using EnKF with different amounts of localization

As you can see from figure 2.7, although covariance localization does seem to reduce the error of the flow estimate, it is not by a significant amount. This result is most likely due to the relatively small size of the system state. This shows that the previous testing of the algorithm without covariance localization is still valid.

# Chapter 3

# Conclusion

## 3.1 Summary

This thesis presented an algorithm (EnAO) that can be used to plan the trajectory of sensor vehicles moving in a flow field. While the end goal is to use this algorithm to track environmental plumes, this thesis presented initial findings based on numerical simulations. The numerical simulations allowed sensor vehicles to move in a grid containing a model flow equation. The sensors gathered measurements from the grid to estimate the current state of the flow and the algorithm used the estimate to create a forecast to determine the best vehicle trajectory to acquire new measurements.

Initial results show that the algorithm does in fact have the ability to accurately estimate a flow field, and it shows advantages over strategies that sweep vehicles and randomly control vehicles across the grid. These initial experiments were performed on size-limiting software and hardware. However, the results suggest that the algorithm could be just as effective at a larger scale. To test this, further simulation is required on larger domains with flow model equations more consistent with real world situations.

## 3.2 Future Work

As I have mentioned previously, the success of this work has largely been limited by the capabilities of the hardware and software used in the experimentation. For this reason, I believe future work should be programmed in C or Fortran. From a software perspective, this will allow for more efficient fine-grained parallelization, which is known to be better when using C or Fortran rather than MATLAB. Utilizing C or Fortran will also make it possible for the simulations to be performed on multi-CPU super computers. This makes it possible to code for coarse-grained parallelization. Specifically it means that the calculations for each ensemble member can be split up onto different CPUs, significantly decreasing computation time.

Aside from allowing for larger simulations, more computing power would make it possible to do more complex simulations. The model equation used in this thesis was adequate to test the algorithm on a convective flow, but real life fluid flows are always modeled using the Navier-Stokes equation. Running simulations with the Navier-Stokes equation will bring the algorithm even closer to being run in real time in real-world environments.

Finally, after large-scale simulations using Navier-Stokes equation are performed and found to be successful, it is a logical next step to design a real-world experiment to test the algorithm.

ACKNOWLEDGEMENTS

This thesis, in part, is currently being prepared for submission for publication of the material. Wong, Mark; Bewley, Thomas. The thesis author was the primary investigator and author of this material.

# Bibliography


[1] D. Zhang, C. Colburn, and T. Bewley, "Estimation and adaptive observation of environmental plumes," *Proceedings of the 2011 American Control Conference*, 2011.

[2] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[3] Y. Tian, W. Li, A. Zhang, J. Yu, Q. Zhang, and Y. Li, "From simulation to validation: Moth-inspired chemical plume tracing with an autonomous underwater vehicle," *2014 Oceans - St. Johns*, 2014.

[4] N. Yilmaz, C. Evangelinos, P. Lermusiaux, and N. Patrikalakis, "Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 4, pp. 522–537, 2008.

[5] R. N. Smith, Y. Chao, B. H. Jones, D. A. Caron, P. P. Li, and G. S. Sukhatme, "Trajectory design for autonomous underwater vehicles based on ocean model predictions for feature tracking," *Springer Tracts in Advanced Robotics Field and Service Robotics*, pp. 263–273, 2010.

[6] G. Evensen, "Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics," *Journal of Geophysical Research*, vol. 99, no. C5, pp. 10143–10162, 1994.

[7] G. Gaspari and S. E. Cohn, "Construction of correlation functions in two and three dimensions," *Quarterly Journal of the Royal Meteorological Society*, vol. 125, no. 554, p. 723757, 1999.

[8] K. Bergemann and S. Reich, "A localization technique for ensemble kalman filters," *Quarterly Journal of the Royal Meteorological Society*, 2010.

[9] G. Evensen, "Advanced data assimilation for strongly nonlinear dynamics," *Monthly Weather Review*, vol. 125, no. 6, pp. 1342–1354, 1997.

[10] H. L. Mitchell, P. L. Houtekamer, and G. Pellerin, "Ensemble size, balance, and model-error representation in an ensemble kalman filter*," *Monthly Weather Review*, vol. 130, no. 11, p. 27912808, 2002.

[11] V. E. J. Haugen and G. Evensen, "Assimilation of sla and sst data into an ogcm for the indian ocean," *Ocean Dynamics*, vol. 52, pp. 133–151, Jan 2002.

[12] S. Ha, C. Snyder, W. C. Skamarock, J. Anderson, and N. Collins, "Ensemble kalman filter data assimilation for the model for prediction across scales (mpas)," *Monthly Weather Review*, vol. 145, no. 11, pp. 4673–4692, 2017.

[13] L. Wang, J. Huang, L. Wang, J. Huang, L. Wang, J. Huang, P. Gao, and H. Wu, "Estimating winter wheat yield by assimilation of modis lai into wofost model with ensemble kalman filter," *2017 6th International Conference on Agro-Geoinformatics*, 2017.

[14] N. K. Singh, S. Bhaumik, and S. Bhattacharya, "Tracking of ballistic target on re-entry using ensemble kalman filter," *2012 Annual IEEE India Conference (INDICON)*, 2012.

[15] A. A. Golovin, A. A. Nepomnyashchy, S. H. Davis, and M. A. Zaks, "Convective cahn-hilliard models: From coarsening to roughening," *Physical Review Letters*, vol. 86, no. 8, pp. 1550–1553, 2001.

[16] J. Corts and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 495–503, 2017.

[17] M. Dunbabin and L. Marques, "Robotics for environmental monitoring," *IEEE Robotics & Automation Magazine*, vol. 17, no. 4, pp. 118–118, 2010.

[18] P. L. Houtekamer and F. Zhang, "Review of the ensemble kalman filter for atmospheric data assimilation," *Monthly Weather Review*, vol. 144, no. 12, pp. 4489–4532, 2016.

[19] C. K. Wikle, M. Katzfuss, and J. R. Stroud, "Ensemble kalman filter," *Wiley StatsRef: Statistics Reference Online*, pp. 1–6, 2018.

[20] H.-L. Choi and J. P. How, "Coordinated targeting of mobile sensor networks for ensemble forecast improvement," *IEEE Sensors Journal*, vol. 11, no. 3, pp. 621–633, 2011.