The Dynamics of Point-Vortex Data Assimilation

by

Natalie Ross

B.S., University of Texas at Austin, 1999

M.S., University of Colorado, 2004

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2008

This thesis entitled:
The Dynamics of Point-Vortex Data Assimilation
written by Natalie Ross
has been approved for the Department of Computer Science

_____

Prof. Elizabeth Bradley

_____

Prof. Jean Hertzberg

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Ross, Natalie (Ph.D., Computer Science)

The Dynamics of Point-Vortex Data Assimilation

Thesis directed by Prof. Elizabeth Bradley

The standard grid-based approach to modelling fluid flows—termed direct numerical simulation (DNS)—achieves impressive accuracy for most flows. However, DNS simulations are very compute-intensive, so they are currently impractical for real-time application domains such as flow control. Several *reduced-order* modelling techniques are available that make various approximations to the Navier-Stokes equations governing fluid dynamics. The point-vortex method is one such technique that achieves a reduction in complexity by making simplifying assumptions about the vorticity distribution and representing the entire flow with a collection of point vortices. The resulting dynamics are governed by an ordinary differential equation, which simplifies the physics significantly. These simplifications are not without penalty—point-vortex simulations are typically much less accurate than DNS schemes. However, if the point-vortex model could be corrected with observations of the physical system—a process known as *data assimilation*—the resulting simulation might be sufficiently accurate for modelling and control applications. Care must be taken to ensure that the computational costs of the data assimilation scheme do not destroy the speed advantages of the point-vortex model. In this thesis, we evaluate point-vortex data assimilation to determine the most efficient and effective assimilation strategy. We have proposed several *dynamics-informed* approaches that attempt to use the system dynamics to determine when the model needs a correction. The goal is to avoid the computational cost of correction when the model is performing well. We compare our dynamics-informed techniques to the standard approach in which the model is corrected at periodic intervals. Numerical experiments with several different vortex configurations and assimilation algorithms facilitate this comparison. The dynamics-informed techniques work very well for some vortex configurations, with a significant decrease in computational cost as compared to periodic correction. For other configurations, we identified some patterns in the vortex dynamics that can degrade the performance of dynamics-informed techniques. To ensure that our results apply to real-world flows, we have also performed a thorough analysis of assimilation using data from a laboratory planar air jet.

Dedication

To my mother, Martha Rooney and my daughter, Zoe Ross

Acknowledgements

First and foremost, I would like to thank my entire family for their continual love and support. My mother, Martha Rooney, has always provided motivation and encouragement for all of my academic endeavors. My husband, Michael, has also been incredibly supportive, especially during times when I have doubted myself. My daughter, Zoe, is a continual source of amazement; I hope my accomplishments will give her a sense of the importance of perserverance and commitment in acheiving your goals. Last but not least, I am so appreciative of everything my mother-in-law, Hui Suk, has done for us over the last couple of years.

My advisors throughout this process have been amazing. Liz Bradley is a wonderful person who inspires others. She has spent countless hours helping me with every aspect of my research, and she has also been so caring and supportive on a personal level. I would also like to thank Jean Hertzberg and Jeff Anderson, who were much more involved in my work than typical secondary advisors. Jean has spent many hours helping me to understand fluids concepts, listening to practice talks, and generally being a good friend and supporter. Jeff has taken the time to meet with me *many* times to help with my DART research, and I have appreciated his caring and genuine approach to advising. I am also grateful to Tim Hoar at NCAR, who supported me in using the coral computing cluster, which significantly speeded my progress.

Finally, I would like to thank some of my colleagues at InfoPrint Solutions Company. Betsy Hicks, Steve Gebert, and Jim Crowell provided crucial management support for my PhD work, allowing me to use some company time to work on my research. Nenad Rijavec has always been willing to assume an additional workload to enable me to make progress. I would also like to thank Arianne Hinds for her positive attitude, encouragement, and example.

Financially, my thesis was supported by the Advanced Learning Assistance Program at IBM and then InfoPrint Solutions Company. I also received a Chancellor's Fellowship from the Graduate School at the University of Colorado. I am very grateful for this monetary support.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

Fast and accurate numerical models are critical for the tracking, prediction, and control of fluid flows. Traditional grid-based modelling techniques, though highly accurate, are often too slow for these purposes. So-called *reduced-order models*—which track abstract flow structures, rather than the details of fluid velocities at every point—are much faster, but the inherent coarseness of their modeling approximation makes them inaccurate. However, correcting a reduced-order model with observations of the fluid—a process known as data assimilation—could produce a model that has both the speed and accuracy required for real-time applications. We explore this hypothesis using the point-vortex model, which tracks only the vortices in the flow. Our primary goal in this exploration is to determine the best assimilation strategy for correcting this solver's mistakes with minimal computational effort. To achieve this goal, we compare traditional assimilation strategies to new *dynamics-informed* techniques that use knowledge about the system dynamics to determine when the corrections will be most effective and when they are not required. We have performed several numerical experiments with various data sets and assimilation algorithms to determine when and why one approach works better than another. In particular, we have identified some patterns in the norms of induced velocity gradients that indicate when dynamics-informed techniques might fail. More sophisticated data assimilation schemes such as ensemble Kalman filtering can be used to overcome some of these difficulties, and we have explored this assimi-

lation strategy using the Data Assimilation Research Testbed (DART), comparing our dynamics-informed techniques to periodic and *spread-based* correction—an existing ensemble correction timing strategy that has been shown to perform as well as or better than periodic correction.

These numerical experiments are a useful first step, but evaluation in the context of a real fluid flow is needed to ensure that these techniques are practical for real-world applications. To this end, we have tested our assimilation strategies with experimental data from a planar air jet. This is a break from most traditional data assimilation research, in which numerical experiments are the norm. To enable this research, we implemented some techniques for extracting vortex data from velocity field measurements of the jet. In Section 3.1, we present an analysis of vortex extraction techniques and explore different ways of assigning point-vortices to the vortices identified in the flow. Initial results using these observations of the planar jet show some promise, but it is clear that we need to improve our vortex extraction techniques and extend our model to include boundary conditions for this flow. In the remainder of this section, we introduce the concepts that are central to this thesis—numerical modelling of fluid flows and data assimilation—and provide an overview of our dynamics-informed strategies and laboratory testbed.

The current state-of-the-art in computational modelling of fluid flows is direct numerical simulation (DNS). DNS models discretize the flow physics by covering the domain with a fine grid and tracking the quantities of interest at each grid point. To evolve the system, DNS models perform a direct integration of the Navier-Stokes equations. This approach works remarkably well, especially at high resolution, and DNS models are used successfully in a variety of important applications, including numerical weather prediction and fundamental studies of combustion, vascular flows, vehicle aerodynamics, and ink jet dynamics, but these models require enormous computational effort and cannot be used in real time.

The development of faster and more-abstract modelling techniques has been a significant research focus in the fluid dynamics community for many years. We review this literature in Section 2.1. One suprisingly successful technique is the point-vortex method, which tracks only the vortices in a flow and ignores all other dynamics. In this method, the vorticity field is idealized as being concentrated into a collection of delta functions, with each point called a point-vortex. The dynamics of these point vortices are governed by an ordinary differential equation, so the resulting computational model is very fast. However, the abstractions and approximations necessary to achieve this simplification do come at a cost: a point-vortex model is not nearly as accurate as a DNS model. As a result, point-vortex modelling is typically not practical in tracking and control applications. If we could correct the solver's mistakes, however, we could capitalize on its speed advantages in these difficult application domains.

One way to improve the accuracy of the point-vortex model is to correct it with measurements of the target system—a process known as data assimilation. Data assimilation is not a new field; it was invented by meteorologists in the 1950s for correcting numerical weather prediction models with atmospheric observations. In these and other geophysical applications, researchers use complex, parameterized DNS models to produce a state estimate. This estimate is then periodically corrected with available observations of the system. These observations are typically sparsely distributed over the domain and are obtained via a wide array of different types of observing equipment with varying noise characteristics. The majority of data assimilation research, which is reviewed in Section 2.2, focuses on the development of algorithms that statistically weight the assimilated observations based on their relative accuracy compared to the model state estimate.

A primary goal of this thesis is to analyze and evaluate assimilation strategies to determine the most efficient method for achieving accurate point-vortex simulations without destroying the speed advantages of using a reduced-order model. If we could

use data assimilation to produce point-vortex simulations that are competitive with DNS models in accuracy but significantly faster, our simulations could enable real-time fluid modelling and control applications. The costs associated with any assimilation algorithm can be divided into two categories: the complexity of the algorithm itself and the costs of gathering and processing the observations for each assimilation time. This thesis focuses on reducing the time required for the latter. In most traditional data assimilation applications, the observations are assimilated at periodic intervals. If we could instead detect when the model is likely to need a correction and correct *only* at these critical moments, we might be able to achieve greater simulation accuracy at a lower cost. Our approach, which shares some similarities with the targeted observing literature reviewed in Section 2.2, is to use the system dynamics to dictate when correction is necessary. In Section 5, we describe several techniques for doing so, including tracking changes in induced velocity gradients, thresholding for large eigenvalues or large shear, and examining differences between test steps in the Runge-Kutta integration. We have applied these techniques in numerical experiments with several different point-vortex configurations, comparing them to each other and to periodic correction. The results of these explorations are described in detail in Sections 7, 9, and 12.

In our initial numerical experiments, we used the simplest possible approach for correcting the point-vortex model—at each assimilation time, we simply replaced the model state variables with the observed values. One problem with this approach is that it does not consider the relative accuracy of the model's state estimate compared to the observations. The observations are taken as the "truth" no matter how inaccurate they are. In Section 8, as we increased the noise in the observations, we found that it was difficult to maintain the accuracy of the simulations. We identified some patterns in the vortex dynamics that can make it difficult for dynamics-informed techniques to work well.

These difficulties led us to explore more-sophisticated data assimilation schemes

that consider the observational error in the correction algorithm. The first of these—Newtonian nudging—is a simple technique for weighting the observations based on their noise characteristics. The model state vector is "nudged" a fraction of the distance toward the observations at each assimilation time. We confirmed in Section 10 that a larger nudge is more appropriate for more accurate observations. The second technique we considered was ensemble Kalman filtering, a significantly more-complex technique that is very popular in the meteorological and geophysics research communities. We used the Data Assimilation Research Testbed (DART) framework from the National Center for Atmospheric Research to facilitate these experiments. Although ensemble filtering may be too computationally intensive for real-time purposes, it is still worthwhile to analyze the results that can be achieved with a state-of-the-art assimilation algorithm. Also, we are the first to explore ensemble filtering using the point-vortex model, and we have confirmed that it works well for the data sets we studied. This is an important contribution, as many atmospheric and oceanic applications are well-suited for point-vortex modelling. The results of our DART analyses are presented in Section 11.

Although numerical experiments are a useful first step, their results rarely generalize to real-world applications. In our research, we have also analyzed point-vortex data assimilation techniques using real experimental data. This is in contrast to most data assimilation studies, which rely on "Observing System Simulation Experiments"—in which simulated observations are generated by the computational models under study—to evaluate new correction techniques. Real-world experiments are rare, because it is usually very difficult to collect and process experimental data from complex systems. We have chosen a much more controlled enviroment in which to study data assimilation: a planar air jet in our laboratory. Observations of this system are collected by particle image velocimetry, which provides velocity field data over the entire domain. Given this velocity field, we can extract vortex positions and strengths to use for the assimilation. In Section 3.1, we describe and compare some existing vortex extraction techniques and

develop two different approaches for assigning point-vortices to the vorticity field. The resulting point-vortex observations are then used in numerical experiments and also in a "real-world" assimilation. Although the initial results presented in Section 12 and 13 are promising, it is clear that our model will need to be enhanced to include boundary conditions for vortex entry and dissipation. Refining the model and the assimilation of real data will be an important component of our future work.

Chapter 2

Background and Related Work

2.1 Numerical Modelling

Perhaps the canonical example of a complex system that engineers and scientists need to model is the fluid flow. Real-world fluids problems simply do not admit analytical solutions, so one has to model them numerically, and their inherent spatiotemporal complexity makes this very hard. One traditional solution to this, termed direct numerical simulation or DNS, involves discretizing the flow quantities using finite-order approximations of time and space [102]. To get the flow details right in face of this discretization, the grids and timesteps involved may need to be very fine. A fine grid translates to an extremely large state vector and/or a multiscale coupled-models approach in the simulation of complicated flows. The algorithmic methods used in many codes to solve problems like this—e.g., successive over-relaxation—can have sensitive numerical dynamics that do not always converge. For all of these reasons, DNS simulations of even fairly simple fluids problems require hours of CPU time on powerful machines with large memories.

If a coarser representation could be used to model the dynamics of the system, the resulting numerical solver would be simpler, and hence much faster, than DNS models. There are many ways to reduce the number of state variables that a model uses to capture the dynamics. Current approaches use knowledge about the flow to decompose it in various meaningful ways: e.g., into its spectral modes [23] or using spatial basis

functions like Proper Orthogonal Decomposition (POD) [15] or wavelets [46, 47]. One can also use various kinds of averaging techniques on the fundamental fluids equations to select the spatial scale of the model; the current frontrunners in this class of approaches are the LANS-α [25, 61, 100] and RANS models [132][1] . Large-eddy simulation (LES) methods use spatial filtering to select the modeling scale [55, 88, 101].

Proper Orthogonal Decomposition (POD), also known as Karhunen-Loeve decomposition [81, 90], is the most popular reduced-order modelling approach in the flow control community [12, 122, 54, 123, 129]. The basic idea is to project the Navier-Stokes equations onto a reduced subspace of order m, where the subspace is chosen such that the error in the projection is minimized. In this subspace, the state of the flow at time t, $x_m(t)$, is a linear combination of basis vectors, or "POD modes", ϕ_i:

$$x_m(t) = \sum_{i=1}^{m} a_i(t)\phi_i \quad .$$

Galerkin projection of the Navier-Stokes equations onto the subspace defined by the set $\{\phi_i\}$ produces the ordinary differential equations that describe the time evolution of the coefficients, $a_i(t)$. Data from numerical simulations (or, less commonly, experiments) is used to obtain—via the method of "snapshots" [130]—the appropriate POD basis for the flow being modelled. Note that the POD basis is flow dependent, and the snapshots used to obtain it must contain a representative sample of all flow dynamics, including the effects of any control action. If this is not the case, the resulting POD basis will prove insufficient for model-based control. Several researchers have investigated ways to ensure that the POD modes are capable of representing the large-scale coherent structures in the flow [57, 76, 95, 105, 106].

Given a set of representative POD modes and an ODE describing their dynamics, one can use the resulting model in a closed-loop control setting. Experimental measurements of fluid properties—typically, velocity or surface pressure—are used to estimate

[1] Lagrangian- and Reynolds-averaged Navier-Stokes, respectively; the α represents the scale of the method.

the current full state of the flow in terms of the POD basis functions. This is typically accomplished via linear or quadratic stochastic estimation [3], but Kalman filters have also been used for this task [121]. This state estimate is used as an initial condition for the ODE that governs the evolution of the coefficients $a_i(t)$. Solving the ODE provides a prediction of the coefficients at some later time, which can be compared with the desired values to determine the appropriate control action. This is essentially a simple form of data assimilation, in which the model state variables—the coefficients $a_i(t)$—are updated periodically based on the experimental sensor data.

One drawback to using POD models in data assimilation research is that they are very sensitive to the empirical data used to obtain the modes, and the resulting model can "overfit" the input data [35]. The POD model is also flow-dependent, which makes it difficult to generalize the results from one flow to another. Because of these limitations, we have chosen to use a different model—the point-vortex model—in our data assimilation studies in the hopes that the results can be applied to a wider variety of fluid flows.

The point-vortex model [127] is an approximation to the Navier-Stokes equations that is flow-independent and does not require any data to determine its structure. The approximation is based on the coherent vortices in the flow, structures whose dynamics are easily modelled and whose dynamics are a good basis for describing the flow physics. The model is inherently grid-free, and the state variables are meaningful flow quantities that are helpful in understanding its dynamics. And, unlike DNS solvers, which depend critically on accurate specification of detailed initial and boundary conditions across the simulation region, the point-vortex model can be easily warm-started, with less information and no initial transient artifacts.

The details of the method are straightforward. Vorticity is a field vector quantity defined as the curl of the velocity; it represents the angular momentum of the fluid. A vortex is a local peak or concentration in the vorticity; circulation, Γ, is the integral

of vorticity around a curve enclosing an area of fluid. In the point-vortex model, all vorticity is idealized as being contained at specific points, which move with the flow field. State variables in the point-vortex model are the positions (x, y) and strengths Γ of these point vortices. The dynamics are the fluid-mechanical analog of point masses evolving under the mutual interaction of Newtonian gravity: a vortex is treated as creating a swirling velocity field around itself, and other entities—vortices, passive tracer particles—move or "advect" with that velocity. The magnitude of the induced velocity falls off as $1/r^2$ with the distance r from the corresponding vortex core. The point-vortex equations use superposition to combine the effects of multiple point vortices. In schematic form, the equations for the evolution of the state of the ith point vortex are:

$$\begin{bmatrix} \dot{\vec{X}}_i \\ \dot{\Gamma}_i \end{bmatrix} = \begin{bmatrix} \vec{f}(\Gamma_j, \vec{X}_j,\ j \neq i) \\ 0 \end{bmatrix}, \tag{2.1}$$

where $\vec{X}_i = (x_i, y_i)^T$, the 2D position of the ith vortex, and $\vec{f}$ is a vector-valued function whose ith component computes the distances $||\vec{X}_i - \vec{X}_j||_2$ from the ith vortex to each of the j others, calculates their influence at that distance (via the $1/r^2$ law, scaled by the strength Γ_j), rotates to the tangential direction, and does a vector sum of the results. These equations are a form of the well-known Biot-Savart equations describing point-vortex motions [84]. One can solve the system (2.1) with any ordinary differential equation (ODE) solver, such as Runge-Kutta.

The point-vortex model is highly idealized; it is a 2D model that assumes that the flow is inviscid and that the vorticity field is discretized into isolated points. These simplifying assumptions are what makes it fast, but idealization also introduces inaccuracy. Real vortices are not concentrated at a single point, and only higher Reynolds number flows can be treated as inviscid. More typically, vorticity is distributed throughout the flow, and it is created and destroyed as the flow evolves. Nonetheless, the point-vortex method works remarkably well [18] if the flow is dominated by isolated regions of high vorticity, the fluid surrounding those regions is basically irrotational, and viscosity is

small—assumptions that are valid for many interesting flows. There are many ways to extend the point-vortex model to handle cases where these assumptions are not valid [4, 11, 24, 29, 50, 67, 119]. It is important to note that vortex dynamics in 3D are very different than in 2D, so Equation 2.1 cannot simply be extended to handle 3D effects[2] . The target application for this project is a planar jet, so the 2D assumption is reasonably valid here. The other improvements mentioned in this paragraph do not play roles in our research, since the solver itself is not our focus. The goal of this thesis is to figure out whether data assimilation can improve the accuracy of the original point-vortex algorithm and present a proof-of-concept example that it works.

2.2 Data Assimilation

In order to combat the small- and large-scale errors introduced by any numerical modelling approximation, one can correct the solver with experimental measurements of the fluid under investigation—a process known as data assimilation (see [33, 136] for an overview). Most data-assimilation research occurs in the atmospheric and oceanic communities; an important atmospheric application with very widespread impact is numerical weather prediction. All of the major operational weather forecasting centers use data assimilation to improve the accuracy of their forecasts [30, 93, 112, 117]. Figure 2.1 shows a schematic of the correction process, and depicts what is typically referred to as a *data assimilation cycle*. The steps in the process are as follows. In the first cycle, one must specify the initial conditions for the model integration. One approach is to initialize the model using prior observations of the system. For example, in atmospheric applications, one can use *climatology*—the average weather observed in the model domain over some historical period [14]. Once the initial conditions have been specified, the model is run for a specified time interval (cycle length) to produce a *forecast* or

[2] DNS schemes, in contrast, can easily be extended to 3D, or to handle viscosity; the local velocity interactions that they model are invariant under rotation, and viscosity just adds a term to the equation for each grid point.

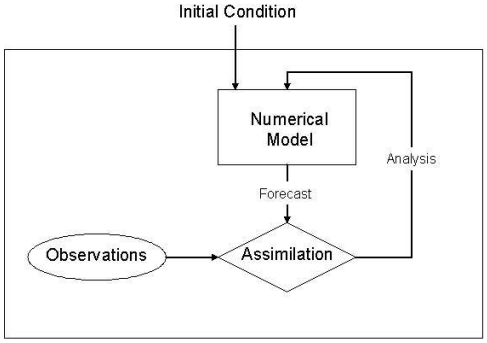


Figure 2.1: Data assimilation cycle. A numerical model is used to generate a forecast or "background state" from a best-guess initial condition. Data assimilation is then used to combine the background state with the available observations, each weighted according to its expected accuracy. The result is an "analysis state", which is used as the initial condition for the next assimilation cycle.

background state. This forecast step is represented by

$$\mathbf{x}^{\mathrm{f}}(t_{i+1}) = M_i[\mathbf{x}^{\mathrm{f}}(t_i)]\,,$$

where M_i is the model dynamics operator and $\mathbf{x}^{\mathrm{f}}(t_i)$ is its state vector at time t_i. Finally, observations of the dynamical system are combined with this background state to produce a new model state known as the *analysis*. Depending on the analysis algorithm and the model, it may be necessary to apply *initialization* techniques to the analysis to ensure that it satisfies dynamic balance conditions. There are many approaches to the initialization problem, including dynamic initialization [99, 104, 137], normal mode initialization [40, 96, 141], and variational techniques [42, 124]. The analyzed/initialized state is then used as the initial condition to start the model forecast for the next data assimilation cycle.

This seemingly simple data assimilation cycle is rich with interesting and challenging problems. Some of the research in this field is devoted to the analysis step, i.e., determining what algorithm should be used to update the model variables, based on the available observations. One naive approach to this is to simply throw out the simulated variable values and replace them with the measured ones (where they exist). We will refer to this method as "direct replacement." One can also use a proportional control strategy, known as "Newtonian nudging" in the meteorological community [34], to perform the correction. This technique corrects the model state by shifting it *toward* the observations, with the size of the shift based on the relative accuracy of the background forecast compared to the observations. This is a simple way to weight the observations based on their noise characteristics. However, neither Newtonian nudging nor direct replacement attempts to explicitly model the effects of observational and modelling errors.

Observational and modelling errors confound the state estimation problem, and developing realistic representations of these errors is a major challenge for data assimi-

lation researchers [20, 36, 37]. Observations of a dynamical system can be represented by the equation

$$\mathbf{y}_i^{\mathrm{o}} = H_i[\mathbf{x}^{\mathrm{f}}(t_i)] + \epsilon_i^{\mathrm{o}}\,,$$

where $\mathbf{y}_i^{\mathrm{o}}$ is a vector of observations (indicated by the superscript o) at time t_i, $\mathbf{x}^{\mathrm{f}}$ is the model forecast state, H is an *observation operator* that maps from the model state space to the observation space, and ϵ_i^{o} is a noise term. Note that ϵ_i^{o} contains both instrument error and *errors of representativeness*, which result from the inability of any discrete model to resolve small-scale processes in continuous dynamics. Errors of representativeness are much more difficult to quantify than instrument error [115]. Also, H may be a time-dependent nonlinear operator, which can introduce additional computational complexity into the assimilation process.

In addition to these observational difficulties, data-assimilation schemes must also consider the model error ϵ^M in the following evolution equation:

$$\mathbf{x}^{\mathrm{t}}(t_{i+1}) \quad = \quad M[\mathbf{x}^{\mathrm{t}}(t_i)] + \epsilon_i^M \quad . \tag{2.2}$$

Here, $\mathbf{x}^{\mathrm{t}}(t_i)$ is the discretized true state (indicated by the superscript t) of the dynamical system at time t_i. Recall that M is the model dynamics operator, which is generally nonlinear. The term ϵ_i^M quantifies both the errors that result from the discretization inherent in a computational model as well as more-serious errors caused by incorrect parameters or dynamics in the model. Many data assimilation algorithms make a *perfect-model* assumption and ignore the model error.

More-complex data assimilation techniques that attempt to account for these various sources of error can be derived based on the conditional probability of the model state given the set of observations to be assimilated [28, 74]. Ideally, we would like to solve for the true state of the sytem at a particular time t_i, denoted by $\mathbf{x}^{\mathrm{t}}(t_i)$. Let $\mathbf{O}(t_k) = \{\mathbf{o}(t_1), \mathbf{o}(t_2), \cdots, \mathbf{o}(t_k)\}$ denote the set of observations available up to time t_k. The probability density function $p(\mathbf{x}^{\mathrm{t}}(t_i)|\mathbf{O}(t_k))$ provides the complete solution to the

data assimilation problem at time t_i. Unfortunately, as our only knowledge of the true system state comes from imperfect (and typically, sparse) observations, it is impossible to obtain an exact solution for the full probability. Thus, data assimilation techniques approximate various statistical properties of the distribution—for example, mean, mode, and variance characteristics. There are different classes of assimilation methods that depend on which set of observations are chosen. *Filtering* methods attempt to estimate $p(\mathbf{x}^{\mathrm{t}}(t_i)|\mathbf{O}(t_i))$—i.e, they estimate the true state using all available observations up to the time of the analysis. The most commonly used technique in this class is the Kalman filter [78], which provides an estimate of the conditional mean and covariance of this probability density. *Smoothing* methods, in contrast, attempt to capture the current state that best describes a set of observations up to some future time, i.e., $p(\mathbf{x}^{\mathrm{t}}(t_i)|\mathbf{O}(t_{i+L}))$. Four-dimensional variational assimilation [42], which estimates the conditional mode for this probability density, is a popular approach that falls into this class.

Kalman filtering has been widely used for state estimation applications since its introduction in 1960 [78, 79]. As mentioned in the previous paragraph, the Kalman filter provides an estimate of the mean and covariance of the probability density $p(\mathbf{x}^{\mathrm{t}}(t_i)|\mathbf{O}(t_i))$. Estimates of this mean state and its covariance matrix are computed in each analysis step. On the next data assimilation cycle, they are propagated forward in the forecast step to the next analysis time. Following the notation in [70], the Kalman equations that govern the forecast step are

$$\mathbf{x}^{\mathrm{f}}(t_i) = M_{i-1}[\mathbf{x}^{\mathrm{a}}(t_{i-1})] \tag{2.3}$$

$$\mathbf{P}^{\mathrm{f}}(t_i) = M_{i-1}\mathbf{P}^{\mathrm{a}}(t_{i-1})M_{i-1}^{\mathrm{T}} + \mathbf{Q}(t_{i-1}) \tag{2.4}$$

Here, M_i describes the model dynamics operator at time t_i. Equation (2.3) starts the data assimilation cycle by running the model forward in time, starting with the analysis state from the previous data assimilation cycle $\mathbf{x}^{\mathrm{a}}(t_{i-1})$ as the initial condition. The

result is the forecast state $\mathbf{x}^{\mathrm{f}}(t_i)$. In the equations above, $\mathbf{P}^{\mathrm{a}}(t_{i-1})$ is the estimated covariance matrix from the previous analysis step. Equation (2.4) forecasts this covariance matrix forward to the next analysis time. In the derivation of this equation, an assumption has been made that the model error denoted ϵ^M in Equation (2.2) is a Gaussian process with zero mean and covariance matrix Q; this is the model error term that appears above. Once the forecasts $\mathbf{x}^{\mathrm{f}}(t_i)$ and $\mathbf{P}^{\mathrm{f}}(t_{i-1})$ have been obtained, the following equations are used to perform the analysis step using the available observations:

$$\mathbf{x}^{\mathrm{a}}(t_i) = \mathbf{x}^{\mathrm{f}}(t_i) + \mathbf{K}_i\mathbf{d}_i \tag{2.5}$$

$$\mathbf{P}^{\mathrm{a}}(t_i) = (\mathbf{I} - \mathbf{K}_i H_i)\mathbf{P}^{\mathrm{f}}(t_i) \tag{2.6}$$

where $\mathbf{d}_i$ is known as the *innovation vector* and $\mathbf{K}_i$ is the *Kalman gain*:

$$\mathbf{d}_i = \mathbf{y}_i^{\mathrm{o}} - H_i[\mathbf{x}^{\mathrm{f}}(t_i)] \tag{2.7}$$

$$\mathbf{K}_i = \mathbf{P}^{\mathrm{f}}(t_i)H_i^{\mathrm{T}}[H_i\mathbf{P}^{\mathrm{f}}(t_i)H_i^{\mathrm{T}} + \mathbf{R}_i]^{-1} \tag{2.8}$$

Recall that H_i is the observation operator that maps from the model state space to the observation space. In this derivation, observational errors have been assumed to be Gaussian with zero mean and covariance matrix $\mathbf{R}$.

The derivation [28, 33, 74, 134] and application of the Kalman filter equations include several other simplifying assumptions. This method provides an optimal, minimum variance state estimate only if the dynamics, M, and observation operator, H, are linear. It must also be the case that the model and analysis error statistics are Gaussian. The well-known extended Kalman filter (EKF) [53, 56] attempts to overcome the linearity assumption. In the EKF, M and H are linearized about the instantaneous trajectory $\mathbf{x}^{\mathrm{a,f}}(t_i)$. Unfortunately, the resulting update equations for the error covariance matrix suffer from a closure problem: they involve a recursive calculation that requires knowledge of all higher-order statistical moments. Typically, these higher-order moments are neglected, and as a result, the resulting EKF can fail to appropriately predict the error

characteristics [44, 52, 97]. More-sophisticated closure schemes involving higher-order moments have better results [49, 85, 86, 97], but the computational cost of storing these moments is prohibitive. A newer technique, known as the unscented Kalman filter (UKF) [77], has been shown to produce better results in highly nonlinear systems and can handle non-Gaussian error statistics.

The Ensemble Kalman Filter (EnKF) [45] was introduced in 1994 to overcome the limitations of the EKF. This Monte-Carlo approach is based on the theory of stochastic dynamic prediction [43]. It begins with a single initial condition obtained from climatology or some other best guess; this state is termed the *central forecast*. An ensemble of initial states is then created with mean equal to the central forecast and variance based on knowledge of the uncertainty in the initial condition. For grid-based models, this amounts to generation of pseudo-random fields with a specific mean and variance; the details of this process are given in [45]. The data assimilation cycle depicted in Figure 2.1 is then applied to each of the N ensemble members, with the Kalman filter (or some variant) used in the forecast and analysis steps. In 1998, [21] and [63] found that it was also necessary to use "perturbed" observations to model the uncertainty due to observational error. These perturbations are required in order for the Monte Carlo algorithm to converge to the Kalman Filter solution for linear/Gaussian problems as the ensemble size gets large. Others have also used parameter perturbations to sample the model error characteristics [62].

The crux of the EnKF method is the assumption that the sample of model states will provide meaningful statistics about the conditional probability density of the true state given the available observations, $p(\mathbf{x}^{\mathrm{t}}(t_i)|\mathbf{O}(t_i))$. In sampling such a probability density, a larger number of samples will generally provide better statistical estimates. However, for most atmospheric and oceanic models, the high computational cost of the analysis for a single model trajectory places a limit on practical ensemble sizes. One result of a small sample size can be an underestimate of the computed sample

covariance, which causes the filter to apply too little weight to the observations in the analysis. When this occurs, the analysis will diverge from the true state—a situation known as *filter divergence*. The core issue here is that with an ensemble size of N, the sample covariance is nondegenerate in only $N-1$ directions. If the covariance structure of the joint probability distribution cannot be adequately represented in this subspace, information is lost on each assimilation cycle [7, 87, 97]. A common solution to this problem is *covariance inflation* [7, 59, 140], in which the sample covariance is simply multiplied by a scalar value larger than one on each assimilation cycle. The appropriate inflation factor is typically chosen using numerical experiments: a search is conducted to find the value that produces a minimum ensemble mean rms error [7]. As an alternative, one can use an adaptive method to vary the inflation factor as needed throughout the course of a simulation [6].

Another likely result of the rank problem described above is spurious correlations between model variables and observations. As a result, a distant observation with a spurious correlation may overwhelm the impact of the closer observations that are truly correlated with the state variable. To overcome this problem, a cutoff radius is usually applied to localize the impact of an observation; state variables outside this cutoff will not be updated by the observation [59, 63, 64, 140]. Applying such a cutoff also improves the likelihood that the linearity assumptions are valid for the local region.

Some additional computational simplifications to the data assimilation cycle have been introduced in the context of the EnKF. Houtekamer and Mitchell [64] and Bishop [17] have shown that observations with uncorrelated error can be assimilated one at a time, which reduces the innovation vector in Equation (2.7) to a scalar value. Anderson [8] provides an algorithm that also permits a serial update of the state variables in the model. Using these modifications, each update step in the EnKF is simplified: complex filtering computations can be performed with scalar values instead of matrices. Ensemble techniques can also be easily parallelized to further reduce the required

computational expense [82].

The Data Assimilation Initiative [1] at NCAR has created a software package called the Data Assimilation Research Testbed (DART) that employs some of these techniques. Each analysis update in DART is an update of a single state variable using one observation. The forward operator H is applied to each ensemble member to map from the model state space to the observation space; the result is a distribution of observation values. The next step is to determine how this ensemble sample of observation values should be incremented to more closely match the actual observation and its error distribution. The result is a set of *observation increments*. DART then applies linear regression to obtain the relationship between these observation increments and the state variable that is being updated. There are various algorithms available in DART for determining how to compute the observation increments, including the perturbed observation methods mentioned previously [21, 63] and the Ensemble Adjustment Kalman Filter [7]. DART also contains tunable parameters for addressing the filter divergence problems discussed above.

While ensemble techniques have become very popular in recent years, in part due to their ease of implementation, they cannot yet compete with the current state of the art in weather forecasting—four-dimensional variational assimilation (4DVAR) [42]. This method provides an estimate of the conditional mode of the probability density function, $p(\mathbf{x}^{\mathrm{t}}(t_i)|\mathbf{O}(t_{i+L}))$; this is a maximum likelihood estimate. The basic idea of 4DVAR is to find the model trajectory that most closely matches the observations. Since the model is deterministic, specification of an initial condition for the model will define such a trajectory. Solving for this initial condition amounts to minimization of a cost function which can be derived from an idealized Bayesian probability analysis [92].

$$J[\mathbf{x}(t_0)] = \frac{1}{2}[\mathbf{x}(t_0) - \mathbf{x}^{\mathrm{b}}(t_0)]^{\mathrm{T}}\mathbf{B}_0^{-1}[\mathbf{x}(t_0) - \mathbf{x}^{\mathrm{b}}(t_0)] + \frac{1}{2}\sum_{i=0} n(\mathbf{y}_i - \mathbf{y}_i^{\mathrm{o}})^{\mathrm{T}}\mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^{\mathrm{o}})\,,$$

where $\mathbf{y}_i = H_i[\mathbf{x}(t_i)]$. The first term represents the deviation of the state estimate

from its forecasted value from the previous assimilation cycle. Here, $\mathbf{B}$ approximates the error covariance matrix of $\mathbf{x}^{\mathrm{b}}$; that is, it weights the first term according to the expected accuracy of the background forecast estimate. The second term measures deviations from the observations. Here again, H is the forward operator that maps from the model state space to the observation space and R is the error covariance matrix for the observations. Other strong or weak constraints [124] can be specified to enforce various "balance" conditions that must be satisfied (for example, geostrophic balance in atmospheric models) or to eliminate other undesirable characteristics in the solution. Under certain conditions, the cost function can be minimized using Gauss-Newton iteration [60] or quasi-Newton methods [38]. Adjoint methods are more commonly used for geophysical problems [31, 39, 42, 89, 135]. The adjoint technique solves for the gradient of J by integrating the model forward in time, then integrating the adjoint of the model backward in time. The difficulty in implementing this is that, in order to perform this backward integration, one must have available the adjoints of the linearized model dynamics and observation operators M and H. The mathematics in the derivation of the adjoint can be complex; an example derivation for a simple model based on the shallow water equations is provided in [41]. However, once these adjoint operators have been obtained, the assimilation algorithm is relatively efficient compared to Kalman filtering and, especially, extended Kalman filtering. But, 4DVAR is not necessarily efficient compared to ensemble Kalman filtering techniques.

In addition to the analysis algorithm, another major challenge in data assimilation research is determining how to obtain and leverage the observations in a way that will most improve the accuracy of the model forecast. In typical geophysical applications, there are many different types of observations available (e.g., satellite, radiosonde, radar, and surface measurements). Researchers must start by determining which observations should be discarded due to, for example, instrument malfunctions. This seemingly simple process, known as *quality control*, is a difficult research issue [9, 51, 73, 91]. Then,

for each type of observation to be assimilated, one must develop a separate observation operator H that maps the model state to the observation space. H is often complex and nonlinear, but it can occasionally be simplified by preprocessing the observations to more-closely match the state variables in the model. One must strike a balance in this trade-off between the amount of observation pre-processing and the complexity of H. Each type of observation also has different error characteristics, resulting in a different error covariance matrix R in the Kalman gain equation and the 4DVAR cost function. All of these challenges need to be addressed once the observations have been gathered and before they can be useful in assimilation.

One can also use more intelligent strategies in the observation gathering process; i.e., make an advance determination of what observations will be most useful. This field of study is known as *targeted* or *adaptive observing*; see [17] for an overview. Adaptive observing attempts to determine an optimal way of distributing observations, in both time and space, so as to improve the forecast in some "verification" region. For example, if a convective system were currently moving toward San Francisco, meteorologists might be interested in improving the forecast over that region at a time two days from now. Targeted observing could be used to determine which flight paths sensor-equipped airplanes should take to obtain the most-influential measurements. Most targeted observing strategies attempt to find regions that are either "dynamically connected" to the verification region or that provide the greatest reduction in the error covariance of the state estimate. Existing approaches include the singular vector technique [19, 110], model adjoint sensitivity techniques [10, 13], and ensemble-based methods [16, 17, 59, 94]. For ensemble filtering assimilation algorithms, for example, one can use the ensemble spread to determine where the model is most uncertain [83]. So far, targeted observing strategies have had mixed results [103]. And, to our knowledge, no one has studied targeted observing using point-vortex models.

All of the aforementioned data assimilation techniques have been developed in

the context of DNS simulations of large-scale atmospheric and oceanic systems; data assimilation into point-vortex models has received much less attention. Ide *et al.* [71, 72] have done some interesting work in this field. The algorithm developed by Ide *et al.* deduces the vortex positions by inverting the velocity superposition arguments that are built into the point-vortex equations and then assimilates that data into point-vortex models using Kalman filters. They have tested this strategy extensively in numerical simulations. They also developed a unique hybrid assimilation method that assimilates data about *both* the positions and strengths of vortices *and* the paths that tracer particles take through a flow. The basic idea is to augment the point-vortex equations (2.1) with a set of tracer advection equations that model the dynamics of particle movement [72]. The key here is that the fundamental link between velocity and vorticity couples these equations, so corrections made to one will "percolate" into the other. That is, one can assimilate tracer particle data into the advection equations and the cross-coupling term will naturally carry those corrections into the point-vortex equations. Ide *et al.* have studied this approach in simulations [72], but it has not yet been implemented with experimental data.

There has also been some prior work by Chen and Snyder[26] on ensemble assimilation of vortex positions and stregnths into a two-dimensional barotropic model. This group used vortex data provided by the National Severe Storms Laboratory (NSSL) and the Cooperative Institute for Mesoscale Meteorological Studies (CIMMS). NSSL extracts vortex positions and strengths from Doppler velocity data using a multi-step classification method that starts by searching for a particular shear profile in one dimension and then applies horizontal and vertical association techniques to identify two-dimensional and three-dimensional patterns that indicate vortex dynamics[98, 133]. CIMMS uses artificial intelligence techniques to identify vortices in infrared satellite imagery[108].

The data-assimilation schemes described in this section all use a periodic or con-

tinuous correction approach. That is, observations are assimilated either at prespecified intervals or whenever they become available. No attempt is made to determine when corrections might be most beneficial to the simulation, or when they are not useful. In the context of the real-time flow control applications that we are targeting, collecting and processing observations can be prohibitively expensive and/or invasive. If one could reduce this burden by collecting observations only when the system dynamics indicate that such observations are necessary for the accuracy of the simulation, it would be a major win. This thesis proposes and evaluates such a strategy and compares it to existing work.

Our work has some similarities to targeted observing: we are trying to use the dynamics of the system to determine when and where the model is likely to go astray. However, the goals of our research are quite different from those of adaptive observing. In adaptive observing, a fixed observing network is augmented with supplementary observations as dictated by the methods described in the previous section. The essential elements of the data assimilation approach, however, are unchanged by the information gleaned from the targeted observing strategies. That is, both the fixed and supplementary observations are still assimilated periodically or continuously. The dynamical information used to obtain the supplementary observations is not used to decide *when* to correct the model. Our approach is to use this dynamical information to intelligently adapt the correction timing in the point-vortex model.

In important contrast to this thesis work, most data assimilation research is not performed in a laboratory context. Geophysicists often rely on *Observing System Simulation Experiments (OSSEs)* to investigate new assimilation strategies. In OSSEs, a long model run is used to generate the observations, which are then modified to achieve the desired observational error characteristics. While these numerical experiments are a useful first step, there are many additional challenges in using real data. We are also starting with OSSEs, as described in Section 4, but we also study point-vortex data

assimilation using experimental data obtained from a planar air jet in our laboratory. These results are presented in Sections 12 and 13.

One of our first challenges in using the experimental data was to determine how to correct our model variables (vortex positions and strengths) with the velocity field data we collected in the lab. One approach to this problem is to preprocess the velocity data to extract vortex positions and strengths. A great deal of research has been devoted to techniques for doing just this. Most of the research in this field attempts to distinguish vortices from other types of coherent structures. One major problem is that vortices are surprisingly difficult to define [114, 138]. The obvious solution of simply thresholding the vorticity, for instance, picks out many different kinds of flow structures, not just vortices—particularly shear layers. Methods that search for local extrema in the vorticity field are somewhat more effective, but still prone to mis-identification [138]. The work of Jeong & Hussain [75] provides a cornerstone for much of the debate in the fluids literature about this topic, along with a useful definition that decomposes the velocity gradient tensor $\nabla\vec{v}$ into symmetric (S) and anti-symmetric (Ω) parts, the latter of which is an effective measurement of vorticity in an incompressible flow. Thus, searching for regions in which the norm of Ω dominates the norm of S can be an effective technique for identifying vortices; this is the Q-criterion of [68]. For two-dimensional flows, the Q-criterion is the elliptic version of the Okubo-Weiss criterion [107, 139] A variety of other approaches use $\nabla\vec{v}$ in different ways; [2] and [27], for instance, take its imaginary eigenvalues as evidence of local swirling motion. More recently, Haller has proposed a frame-independent method that releases numerical tracer particles in the flow, calculates their paths with an ODE solver, then computes the strain acceleration tensor at every point along each path. Particle paths that remain wholly in "elliptical" regions, where the flow does not stretch out, are then classified as vortices [58]. Many groups have worked out ways to fit velocity data to various analytical forms, such as wavelets [22, 46, 126] or orthogonal and Fourier decompositions [47, 109, 116], and then

use those decompositions to find the vortices.

Our problem is slightly different from the research described in the preceding paragraph. We are interested in vortex extraction for the purposes of correcting a point-vortex model. The goal in point-vortex modelling is to approximate the entire vorticity field with a collection of point-vortices. So, unlike the research described above, if shear layers were present in our flow, we would want point-vortices in our model to represent the vorticity in those layers. With this in mind, it is possible that a simple technique such as vorticity thresholding may work well for our purposes. In the next section, we explore and compare vorticity thresholding with the Okubo-Weiss method.

Chapter 3

Laboratory Planar Air Jet and PIV Measurements

The specific aim of this thesis is to investigate data assimilation into a point-vortex model and to develop and evaluate new targeted-observing strategies. Recall that the goal of targeted observing is to identify the times and locations where a correction will have a significant impact on simulation accuracy. We have devised several new targeted assimilation techniques that use information about the system dynamics to guide the correction process. The bulk of this thesis is devoted to analyzing these new strategies and comparing them to the traditional periodic correction approach to determine when and why one technique works better than another. Recall that for our target application domain—real-time modelling and control of fluid flows—computational cost is a vey important component of this analysis. We hope to develop a data assimilation technique that will enable us to sufficiently improve the accuracy of a point-vortex simulation without destroying its speed advantages by introducing an unecessary computational burden.

Although numerical experiments, such as the "Observing System Simulation Experiments" described on page 23 form an important component of our investigations, we are very interested in the real-world applicability of our results. With this in mind, our research group developed an experimental fluid flow in the laboratory[1] —a planar air jet [113]. We used this flow as the motivating example for the work presented in

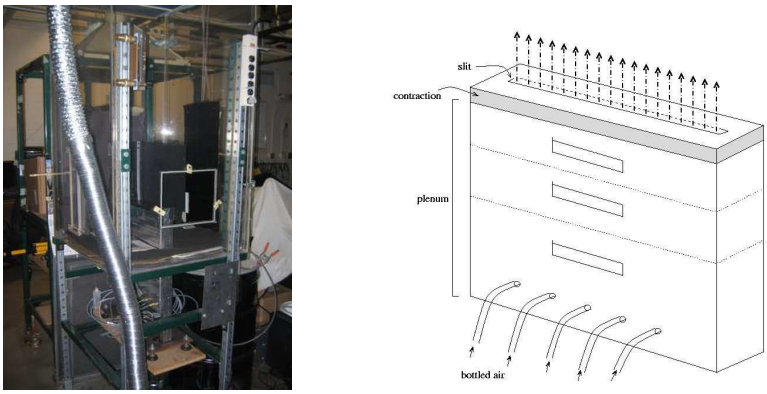
[1] This laboratory setup was in place prior to the start of this thesis

this thesis, and data collected from the jet was used to generate initial conditions for our numerical experiments. The laboratory setup and our data collection algorithms are described in detail in the following section.

Our experimental apparatus consists of a planar air jet and a set of actuators that can be used to selectively perturb the boundary layer of its flow field. Figure 3.1 is a schematic and a photograph of the jet. Compressed air is squirted into the bottom of the jet via a collection of hoses, five of which are visible in Figure 3.1. The air flows vertically through a plenum (the aluminum tower sections) and a contraction (the black piece on top) and emerges through a rectangular slit with dimensions of 2.50 ± 0.01 mm by 400 mm, giving an aspect ratio of 160:1. The plenum contains a variety of flow-conditioning devices that serve to reduce the turbulent intensity in the flow; the contraction is designed to further suppress turbulence and reduce non-uniformities in the velocity profile. Because the resulting jet is so long in the transverse direction, the flow field may be considered two-dimensional across its central cross-section; because it is so narrow, the entire jet is really a paired free shear layer.

Using actuators at the base of the jet [131], we can force the flow to assume one of its two unstable modes. Vortices are well-defined in both the sinuous and varicose modes, which makes the forced jet a good candidate for point-vortex modelling. There are different types of actuators that can provide the necessary forcing, including loudspeakers, piezoelectric actuators, or MEMS flaps [113]. For the data presented in this thesis, a single loudspeaker connected to a function generator was used to excite the varicose mode of the jet. This was accomplished by driving the loudspeaker at the natural frequency of the jet (16.83 Hz) [113]. A picture of the jet in its varicose mode is displayed in Figure 3.2.

We also have a mechanism for gathering velocity data from this flow—particle image velocimetry (PIV) [118]. A PIV system works by taking two successive photographs of a flow seeded with particles and cross-correlating these photographs to determine the



(a) (b)

Figure 3.1: A planar air jet. (a) is a photograph of the experimental apparatus that is used to house the jet. (b) is a schematic of the structures that condition the flow prior to its exit at the narrow slit. The purpose of the conditioning and the narrow slit is to ensure the uniformity of the flow along the length of the slit and enable an assumption of two-dimensional flow for any cross-section.

Figure 3.2: A planar air jet. Reynolds number ≈ 70. Synthetic jet actuator on one side at 1.94 KHz, modulated at 10 Hz. Vortices are clearly visible in this photograph of the jet.

displacements of the particles. Using these displacements and the time lag between the photographs, a velocity field can be constructed. To enable PIV measurements, some of the air that flows into the jet is seeded with theater fog. Then, a laser situated perpendicular to the jet slit is used to illuminate a cross-section of the flow. Theoretically, this cross-section should be the same along the length of the slit; this is the approximation that enables us to assume the flow is two-dimensional. The PIV camera equipment is directed at the front of the slit, so that it captures a photograph of the illuminated cross-section. For more details on the experimental setup, please see [48].

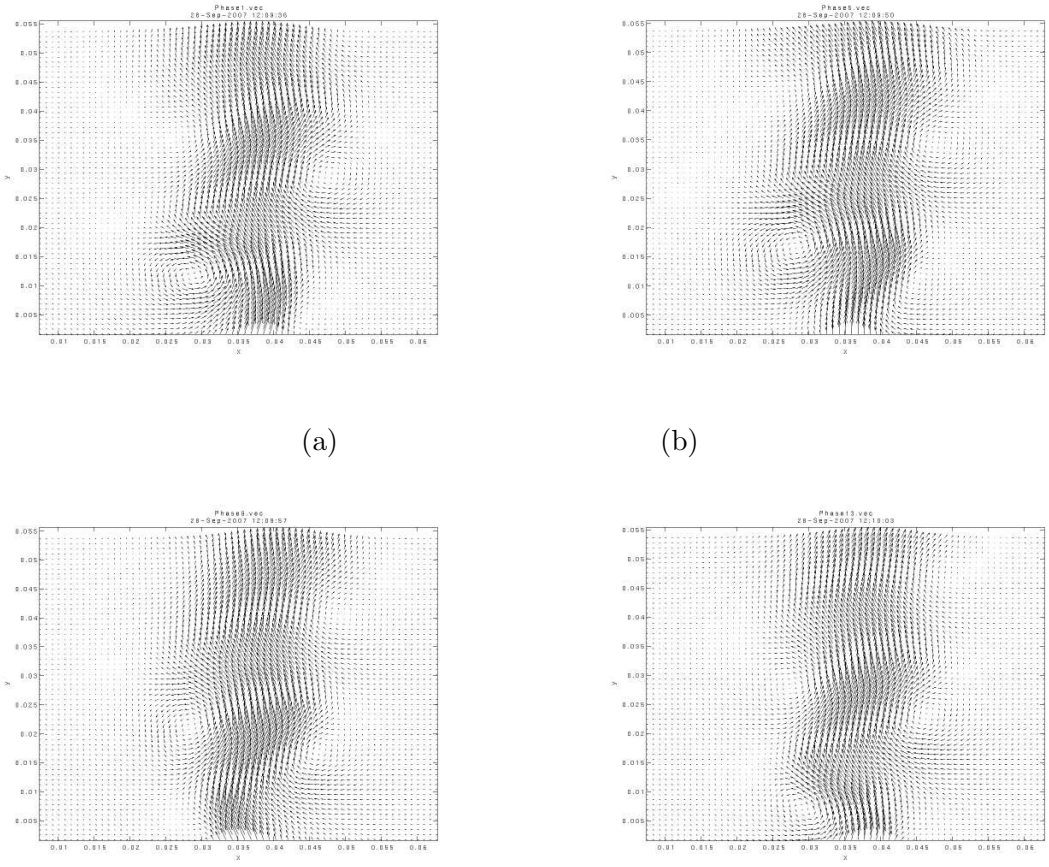
For our data-assimilation experiments, we would like to obtain a sequence of velocity field measurements in time. Note that when the jet is forced into one of its modes, the flow behavior is periodic, so an ideal sequence of measurements would include several different points in a single cycle of the flow. However, our observing equipment is much slower than the jet's natural frequency, so we are forced to take *phase-averaged* data [69]. For a single measurement, we set a particular phase shift relative to actuator timing for the measuring equipment, so that it measures the velocity field at the same offset in each forcing cycle for several cycles of the flow. Since the flow is periodic, we expect these measurements to be very similar, and we approximate a single velocity field measurement with the average of several hundred of these phase-locked fields. This is a common practice in the fluid dynamics community because it enables more fine-grained measurements of a rapidly changing flow than the observing equipment would permit. Figure 3.3 shows the phase-averaged data that we collected at four different points in the flow cycle, each computed by averaging 480 PIV measurements. The time lag between each image is approximately 15 milliseconds. The value of 480 was chosen using a convergence test: we kept doubling the number of fields used to compute the average until the RMS difference between averaged fields was not changing significantly. Given the 480 vector fields, we computed standard deviations for U and V at each gridpoint. These standard deviations ranged from roughly 0.01 m/s to 0.3 m/s, with a majority

of the values being <= 0.1 m/s. This gives us an approximate uncertainty in our PIV measurements of the velocity field.

It is interesting to consider how phase-averaged measurements affect the error in the resulting velocity fields. In a single PIV measurement, errors in the cross-correlation of particles in the PIV images are likely to manifest as spurious vectors in the flow field—vectors that differ significantly from the local velocity field [118, 128]. The process of averaging several hundred of these flow fields significantly reduces the impact of these spurious vectors. For our purposes, we will be using these flow fields to extract vortex positions and strengths, so we are interested in errors made in extracting these quantities from the phase-averaged data. One such error will occur because the vortices will be in slightly different positions in each of the 480 vector fields that we average to produce the images in Figure 3.3. This phenomenon is known as *phase jitter*, and the result is that a vortex in the averaged velocity field will be slightly larger and weaker and will have smoother edges than a vortex in single measurement. In Section 8, this error analysis will inform the assimilation experiments in which we add noise to simulated observations: knowledge about the types of errors we expect in our measurements will enable us to determine a reasonable expectation for the noise in our extracted vortex positions.

3.1 Vortex Extraction

After collecting the PIV measurements from the planar jet, our next step was to process the velocity fields into a format that could be used to correct the point-vortex model. As mentioned on page 21, we had two options here. We could use the velocity field observations to correct the model—performing the correction by mapping the model's point vortices into the observation space. Or, we could extract the vortex positions and strengths from the velocity field data and use that information to update the point-vortex state variables directly. We chose the latter approach because we felt



(a) (b)

(c) (d)

Figure 3.3: PIV measurements of the planar jet at approximately (a) 22.5 ° (b) 90 ° (c) 202.5 ∘ and (d) 292.5 ° into its cycle.

that performing the assimilation in the natural state space of the model would isolate our exploration of the dynamics of data assimilation from the complications introduced by mapping back and forth between state spaces. Also, reasoning about the flow field at a more-abstract level—in terms of its coherent vortices—simplifies the physics and makes it easier to understand the flow behavior.

As discussed on page 24, vortex extraction is a difficult problem, and there is a wealth of literature devoted to the topic. Some of this research attempts to distinguish vortices from other structures that have high vorticity, such as shear layers (e.g., [114, 138]). Others address the identification of vortices in a frame-independent fashion—something that is important for rotating flows and flows with interacting vortices [58]. Our problem is much simpler—we are working with a planar flow and phase-averaged observations in which the vortices are very well defined. Also, there are no shear layers present in our velocity observations. It is worth mentioning, however, that if shear layers were present, we would want our extraction technique to identify them. This is in stark contrast to the goals of the vortex extraction community, but it makes sense considering that the point-vortex model is modelling the entire vorticity field—not just the vortices. So, it is desirable to have point vortices distributed along a shear layer to accurately model the vorticity there. Given the well-defined, clearly separated vortices and the lack of shear layers in our PIV data, a fairly simple vortex extraction technique should be sufficient for our work.

In the paragraphs that follow, we evaluate two of the simplest techniques for two-dimensional vortex extraction: vorticity thresholding and the Okubo-Weiss method [107, 139]. Both of these techniques perform an analysis on the vorticity field. Recall that to compute the vorticity field we simply take the curl of the velocity field. In Figure 3.4, we reproduce the velocity fields from Figure 3.3 and display the corresponding vorticity field for each data set. Given the vorticity field, the vorticity thresholding technique applies the following algorithm to obtain the positions and strengths of the

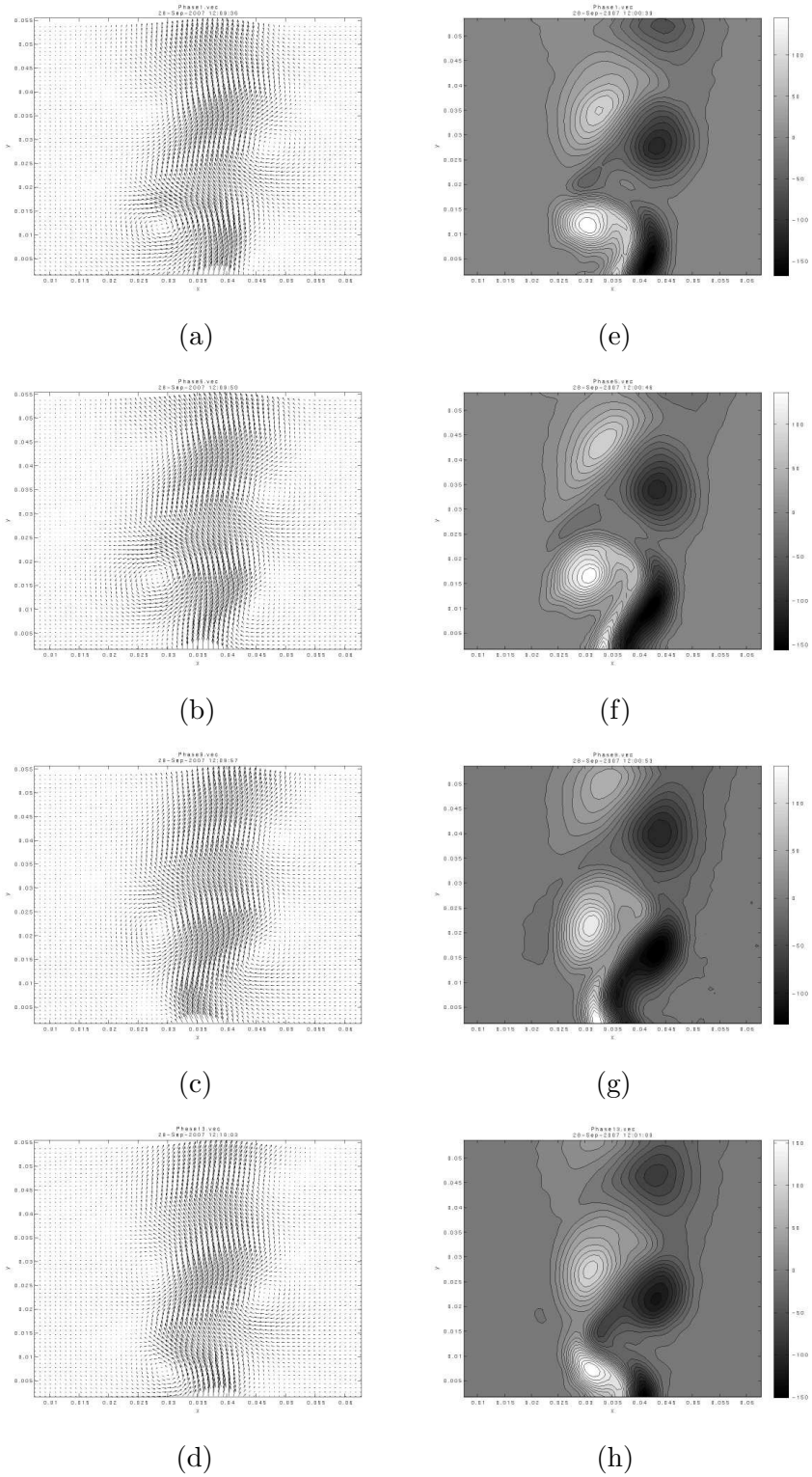


Figure 3.4: Vorticity fields from PIV data. (a)-(d) show velocity field data collected using particle image velocimetry at the times indicated in Figure 3.3. (e)-(h) are the corresponding vorticity fields, computed by taking the curl of the velocity field in the left column.

point vortices:

(1) Take the absolute value of the vorticity field and find its maximum value. Label this gridpoint (x, y).

(2) Let $\mathbf{V}$ represent the set of points that encapsulate a particular vortex. Initially, we set $V = \{(x, y)\}$. Then, starting from position (x, y), we construct a "connected component" for which all points in the component have absolute vorticity above the chosen threshold. For this analysis, a point is considered connected to four "neighbors"—one above, one below, one to the left, and one to the right. Label these points $(x, y+1), (x, y-1), (x-1, y)$, and $(x+1, y)$, respectively. We recursively check these neighboring points and include a given point in the component if the absolute vorticity at the point exceeds the threshold. That is,

$$\begin{aligned}
&\text{if } |\omega| \text{ at } (x, y+1) > T \quad \text{then} \quad \mathbf{V} = \mathbf{V} \cup \{(x, y+1)\} \\
&\text{if } |\omega| \text{ at } (x, y-1) > T \quad \text{then} \quad \mathbf{V} = \mathbf{V} \cup \{(x, y-1)\} \\
&\text{if } |\omega| \text{ at } (x-1, y) > T \quad \text{then} \quad \mathbf{V} = \mathbf{V} \cup \{(x-1, y)\} \\
&\text{if } |\omega| \text{ at } (x+1, y) > T \quad \text{then} \quad \mathbf{V} = \mathbf{V} \cup \{(x+1, y)\}
\end{aligned}$$

where $|\omega|$ is the absolute vorticity and T is the chosen vorticity threshold.

(3) Compute the average vorticity for the points identified in (2) by summing the vorticity at all points and dividing by the number of points. Mathematically,

$$\omega_{av} = \frac{\sum_{(a,b)\in\mathbf{V}} \omega_{(a,b)}}{|\mathbf{V}|}$$

where ω_{av} is the average vorticity over the region and $\omega_{(a,b)}$ is the vorticity at gridpoint (a, b). Multiply this by the area covered by the points in (2) to get an approximate circulation, or strength, for the vortex.

(4) Place a point vortex at the location identified in (1) with the strength computed in (3).

(5) Remove the points identified in (2) from consideration and repeat the process to find the next vortex.

Note that Step (3) amounts to an application of Gauss's theorem to obtain an approximation of the vortex circulation[32, 111]. Recall that the circulation is the line integral of the velocity field around a closed curve. Gauss's theorem allows us to instead take the surface integral of the vorticity to calculate the strength. We use the resulting approximation that

$$\omega_{av} = \frac{\Gamma}{A}$$

where ω_{av} is the average vorticity in the region identified in (2), A is the area of the region, and Γ is the circulation or strength of the vortex enclosed by the region.

Step (2) of the vorticity thresholding algorithm requires that we define a threshold value T for the vorticity. We chose

$$T = |0.15\ [\ \max(\omega) - \min(\omega)\ ]|,$$

where $\max(\omega)$ and $\min(\omega)$ are the maximum and minimum values of vorticity over all gridpoints in the data set. We stopped "growing" the connected component when the absolute vorticity at a particular location was relatively close to zero. For the results presented in this section, the threshold was chosen by hand-tuning T until the results looked reasonable—i.e., we chose a threshold for which the vortex boundaries identified by the algorithm were similar to the vortex boundaries we might draw by hand on the vorticity plots—and our choice of threshold is obviously fairly arbitrary. The point is to choose a threshold that will enable the connected component algorithm to include as much area as possible for each vortex without accidentally grouping two distinct vortices together. An underlying assumption here is that the vortices are well-separated by regions of low vorticity. Note that this is the case for our planar jet PIV data, but may not be true for other types of vorticity data. However, our goal here is not to provide a vortex extraction result that is fully generalizable; rather, the focus of this

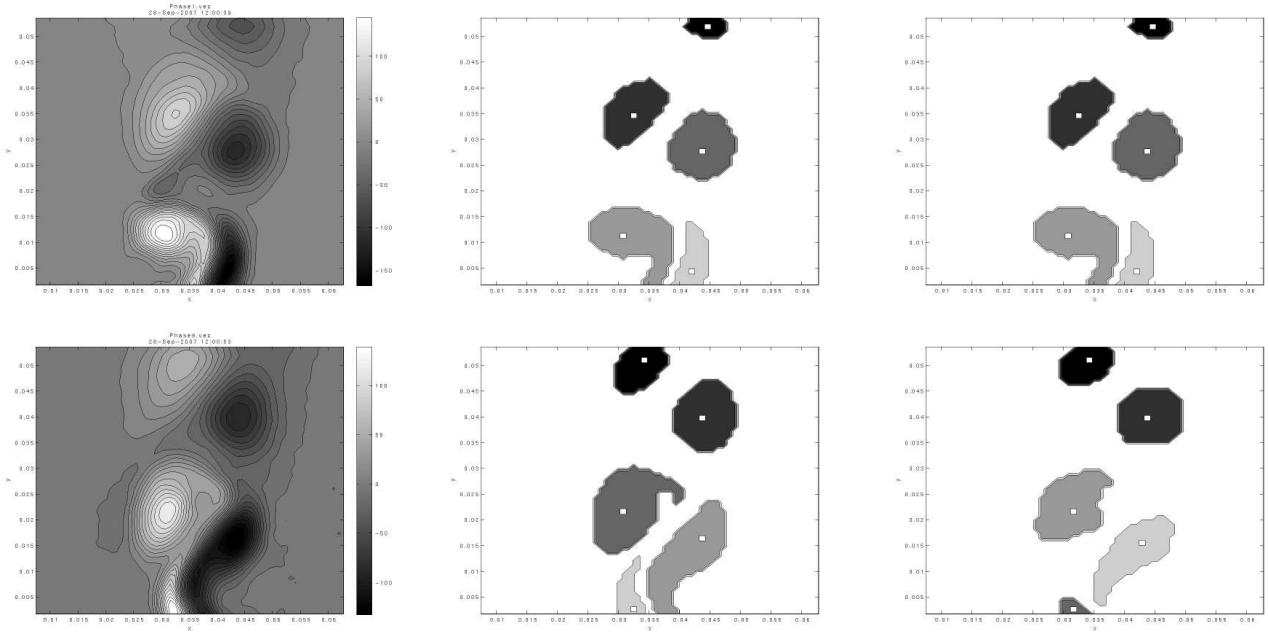
thesis is to use the extracted vortex positions to correct the point-vortex model and study data-assimilation dynamics.

Vorticity thresholding is an obvious technique for locating high-vorticity regions, but it is not so clear how to define the vortex boundaries after the threshold has been applied. Recall that our objective is to distill the information in the vorticity plots into a set of point-vortex positions and strengths for the point vortex model. Many of the vortex extraction techniques described on page 24 provide sophisticated ways of distinguishing vortices from other parts of the flow, but the goal is usually to plot the results and demonstrate visually that the method identifies vortices appropriately. None of these references discusses how to partition the data into individual vortices and assign strengths. Our approach to this problem, described in Step (2), is to define a single vortex by searching for a connected-component of high vorticity. This approach calls on our previous work in using connected components for topological signal separation [120].

The Okubo-Weiss criterion [107, 139] is a more-reliable means of vortex identification in the sense that it is invariant in translating frames of reference (Galilean invariance). It is also more computationally intensive than the vorticity thresholding technique. The Okubo-Weiss algorithm looks for regions where the squared rate of rotation, $|\Omega|^2$, dominates the squared rate of strain, $|S|^2$, where

$$\begin{aligned} \Omega &= \frac{1}{2}[(\bigtriangledown v) - (\bigtriangledown v)^T] \\ S &= \frac{1}{2}[(\bigtriangledown v) + (\bigtriangledown v)^T]. \end{aligned}$$

In these regions, the flow behavior is "elliptical" in nature; outside these regions, the fluid motion is "hyperbolic." It is worth mentioning that Haller's method [58] also identifies elliptical regions and is invariant under *any* type of coordinate change. Haller's method works by sprinkling fluid "particles" throughout the flow and measuring which of their trajectories stay in the identified elliptic regions. There are also higher-order



(a) (b) (c)

Figure 3.5: Vortex extraction. (a) vorticity fields from Figures 3.4(e) and 3.4(g). (b) "connected components" where absolute vorticity exceeds a threshold value (as described in Step (2) of the vorticity thresholding algorithm). The small white squares inside each "component" indicate the position where the point vortex was placed. The strengths of the vortices (in top to bottom order) were -0.0012, 0.0063, -0.0085, 0.012, and -0.0058 for the data set on top and 0.0030, -0.0069, 0.011, -0.012, and 0.0034 for the data set on bottom. (c) "Connected components" where the Okubo-Weiss criterion was positive. The strengths of the vortices (in top to bottom order) were -0.0015, 0.0063, -0.0081, 0.0011, and -0.0035 for the data set on top and 0.0028, -0.0063, 0.0097, -0.011, and 0.0013 for the data set on bottom.

corrections to the Okubo-Weiss criterion that include acceleration terms [65, 66]. And, the Okubo-Weiss method has been shown to be inaccurate if the velocity gradient tensor is time-varying [125]. Although incorporating higher-order correction terms or using Haller's method might yield higher accuracy, the additional computational cost that would be incurred is not justified for our planar jet PIV data. We are working with a two-dimensional flow with very well-defined irrotational vortices.

The specific steps in our implementation of the Okubo-Weiss algorithm are similar to the vorticity thresholding algorithm:

(1) Compute $W = [|\Omega|^2 - |S|^2]$. Remove from consideration any gridpoints (a, b) for which

$$W_{(a,b)} \leq 0.05\,[\max(W) - \min(W)].$$

These points are removed from consideration by setting $W_{(a,b)} = 0$.

(2) Search for the maximum positive value of W.

(3) Starting from the point identified in (2), compute the "connected component" where $W > 0$ (as described in Step (2) of the vorticity thresholding algorithm).

(4) Compute the average vorticity for the points identified in (3) by summing the vorticity at all points and dividing by the number of points. Multiply this by the area covered by the points in (3) to get an approximate circulation, or strength, for the vortex.

(5) Place a point vortex at the location identified in (2) with the strength computed in (4).

(6) Remove the points identified in (3) from consideration and repeat the process starting with Step (2) to find the next vortex.

The Okubo-Weiss criterion defines a vortex as any region where $W > 0$. However, we remove some of these points from our vortex extraction algorithm in Step (1).

Specifically, we remove any points where W is less than 5% of its dynamic range. The purpose of doing this is to distinguish the vortices from one another. In our data sets, some of the vortices are very close together, and the elliptical regions identified in the Okubo-Weiss algorithm can run together. Setting this threshold for W prevents the algorithm from grouping two separate vortices into a single connected component in Step (3).

Since algorithmic speed is a major concern for our application domain, a comparison of the computational complexity of the vorticity thresholding and Okubo-Weiss algorithms is in order. Notice that the algorithms share some similarities. For each vortex identified, they compute a maximum value and then find a connected component that defines the vortex. The maximum computation has $O(nm)$, where n is the number of gridpoints in the x direction and m is the number of gridpoints in the y direction. The complexity of the connected component algorithm for vortex i has order equal to the number of gridpoints in its connected component—label this g_i. The connected components covered by all of the vortices are never larger than the entire domain, which has size nm, so

$$\sum_{i=1}^{p} g_i \leq nm,$$

where we have let p equal the number of vortices identified by the algorithm. So, the overall complexity of the vortex identification loop for both algorithms is dominated by the maximum computation and is equal to $O(pnm)$. However, the Okubo-Weiss algorithm also requires an additional computation of the gradients $dU/dx, dU/dy, dV/dx$, and dV/dy. These computations are each $O(nm)$, so in traditional complexity theory, this extra work would get absorbed into the $O(pnm)$ of the algorithm. However, for our special application domain—real-time flow control—the extra $O(4nm)$ incurred by the gradient computation may make a significant difference[2] . Especially for flows like the

[2] Notice here that we have purposely *not* dropped the constant 4 from the big-Oh notation because this multiplier might be significant, as well.

planar jet, where p is equal to four or five, the gradient computation basically doubles the cost of the algorithm.

We applied the vorticity thresholding and Okubo-Weiss algorithms described above to two of the vorticity fields from Figure 3.4; the results are presented in Figure 3.5. The connected components and point vortex positions identified by Okubo-Weiss appear to be a fairly close match to the results of the vorticity thresholding algorithm. How can we tell which technique is better? If the results of vorticity thresholding technique are comparable to the Okubo-Weiss method, we can avoid the additional computational cost incurred by Okubo-Weiss. In the next section, we consider quantitative evaluation methods for these vortex extraction techniques.

Although the vorticity thresholding and Okubo-Weiss techniques are well-known vortex identification schemes, we are not aware of any research that uses these techniques to extract vortex positions for use in point-vortex modelling. And, as mentioned previously, we have augmented these techniques with a topological approach [120] to defining the vortex boundaries. Perhaps most importantly, few research groups have explored using vortex extraction techniques to collect data for point-vortex data assimilation. These are all interesting and important contributions of this thesis.

3.2 Validation and Evaluation of Vortex Extraction

Now that we have extracted point-vortex information from the laboratory PIV observations, it is worthwhile to perform some analysis on the results. First, we can consider how accurately a given collection of point vortices represents the true flow dynamics. A natural way to measure this accuracy is to compare the velocity field induced by the collection of point vortices to the original PIV velocity field. Recall from Section 2.1 that, given a collection of point vortices, we can compute the velocity at any point in the domain by summing over the velocities induced by the individual vortices. To compute the velocity (U, V) induced at location (x, y) from a single point vortex at

position (x_1, y_1), we could use the point-vortex equations:

$$\begin{aligned} U(x,y) &= \frac{x_1 - x}{2\pi r^2} \\ V(x,y) &= \frac{y - y_1}{2\pi r^2} \end{aligned}$$

where $r^2 = (x - x_1)^2 + (y - y_1)^2$. However, in our connected component algorithm, we have calculated the vorticity of each point vortex by summing vorticities over a region of space. So, when we reconstruct the velocity field, it makes sense to give each point vortex a small "radius". There is also a singularity at $(x, y) = (x_1, y_1)$ in the point-vortex equations[3] . To address these two issues, we use a "blob model" to compute the induced velocity field. The blob model introduces a radius, r_c, for each vortex. For points outside this radius, the induced velocity is given by the point vortex equations above; inside the radius, the dynamics are treated as equivalent to solid body rotation, i.e.,

$$\begin{aligned} U(x,y) &= \frac{r}{r_c}\frac{x_1 - x}{2\pi r_c^2} \\ V(x,y) &= \frac{r}{r_c}\frac{y - y_1}{2\pi r_c^2} \end{aligned}$$

Basically, r/r_c is a linear attenuation function on the growth of the velocity near the vortices. The velocity attains its maximum value at $r = r_c$ and then linearly decreases to zero at the center of the vortex blob. For the results presented in this section, we chose $r_c = 0.005$, which roughly approximates the radius of the vortices observed in our PIV data. To determine this radius, we measured the distance between the center of each vortex and the location where the transverse velocity induced by that vortex attains its maximum value. This is a rough approximation that was obtained by visually inspecting vortices present in the PIV velocity fields.

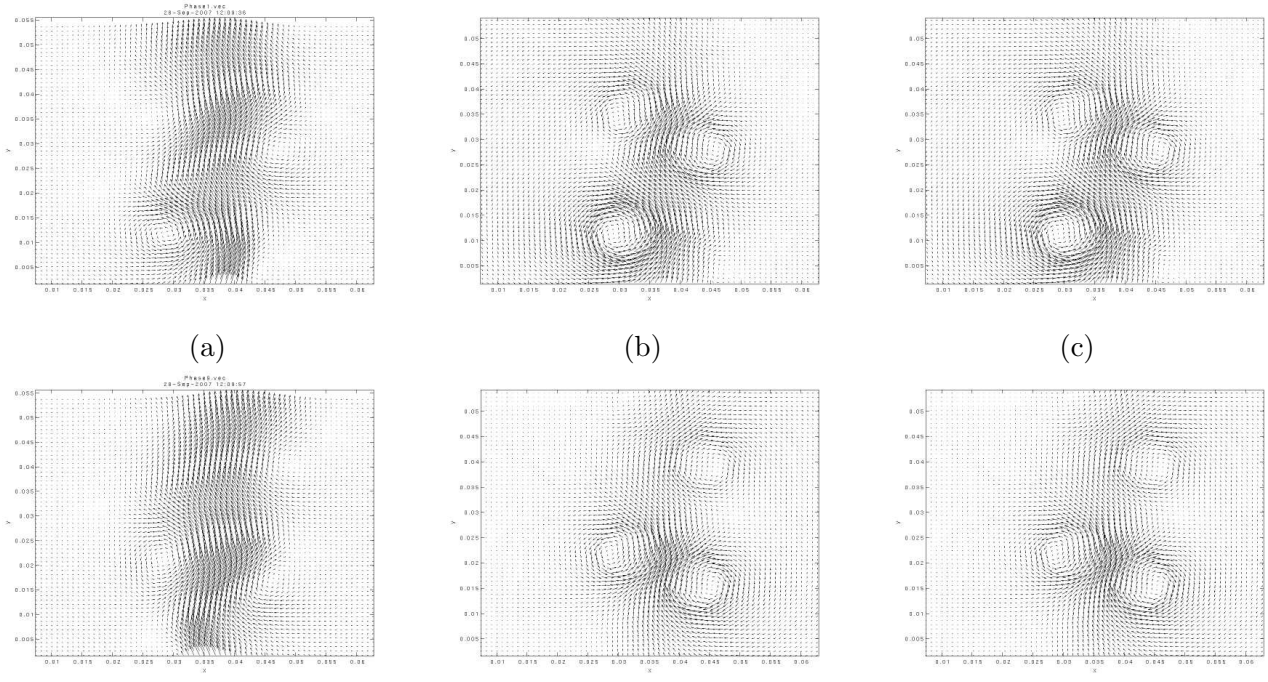
In Figure 3.6, we present the results of using the blob model equations to compute the velocity field induced by the point vortices identified in Figure 3.5. Again, the point

[3] The induced velocity approaches infinity as $(x, y) \to (x_1, y_1)$.

Figure	Data Source	U Range	V Range	$\|\mathbf{U^2 + V^2}\|$ Range
3.6(a)	PIV	[-0.614,0.408]	[-0.266,1.18]	[0.00261,1.22]
3.6(b)	Vort. Thresh.	[-0.339,0.375]	[-0.299,0.483]	[0.000878,0.517]
3.6(c)	Okubo-Weiss	[-0.325,0.334]	[-0.283,0.418]	[0.000436,0.434]
3.6(d)	PIV	[-0.392,0.383]	[-0.233,1.11]	[0.00144,1.16]
3.6(e)	Vort. Thresh.	[-0.364,0.314]	[-0.275,0.572]	[0.00182,0.584]
3.6(f)	Okubo-Weiss	[-0.295,0.295]	[-0.244,0.489]	[0.00462,0.536]

Table 3.1: Quantitative velocity field comparison. For each of the velocity fields presented in Figure 3.6, we computed the range of the horizontal and vertical components of the velocity field as well as the range of the magnitude of the velocities. We can compare these dynamic ranges to gauge how successful our vortex extraction techniques were.

here is to see how well our vortex extraction algorithms reproduce the original velocity field. In the left column, we display the velocity fields from the PIV system. The middle and right columns display the velocity fields induced by the point vortices extracted using the vorticity threhsolding and Okubo-Weiss algorithms, respectively. Our first step in comparing these results was to perform a simple sanity check for our vortex extraction algorithms by comparing the dynamic range of the velocities in each data set. Table 3.1 displays the range of velocity magnitudes for the six data sets presented in Figure 3.6. The velocity magnitudes in each of the three cases are fairly comparable for the two data sets we studied. We do not expect them to be exact because reducing the vorticity field to five point vortices is a significant approximation. However, recall that the standard deviations for U and V for the set of 480 averaged velocity fields were in the range 0.01 m/s to 0.3 m/s. For the U velocities, the maximum and minimum values in the induced velocity data sets are within one standard deviation of the PIV maximum and minimum. For the V velocities, the minimum values are close between the data sets, but the maximum values in the induced velocity fields are too low. That is, we are underestimating the vertical velocity in some cases. This is likely due to our approximation of the entire vorticity field as a collection of three or four "blob"-vortices. In comparing the velocity fields in Figure 3.6, we can see that the induced



(a) (b) (c)

(d) (e) (f)

Figure 3.6: Evaluating vortex extraction. (a) and (d) are the velocity fields captured with our PIV system at 22.5° and 202.5° phase shifts. (b) and (e) are velocity fields induced by the point vortices extracted from (a) and (d) via the vorticity thresholding method. (c) and (f) are the velocity fields induced by the point vortices extracted from (a) and (d) using the Okubo-Weiss method.

Figure	Data Source	U MSE	V MSE	$\lvert \mathbf{U^2 + V^2} \rvert$ MSE
3.6(b)	Vort. Thresh.	0.0104	0.0779	0.0828
3.6(c)	Okubo-Weiss	0.00991	0.0835	0.0874
3.6(e)	Vort. Thresh.	0.00772	0.0741	0.0749
3.6(f)	Okubo-Weiss	0.00777	0.0845	0.0853

Table 3.2: Comparison of extraction techniques. For each of the velocity fields presented in Figure 3.6, we computed the mean-squared error between the reconstructed velocity field and the original PIV velocity data. Based on this error metric, vorticity thresholding appears to be more successful than the Okubo-Weiss technique.

velocity fields do not replicate the stream characteristics of the jet. This suggests that we may need to improve our approximation of the vorticity field by adding more point-vortices. In Section 3.3, we explore one technique for doing this; refining the point-vortex distribution will also be an important component of our future work.

The velocity ranges presented in Table 3.1 do not provide much information about which vortex extraction technique works better. It is more meaningful to compute a mean-squared error measure between the PIV data and the induced velocity fields from each method; the results of these computations are presented in Table 3.2. We can again compare these values to the range of standard deviations for the phase averaged velocity fields, 0.05 m/s to 0.3 m/s. Since the RMS error for each induced velocity field measure falls within these ranges, we can conclude that we are reproducing the original velocity field fairly well. Also, in comparing vorticity thresholding to the Okubo-Weiss method, we see that the MSE measures for the two methods are very similar. Any difference in the two methods is negligible when you consider the uncertainty introduced by the arbitrary thresholds defined in the two algorithms. Still, for the type of observations we have from the planar jet, we can conclude that simple vorticity thresholding is adequate for vortex extraction. This is a positive result for our work, because we can save the additional computational costs of computing the gradients required by the Okubo-Weiss method. This cost savings may not be significant in many applications, but for real-time

modelling and control of fluid flows, it might be critical.

3.3 Quantization of Large Vortices

It may be the case that distilling the vorticity fields in Figure 3.4 down to five point vortices is too crude an approximation even with the added power of data assimilation. We can obviously improve the approximation by distributing more point vortices—in fact, if we place a point vortex at each gridpoint, we can model the discretized vorticity field exactly. However, an increase in the number of point vortices translates to an increase in the number of state variables in our point-vortex model, so we are faced with a computational cost versus accuracy tradeoff. A reasonable middle ground is to distribute *several* smaller point vortices in the vicinity of each vortex such that their superposition models the large-scale vortex. There are several ways to accomplish this—for example, we could take each vortex region in Figure 3.5(b) and assign a point vortex to each gridpoint in the region, with the strength equal to the vorticity value at that gridpoint. Or, we could choose a fixed quantum for the strength of each point vortex and distribute as many as are required to represent the local vorticity field. We chose the latter approach because keeping the strength of the point vortices fixed has some positive implications for point-vortex modelling. First, working with a fixed vortex strength allows us to remove 1/3 of our state variables, decreasing our computational costs and storage requirements. Also, since the vortex strengths are fixed in the two-dimensional point-vortex model, sprinkling smaller point vortices throughout the domain allows the model to move vorticity around at a more fine-grained level.

Our approach to assigning the fixed-strength point vortices was implemented as follows. Recall that the vortex extraction techniques of Section 3.1 identify a connected component for each large-scale vortex. We chose the weakest vortex to determine a strength quantum for our point vortices, selecting a value that would enable us to assign two point vortices to the weakest vortex. We then applied the following algorithm to

each vortex component:

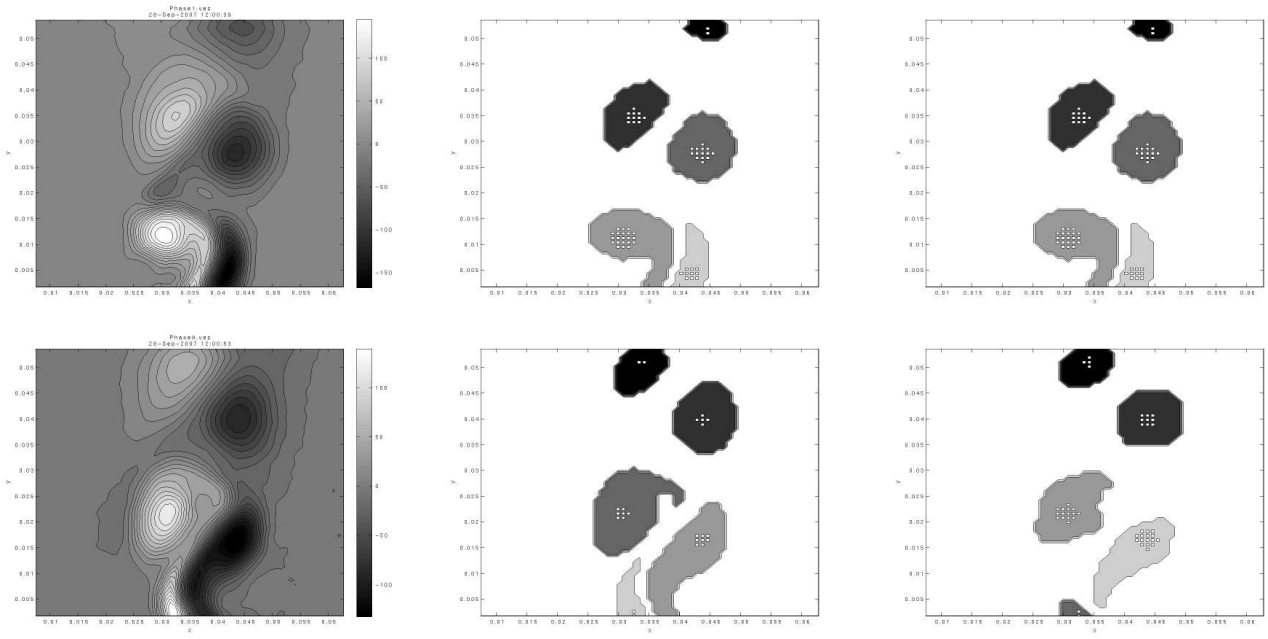
(1) Find the maximum value of absolute vorticity in the area covered by the vortex.

(2) If there is not already a point vortex at location (1), place one there. If there is, find the closest uncovered point (by Euclidean distance) and place the point vortex there.

(3) Subtract the vortex quantum from the absolute vorticity at the location in (1) and repeat

Quantization of the vorticity field into small point vortices enables the point-vortex model to advect the vorticity at a more fine-grained level than could be accomplished without such quantization. Since the two-dimensional point-vortex equations do not change the strength of the point-vortices in the simulation, having more point-vortices provides the ability to achieve more-realistic vorticity distributions. The quantization approach that is common in the fluids literature is to place one point vortex at each gridpoint in the discretized vorticity field. However, this is exactly what we are hoping to avoid, because we are interested in the point-vortex model for its speed advantages and small state vector. As we increase the number of point vortices in our simulation, we lose these advantages. Our quantization scheme is an attempt to reach a compromise between accuracy and computational cost. We integrated our quantization approach into both the vorticity thresholding and Okubo-Weiss algorithms; the placement of the point vortices for each case are shown in Figure 3.7. Note that the number of vortices depends both on the quantum determined for each data set and on the strength of the vortices identified in the vortex extraction algorithm.

To analyze these point-vortex distributions, we again computed the induced velocity fields for each data set and compared them to the PIV velocity fields. The induced fields were computed using the blob model as described on page 41. Side-by-side plots of these velocity fields are presented in Figure 3.8; the original PIV data is in the left



(a) (b) (c)

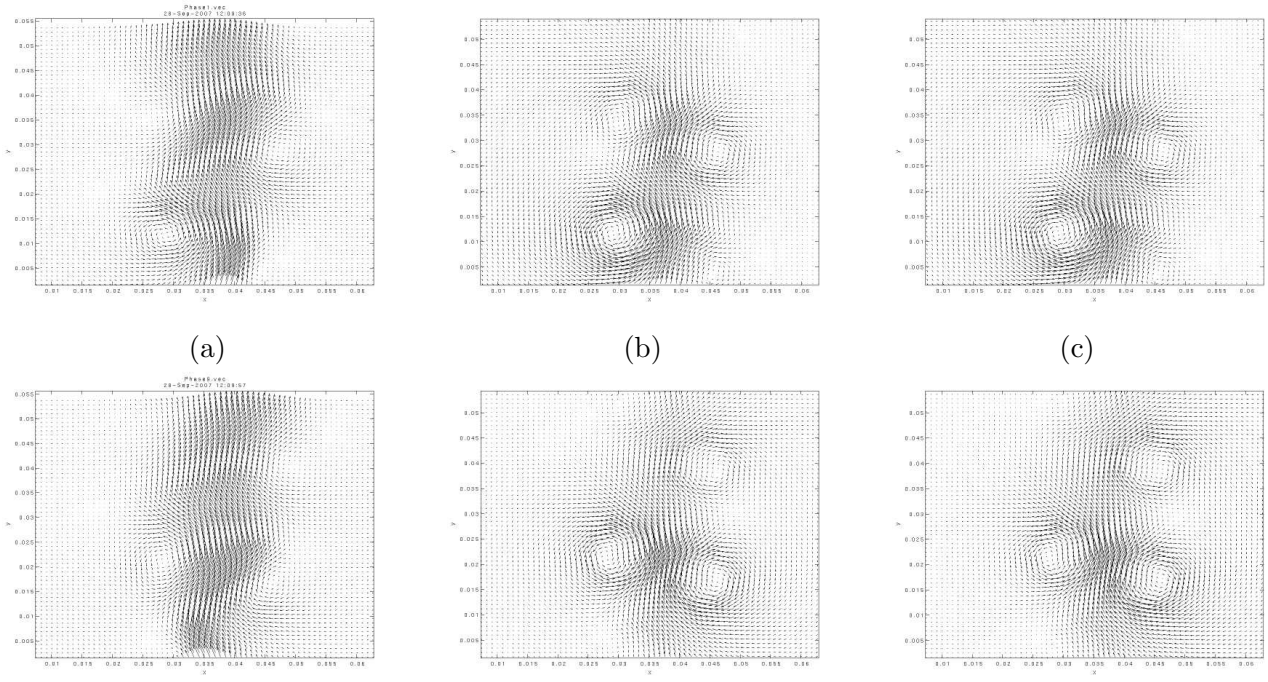
Figure 3.7: Quantization of vortices. We reproduce the data from Figure 3.5 and display how our quantization algorithm would place multiple vortices with fixed strengths to represent each large-scale vortex. (a) Vorticity fields from Figure 3.4(e) and (g). (b) "Connected components" where absolute vorticity exceeds a threshold value, as described in Step (2) of the vorticity thresholding algorithm on page 35. The small white squares inside each "component" indicate the position(s) where each point vortex was placed. The strength quantum chosen for the vortices was ± 0.00059 for the data set on top and ± 0.0015 for the data set on bottom. (c) "Connected components" where the Okubo-Weiss criterion was positive. The strength quantum chosen for the vortices was ± 0.00075 for the data set on top and ± 0.00067 for the data set on bottom. The point-vortex strength for each data set was chosen such that the weakest vortex region would be assigned two point vortices.

Figure	Data Source	U Range	V Range	$\|\mathbf{U^2} + \mathbf{V^2}\|$ Range
3.6(a)	PIV	[-0.614,0.408]	[-0.266,1.18]	[0.00261,1.22]
3.7(b) - top	Vort. Thresh.	[-0.261,0.303]	[-0.231,0.414]	[0.000475,0.455]
3.7(c) - top	Okubo-Weiss	[-0.265,0.279]	[-0.223,0.364]	[0.000773,0.378]
3.6(d)	PIV	[-0.392,0.383]	[-0.233,1.11]	[0.00144,1.16]
3.7(b) - bottom	Vort. Thresh.	[-0.318,0.255]	[-0.232,0.519]	[0.000787,0.531]
3.7(c) - bottom	Okubo-Weiss	[-0.243,0.219]	[-0.188,0.459]	[0.00116,0.476]

Table 3.3: Quantitative velocity field comparison. For each of the velocity fields presented in Figure 3.8, we computed the range of the horizontal and vertical components of the velocity field as well as the range of the magnitude of the velocities. We can compare these dynamic ranges to gauge how successful our vortex extraction techniques are.

column and the vorticity thresholding and Okubo-Weiss results are in the center and right columns, respectively. Visually comparing these induced velocity fields to those in Figure 3.6, it does not seem that the quantization into smaller point vortices has a significant impact on the induced velocity fields. However, the velocity fields induced by the quantized point-vortices do seem to replicate slightly more of the detail in the original PIV data.

We can also perform a quantitative analysis on the quantized point-vortex data. Table 3.3 provides a comparison of the velocity dynamic ranges for each of the data sets in Figure 3.8, and Table 3.4 provides the MSE measures between the induced velocity fields and the original measured field. Interestingly, comparing these values to those in Tables 3.1 and 3.2, one sees very little improvement in accuracy. Purely for accuracy reasons, it does not seem worth the effort to quantize the large vortices. However, as mentioned previously, quantization does have modelling implications as well. In Section 12, we explore these issues with the data set in Figure 3.3(c) via some numerical experiments with the vorticity thresholding technique. In particular, we perform experiments with both the large point vortices and the smaller clusters of vortices described in this section, to determine whether the modelling implications of a finer-grained vorticity distribution justify the costs of increasing the number of point



(a) (b) (c)

(d) (e) (f)

Figure 3.8: Evaluating vorticity quantization. (a) and (d) are the velocity fields captured with our PIV system at 22.5° and 202.5° phase shifts. (b) and (e) are velocity fields induced by the point vortices in column (b) of Figure 3.7. (c) and (f) are velocity fields induced by the point vortices in column (c) of Figure 3.7.

Figure	**Data Source**	**U MSE**	**V MSE**	$\|\mathbf{U^2 + V^2}\|$ **MSE**
3.7(b) - top	Vort. Thresh.	0.0106	0.0787	0.0837
3.7(c) - top	Okubo-Weiss	0.0107	0.0824	0.0879
3.7(b) - bottom	Vort. Thresh.	0.00766	0.0750	0.0752
3.7(c) - bottom	Okubo-Weiss	0.00869	0.0823	0.0848

Table 3.4: Decomposition of large vortices. After applying our algorithm to deconstruct the large vortices from Figure 3.5 into smaller point vortices, we again computed the induced velocity field for each vortex configuration. We then computed the mean-squared error between the reconstructed velocity field and the original PIV velocity data. If we compare these results to those in Table 3.2, it appears that vortex quantization does not improve the ability to reproduce the original velocity field.

vortices.

Chapter 4

Numerical Experiments

Our ultimate goal is to use the laboratory setup described in the preceding sections to evaluate point-vortex data assimilation with real observations and determine the best strategy for correcting this type of model with data from these kinds of flows. The work presented in this thesis takes several important steps toward attaining that goal. First, we have devised and performed several numerical experiments to analyze and compare various point-vortex data-assimilation strategies. These initial experiments provide an opportunity to explore the dynamics of point-vortex data assimilation without addressing the inevitable complexities of using real data. The results of this exploration are presented in Section 7. Next, we augmented our numerical experiments with observational noise in anticipation of one of the more significant challenges of using real data. In Section 8, we present a detailed analysis of the effects of this observational noise on the data-assimilation schemes we have developed. As part of this analysis, we consider whether it is necessary to use a more-sophisticated assimilation algorithm to mitigate the negative effects of the noise. We explore Newtonian nudging–a very simplistic scheme for weighting observations–in Section 10. A more-sophisticated scheme that has gained popularity in the meteorological and geophysics communities is ensemble Kalman filtering. In Section 11, we present an analysis of targeted observing using ensemble Kalman filtering with the point-vortex model. This analysis was performed using NCAR's Data Assimilation Research Testbed (DART). Finally, in Section 13,

we present preliminary results for a simulation that uses the experimental data in the assimilation.

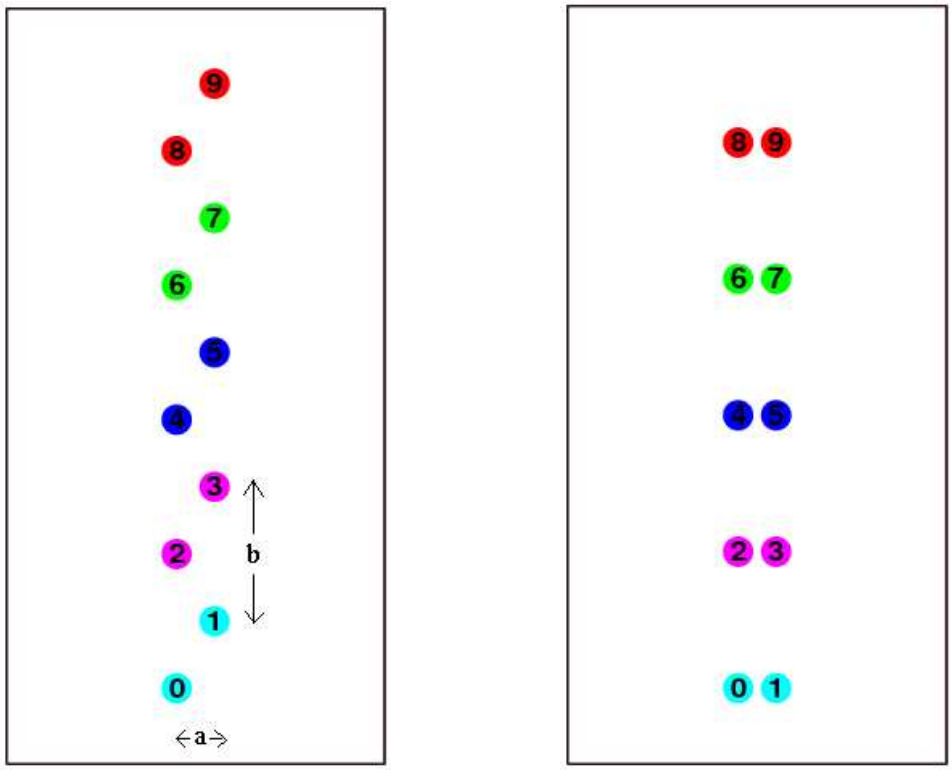
In the initial stages of our research, we devised several numerical experiments to facilitate our study of point-vortex data assimilation and targeted observing. The basic idea here is common in the numerical computing community: use one simulation as an ansatz for the "true" behavior of the system—call this simT—and correct a "modified" simulation—simM—using "observations" extracted from the truth run. The goal is to construct the parameters of these two simulations so that the difference between them is a reasonable estimate of the difference between the real world and our model of it. This is very difficult in practice because (1) the model is imperfect, so there is no way to get a simulation that accurately represents the real world and (2) we do not have a detailed understanding of how the model misrepresents the real world, so we do not know how to make simM differ from simT in a physically meaningful way. The standard approach is to simplify this problem by making an assumption that the model is perfect and that observations have additive, zero-mean, Gaussian noise. With these assumptions, simT is generated by running the model from a representative initial condition, I. Noise is added to observations collected from this truth run, and these observations are used to correct simM, which is initialized by perturbing the initial condition I (usually by adding Gaussian noise). This technique is referred to in the data-assimilation literature as a "perfect-model experiment" or "Observing System Simulation Experiment" (OSSE).

Another approach, which we will refer to as an "imperfect-model experiment", is to attempt to generate a simple representation of model error and incorporate it into the design of the assimilation experiment. Our approach is as follows: run a high-resolution simulation to represent the "truth", simT, and a coarser-resolution simulation as the "model", simM. Using a larger timestep for simM introduces numerical errors into the simulation, causing it to differ significantly from simT. This is much like what happens when a simulation diverges from reality, as floating-point error and physical noise have

many of the same effects, and the numerical errors in both cases are compounded by the model integration. It is our opinion that an imperfect model experiment is a more stringent test of any data-assimilation algorithm, so most of our numerical experiments have been performed using this technique. However, we have also conducted some perfect model experiments using the ensemble Kalman filtering software package, DART. These results are presented in Section 11.

Another challenge in improving realism in any numerical experiment is to find appropriate initial conditions for the simulations. Ideally, these initial conditions should be derived from measurements of the physical system under study. In our case, we could use the point-vortex data extracted from our PIV measurements of the planar air jet. In the initial stages of this thesis work, these PIV measurements were not available, so we attempted to construct realistic initial conditions that were similar to what we expected to observe with the PIV equipment; these are displayed in Figure 4.1. Much of our analysis in the rest of this section was performed with these initial conditions. Once we obtained real data from the PIV system, we were able to extend our analysis to vortex configurations obtained directly from the PIV measurements; the results of this exploration are the subject of Sections 12 and 13.

Figure 4.1 displays two model vortex configurations that mimic the symmetric and antisymmetric modes of the jet. We used these configurations as initial conditions for all of the simulations described in this section. The vortex configuration displayed in Figure 4.1(a) is derived from the well-known von Karman vortex street. Von Karman proved [84] that two infinitely long parallel rows of vortices will remain stable if two conditions are satisfied: (1) the strength of each vortex is identical, with vortices in the left column having opposite vorticity from those in the right column and (2) the spacing between vortices satisfies $a/b = 0.281$, where a and b are labelled in Figure 4.1(a). It is worth noting that the vortices in the antisymmetric mode of our planar jet (see Figure 3.4) are arranged in an antisymmetric configuration similar to the von



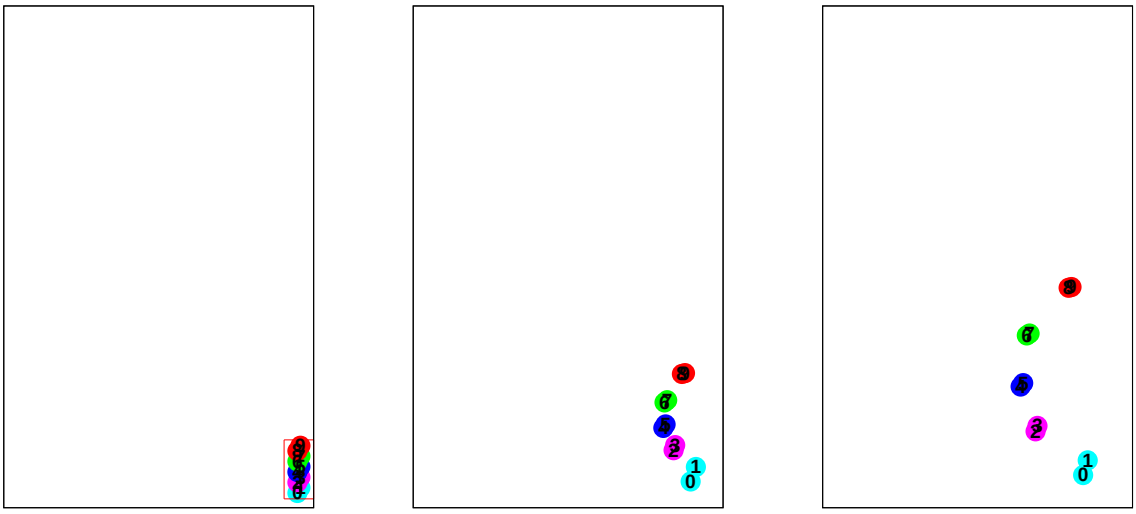
(a) (b)

Figure 4.1: Vortex configurations. Initial conditions in (a) are derived from the stability condition for a von Karman vortex street. Vortices are spaced 1 unit apart in the x-direction and $b = 1/0.281$ units apart in the y-direction. Similarly, an x-spacing of 1 and a y spacing of 3.6 are used to obtain the symmetric configuration displayed in (b). In both cases, the vortices in the left column have strength -1 (counter-clockwise rotation), and those in the right column have strength 1 (clockwise rotation).

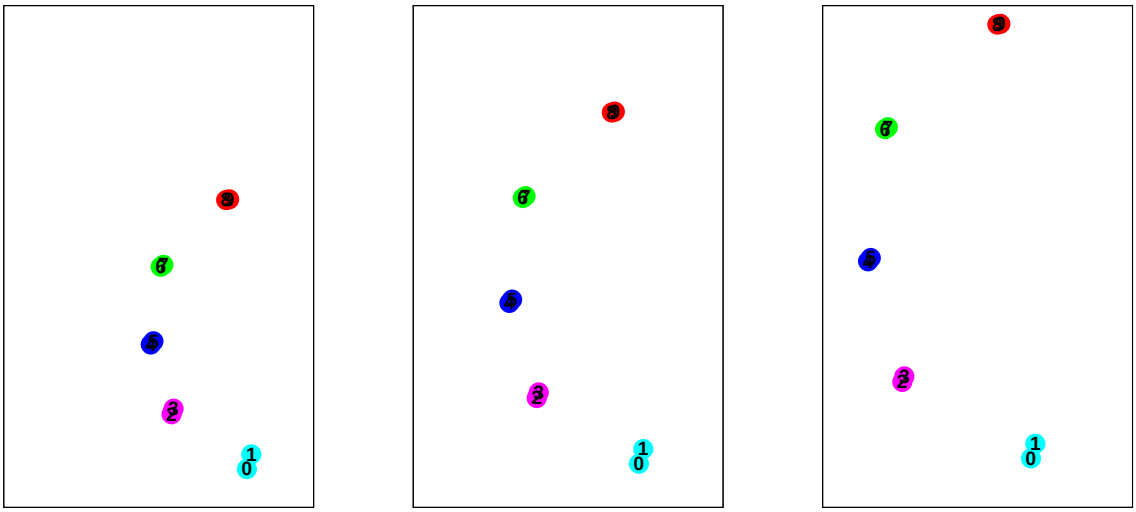
Karman street, but the measured spacing is approximately $a/b = 0.5$. Although the antisymmetric mode measured in the laboratory is slightly different from the von Karman configuration, it is still a worthwhile vortex pattern to study because it arises naturally in many physical systems. Clearly, in our numerical experiments, we cannot use an infinitely long vortex street, but even with a finite number of vortices, we expect the von Karman arrangement to result in relatively stable dynamics. In contrast, the symmetric pattern displayed in Figure 4.1(b) is highly unstable. We have also been able to realize the symmetric mode in our laboratory jet, but only briefly and after very careful tuning of the forcing. Thus, these two vortex configurations provide two very different contexts—both of which are physically realistic—in which to study data-assimilation methodologies. The images in Figures 4.2 and 4.3 display snapshots of the dynamical evolution of each system.[1]

Starting from these initial conditions, we conducted a comprehensive set of the imperfect model experiments described above. Recall that our approach was to use a high-resolution simulation as the "truth" and apply observations of the truth to correct a coarser-timestep simulation. Each simulation in our experiments was a fourth-order Runge-Kutta integration of the Biot-Savart equations from Section 2.1, starting from one of the initial conditions described in the previous paragraph. To produce the "truth" simulation, we ran the numerical integration with a very small timestep, chosen using a convergence test—i.e., starting with a fairly large timestep, say $1s$, and decreasing it until the differences between vortex trajectories in each successive simulation were less than 10^{-5}. This method yielded a timestep of $0.0001s$ for the symmetric intial conditions and a timestep of $0.1s$ for the von Karman experiments. Figures 4.2 and 4.3 display snapshots of the truth simulations for these experiments. The timesteps for the larger-timestep "model" simulations were chosen based on the resulting mean-squared

[1] Note that Figure 4.3 displays a much shorter simulation.



(0s) (200s) (400s)



(600s) (800s) (1000s)

Figure 4.2: Results of a 5400s simulation with a 0.1 second timestep starting from the von Karman initial condition. Figure 4.1(a) is a closeup of the square region in the lower right corner of (a). Images display instantaneous positions of the vortices at the times indicated below the figures.

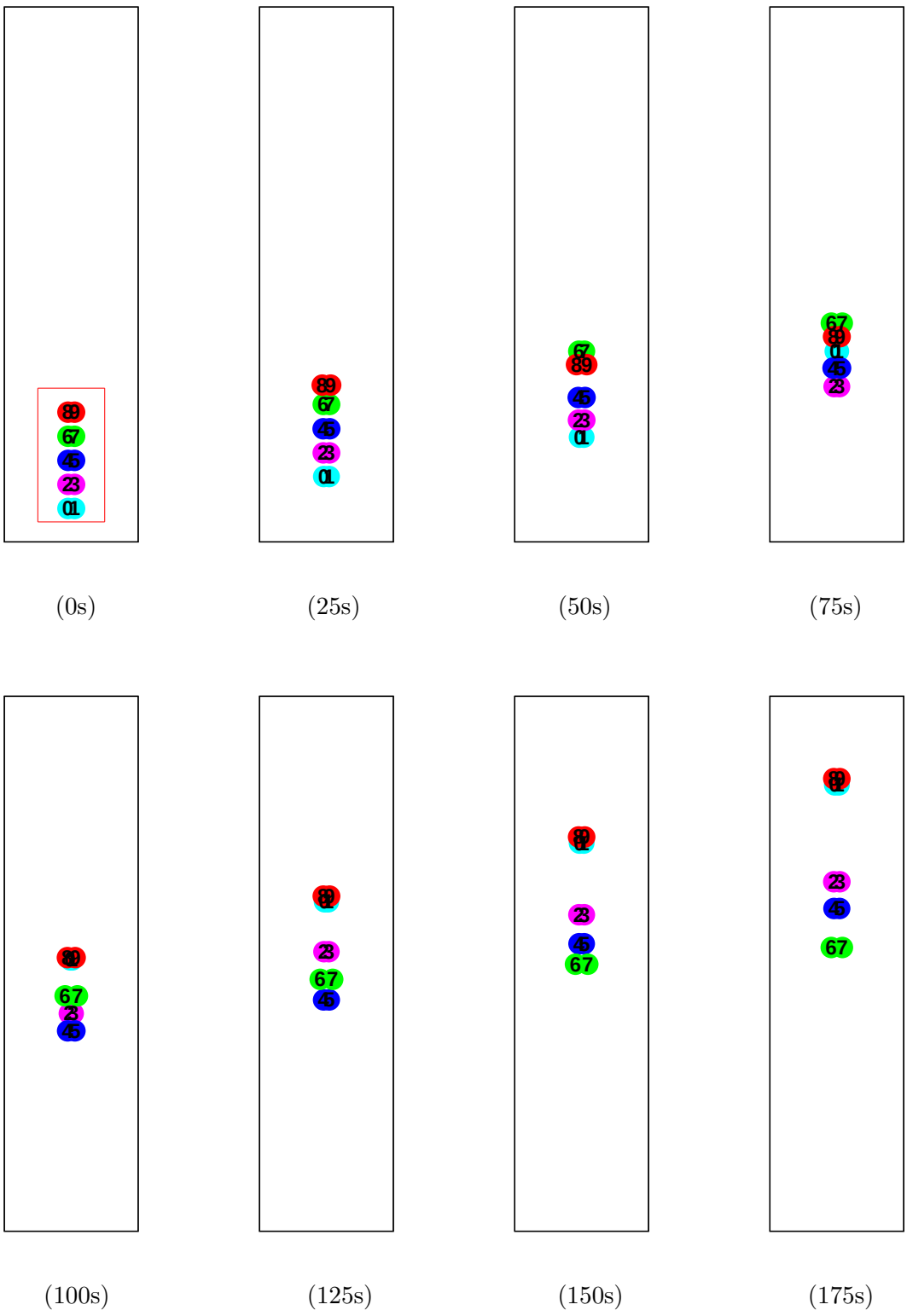


Figure 4.3: Results of a 200s simulation with a 0.0001 second timestep starting from the symmetric initial condition. Figure 4.1(b) is a closeup of the square region in (a). Images display instantaneous positions of the vortices at the times indicated below each figure. Note that this is a much shorter simulation than in Figure 4.2, as this configuration is far more unstable.

error (MSE) between the model and truth simulation[2] . We chose a timestep of $1s$ for the symmetric experiments, which resulted in a MSE difference of 84.9 length units. This value is roughly comparable to the distance that the vortices travel in the y-direction over the course of the $200s$ simulations depicted in Figure 4.5. Thus, on average, each vortex has an error equal to the square root of the total distance travelled. This choice is fairly arbitrary, but we have found that the errors introduced are significant enough to permit a useful exploration of the data-assimilation process. The simulation length and large timestep for the von Karman experiments were chosen to produce a similar value for the MSE between the truth and model simulation. A simulation length of $5400s$ and a timestep of $50s$ results in an MSE of 61.8 between the simulations depicted in Figures 4.4(a) and (b).

The final step in this preliminary evaluation of our data-assimilation strategies was to use the "truth" simulations of Figures 4.2(a) and 4.3(a) to correct the "model" ones in Figures 4.2(b) and 4.3(b). We first attempted a direct, continuous assimilation approach, simply replacing the simulated variables in the "model" run with the "true" values at various intervals. This is the standard "periodic correction" approach used in most of the data-assimilation research reviewed in Section 2.2. With this strategy, the appropriate correction interval must be chosen in advance, keeping in mind that more corrections will most likely result in a more-accurate simulation. We applied this technique to our numerical experiments as follows. The simulation displayed in Figure 4.5(b) is 200 seconds in length with a 1 second timestep, so we could examine correction intervals ranging from 1 second to 200 seconds. A simulation that is corrected at every timestep (every 1s), then, will be identical to the reference simulation displayed in Figure 4.5(a), whereas a simulation corrected every 25 seconds will only be corrected 8 times over the course of the 200 second model run. Similarly, for the von Karman

[2] The MSE was computed by summing over the errors for each vortex at each timestep and dividing by the number of vortices and the number of timesteps. More information on our MSE calculations can be found in Section 6.

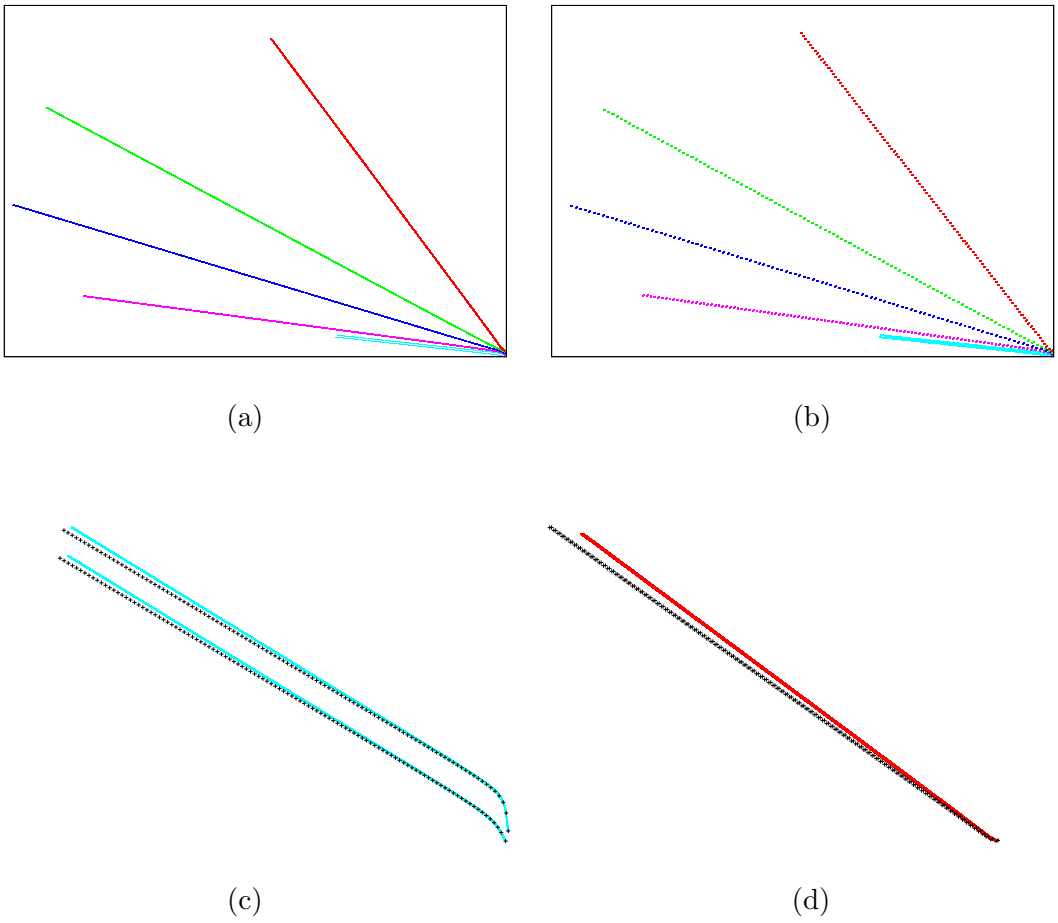
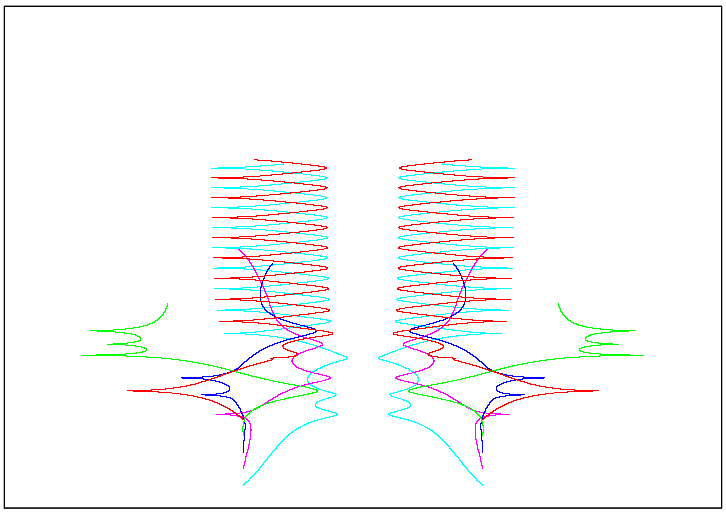
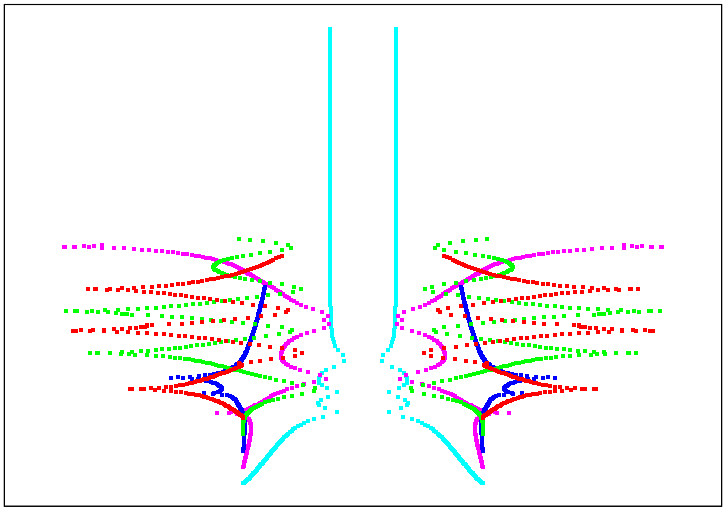


Figure 4.4: Full trajectories of (a) "truth" and (b) "model" simulations starting from the initial conditions in Figure 4.1(a). Part (a) shows a 5400 second simulation with a 0.1 second timestep and (b) shows a 5400 second simulation with a 50 second timestep. Note that there are 10 vortices in these simulations; each "arm" in (a) and (b) is actually a pair of vortices travelling very close together. In our numerical experiments, we use the more-accurate trajectories from (a) to correct the vortices in (b). It is difficult to see the differences between the vortex trajectories in (a) and (b) due to the large scale of the plots. In (c) and (d), we provide closeups of the lower and upper pairs of vortices, respectively. Here, the solid paths are taken from the "truth" simulation in (a) and the black $++++$ paths are the corresponding trajectories from (b). These lower plots show the subtle differences between these two simulations.



(a)



(b)

Figure 4.5: Full trajectories of (a) "truth" and (b) "model" simulations starting from initial conditions in Figure 4.1(b). Note that these plots are not to scale; we have zoomed in on the x-range to make it easier to see the interesting dynamics. Part (a) shows a 200 second simulation with a 0.0001 second timestep and (b) shows a 200 second simulation with a 1 second timestep. In our numerical experiments, we use the more-accurate trajectories from (a) to correct the vortices in (b).

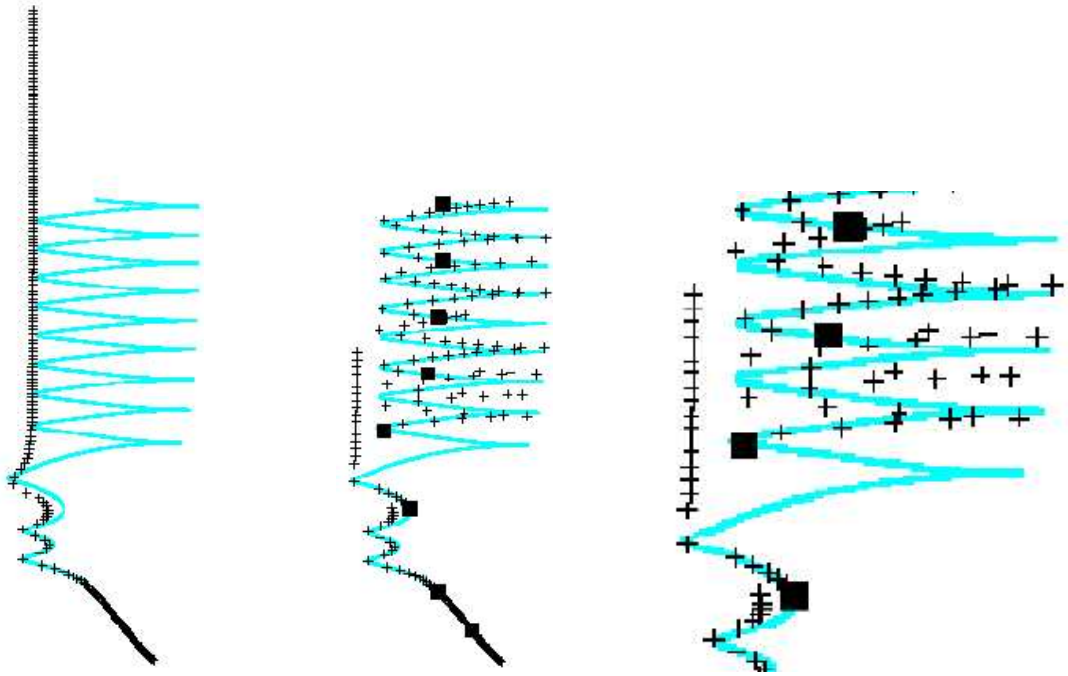
type simulation displayed in Figure 4.4(b), we can select any correction interval that is a multiple of the 50 second timestep. We use this standard periodic correction approach as a baseline against which to compare other techniques that use more-sophisticated correction timing.

Chapter 5

Dynamics-Informed Data Assimilation

Much data-assimilation research is devoted to improving the algorithms used to assimilate the observations. It is perhaps equally important to examine how much new information is provided by the observations that are assimilated. When the model is highly accurate, the relative information content in the observations is fairly low—i.e., the assimilated observations do not impart a significant change to our prior knowledge of the system. In contrast, when the model is failing to track the true dynamics, the observations can—potentially—drastically improve the simulation. If we can detect when the model might be diverging from reality, then, we can intelligently select when to correct it. This is a difficult task, as one does not know the true state of the system in practical data-assimilation applications. Fortunately, we do know that solver algorithms are based on interpolations and extrapolations, and they make mistakes in regions where the system derivative varies rapidly. A small numerical error in the calculation of a vortex position will be amplified if the landscape of the surrounding velocity field is "steep".

Figure 5.1(b) shows an example of this phenomenon. This image displays the "true" trajectory as a solid line and the trajectory from the coarser time step simulation as a series of + symbols. The latter is corrected to the former at the locations indicated by the black squares. In the middle section of (b), shown magnified in (c), notice how the model loses track of the true dynamics in a region where the forces acting on



(a) (b) (c)

Figure 5.1: Assimilating data into the point-vortex model: The numerical results of Figure 4.5(a) are used to correct the vortices in the simulation of Figure 4.5(b). The solid line and the $+++++$ path are the true and corrected trajectories, respectively; the data-assimilation scheme corrects the latter to the former at the points indicated by the black squares. (a) displays the results when no correction is applied to the $+++++$ path (b) displays the results of periodically correcting the simulation at 25s intervals and (c) is a closeup of the middle section of (b). The mean-squared error was 61.7 in (a) and 1.12 in (b) and (c).

the vortex are rapidly changing, as evidenced by its abrupt change in direction. An algorithm that chooses to apply the correction in this sensitive location could produce a simulation that tracks the "true" dynamics much more faithfully. Note that traditional data assimilation techniques disregard the system dynamics, although ensemble methods register dynamically sensitive regions via the ensemble spread. *Explicitly* using the dynamics to inform the correction timing is uncommon in the literature.

The analysis presented in the preceding paragraphs led us explore some new data assimilation schemes that are based on our hypothesis that the dynamical stability of the system should dictate the correction timing. We will refer to any technique in this class of methods as a *dynamics-informed assimilation algorithm*. One way to evaluate the dynamical stability of the system is to analyze the spatial gradients of the induced velocity at each vortex position. We evaluated six targeted observing strategies—described in Section 5.1 below—that each apply a different metric to these velocity gradients in order to make a correction decision. A seventh approach makes use of the Runge-Kutta test steps to determine if the system is in a region where error growth is likely; this technique is described in Section 5.2.

For each of these techniques, the decision criterion is applied separately to each vortex. So, at a given timestep, any number of the vortices can be in a "dangerous region" where the associated dynamical condition is satisfied. We have tried correcting only the identified vortices as well as correcting all vortices, and we found, not surprisingly, that correcting all of the vortices resulted in more-accurate simulations. So, correcting all vortices is the strategy used in the experiments presented in this thesis. Of course, there is an associated computational load here, and correcting fewer vortices would be preferable. In our future work, we will explore techniques for achieving accurate simulations when correcting only a subset of the vortices.

Each of the dynamics-informed techniques outlined below requires the specification of a threshold T_c that is used in the correction decision. In general, choosing the

correct value of T_c for a particular data assimilation experiment requires some domain knowledge and hand tuning to achieve a desired correction frequency. For our purposes, we are exploring the performance of the method, so we varied T_c to analyze a wide range of dynamical correction timings. Recall that the value chosen for T_c dictates how many corrections are applied to the simulation. So, we can effectively compare the performance of any dynamics-informed method to that of periodic correction by evaluating the success of each method *when the same number of corrections is performed*. In our future work, we will explore a more-quantitative approach to selecting the threshold T_c, and possibly implement an adaptive technique.

5.1 Gradient-Based Methods

For all of the methods that rely on a measure of the induced velocity gradients, we performed the same velocity gradient computations on each timestep. Our approach was as follows: we computed the components of the Jacobian of the induced velocity field at the location of each vortex using divided differences—that is,

$$J = \begin{pmatrix} \frac{\partial U}{\partial x} & \frac{\partial U}{\partial y} \\ \\ \frac{\partial V}{\partial x} & \frac{\partial V}{\partial y} \end{pmatrix}$$

where U and V are the induced velocities in the x and y directions, respectively. The divided-difference method calculates $\frac{dU}{dx}$ as follows:

$$\frac{dU}{dx} = \frac{U(x+h) - U(x-h)}{2h} ,$$

where we chose h using a convergence test[1] for each set of initial conditions studied. For the von Karman conditions, we used $h = 1e^{-6}$ and for the symmetric case, we used $h = 1e^{-7}$. The other components of the Jacobian were computed similarly.

The idea behind gradient-based assimilation is to correct the vortices only when

[1] We started with $h = 1$ and iteratively set $h = h/10$ until the MSE difference in induced velocities from one iteration to the next was less than 10^{-5}.

the induced velocity gradients indicate that the system is in a dynamically sensitive region. We explored several methods for making this determination:

- **Norm of Jacobian above a threshold.** Here, we compute the norm of the Jacobian and correct the system if the norm exceeds a pre-determined threshold, T_c. We chose to use the L_1 norm for the analysis in this thesis; a comparison of the L_1, L_2, and L_∞ norms for our test cases showed very little variation, so any choice of norm should provide roughly the same results (although computation of the L_∞ norm can be numerically ill-conditioned). Recall that the L_1 norm calculates the maximum column sum of a matrix, i.e.,

$$||J|| = \max\left\{\left|\frac{dU}{dx}\right| + \left|\frac{dV}{dx}\right|, \left|\frac{dU}{dy}\right| + \left|\frac{dV}{dy}\right|\right\}$$

- **Change in Norm of Jacobian.** In this class of methods, we compare the L_1 norm of the Jacobian at the current time step, $||J||_t$, to the norm computed on the previous timestep, $||J||_{t-1}$. We tried three different correction schemes using these two values: correcting when

  (1) $||J||_t$ was a certain percentage larger than $||J||_{t-1}$

  (2) $||J||_{t-1}$ was a certain percentage larger than $||J||_t$

  (3) $||J||$ either increased or decreased by the thresold percentage.

  The mathematics for each decision criterion are:

$$\begin{aligned}
(1) \quad & \frac{||J||_t - ||J||_{t-1}}{||J||_{t-1}} > T_c \\
(2) \quad & \frac{||J||_{t-1} - ||J||_t}{||J||_t} > T_c \\
(3) \quad & \frac{|\,||J||_t - ||J||_{t-1}\,|}{||J||_{t-1}} > T_c
\end{aligned}$$

- **Positive eigenvalues above a threshold.** The stability of a fixed point or equilibrium point in a dynamical system can be determined by computing the

eigenvalues of the Jacobian matrix. If the eigenvalues are positive, the system dynamics at the point are unstable. We use this same approach to analyze each vortex location to determine if the local dynamics are unstable. If the eigenvalues of the Jacobian for any vortex exceed a pre-determined positive threshold T_c, we correct the system.

- **Shear rate.** Shear rate is a measure of the rate of change of velocity of one layer of fluid passing over an adjacent layer. This is another measure of the local velocity gradients in the fluid at the location of each point vortex. It is calculated as follows:

$$\dot{\gamma}_{xy} = \frac{\partial U}{\partial y} + \frac{\partial V}{\partial x}$$

where U and V are the induced velocities in the x and y directions respectively. In this method, we apply the correction when $\dot{\gamma}_{xy}$ exceeds a pre-determined threshold T_c.

5.2 Runge-Kutta Test Step Method

All of the gradient-based methods described above incur a significant computational cost in computing the Jacobian of the induced velocity gradients at each vortex position. It would be preferable to find a method that uses some information that is more readily available in the simulation. Conveniently, we can glean some dynamical information from the numerical integration process. The fourth-order Runge-Kutta algorithm that we use to integrate the point-vortex Biot-Savart equations takes four "test steps", computing the induced velocities at each test location. These test steps can be co-opted to provide a picture of how dynamically sensitive the point-vortex system is at the current time in the simulation. Specifically, if the induced velocities at the test locations (which are "nearby" in time and space) differ significantly from each other, we can conclude that the induced velocity gradients must be large. Stated differently,

the higher order derivatives in the Runge-Kutta expansion are too large to ignore. This indicates that error growth is likely, because the model integration will be less accurate than the dynamics. We use this information as a trigger for correcting the model by computing a mean-squared error measure on the test steps as follows. Let k_1, k_2, k_3, k_4 denote the four Runge-Kutta test steps. In our application, these vectors represent induced velocity gradients at each vortex position. For each vortex, i, we find the minimum and maximum induced velocity values and compute the squared difference between them. We sum these values and divide by the number of vortices to obtain the MSE:

$$MSE = \left(\sum_{i=1}^{n} \max\{k_1[i], k_2[i], k_3[i], k_4[i]\} - \min\{k_1[i], k_2[i], k_3[i], k_4[i]\} \right) / n,$$

where n is the number of vortices in the simulation. If the MSE exceeds the threshold T_c, we correct the model.

This technique of comparing solver test steps is not as generalizable as the gradient-based methods described in the previous section because it only applies to simulations that use a particular type of solver, namely Runge-Kutta. Clearly, the technique would not work with Euler methods because only one system derivative is considered in advancing the model at each timestep. However, comparing solver steps might work well for other types of solvers that use more than one data point to move the simulation forward in time. For example, multistep methods keep track of more past values in computing the next value of the system. Comparing the MSE between values in the recent past might provide some information about the local dynamics, as long as the timestep is small enough. However, this would require storage of past values, increasing the memory requirements for the assimilation. An evaluation of whether or not this correction technique will work well with other numerical solvers is outside the scope of this thesis, but it will be interesting to explore in our future work.

Chapter 6

Evaluation and Comparison of Assimilation Methods

When evaluating new data-assimilation methods, one can use various metrics for success. Perhaps the two most important are the accuracy of the simulation and the computational cost of the algorithms. In order to measure the accuracy of the simulation, we considered three different error metrics. The simplest is the mean-squared error between the reference simulation and the data-assimilation model run. That is, we computed the square of the distance between the true and corrected positions of each vortex at each timestep, summed the results, and divided by the length of the simulation and the number of vortices. This mean-squared-error measure captures our visual interpretation of the accuracy of the simulation in Figure 5.1, where our eyes register the accrued differences between the $++++$ path and the solid line. A second method that we evaluated measures how often the simulated vortex positions are close to their respective reference positions—that is, how often the simulated position is within an ϵ-disc around its "true" location. To compute this, we simply summed the number of timesteps in which the vortices satisfied this criterion and took the mean of this count over simulation length and number of vortices. For simulations in which some small amount of error is acceptable, the user may wish to know how often the model is within the desired distance from the truth, and this metric captures that information. A third error metric measures how long the correction keeps the vortex "on-track." For each vortex, we counted how long the simulated vortex position remains within an ϵ-disc

around the associated "true" vortex position after each correction. Note that another correction could be applied during this measurement. The final error measure was also normalized over the number of vortices and the number of corrections applied during the simulation. Note that this is different from the previous metric in that it specifically measures how long *after a correction* the simulation remains within an acceptable error tolerance. This provides some information about the sensitivity of the system. If the simulation is quickly going astray after each correction, the underlying dynamics are clearly very sensitive to small numerical errors. Note that the second and third error metrics require a choice of ϵ, which may be fairly arbitrary and/or application-specific. Thus, in this thesis, we will primarily use the mean-squared-error metric to compare the accuracy achieved by various data-assimilation methods.

When considering the computational cost of a data-assimilation technique, there are several factors to take into account, including the speed of the model, the difficulty in gathering the observations, and the complexity of the assimilation algorithm. If the observations can be gathered easily, it may make sense to compare techniques based on the complexity of the assimilation algorithm itself. If, on the other hand, the cost of gathering and assimilating the observations is prohibitive, then the number of corrections performed is a good measure of the computational expense of the correction methodology. This is the case for data assimilation in the context of a point-vortex model: the computational complexity of the model itself is low and processing the velocity field observations to extract vortices is relatively expensive. As a result, in devising our dynamics-informed approach, we chose to increase the computational complexity of the model (by tracking velocity gradients) in favor of limiting the number of corrections performed. Since we are working in a context in which gathering, processing, and assimilating observations is very costly, we have chosen to use the number of corrections as an appropriate measure of the computational cost.

To investigate the performance of a given data-assimilation technique, we con-

ducted an ensemble of experiments with the data sets depicted in Figures 4.2 and 4.3. Each simulation was evaluated using the mean-squared error (MSE) metric described in this section. By plotting the MSE results as a function of the number of corrections performed, we were able to investigate the accuracy of the assimilation technique as a function of its computational cost. This also provided a simple mechanism to compare the performance of two different methods, such as a periodic versus a dynamics-informed correction.

When evaluating the periodic correction approach, we investigated all possible correction intervals for our test systems. These correction intervals were determined by the simulation length and time step. For the von Karman configuration, the simulation length was $5400s$ with a time step of $50s$ seconds, so we investigated assimilation intervals of $50s, 100s, 150s, \ldots, 5450s$. Note that the $50s$ correction period results in 0 MSE, while the $5450s$ produces an uncorrected simulation and provides a baseline error measure. For the symmetric experiments, the simulation length was $200s$ and the time step was $1s$, permitting periodic correction intervals of $1s, 2s, 3s, \ldots 201s$. Here again, the $201s$ period provides the MSE for an uncorrected simulation.

For each dynamics-informed approach, we ran simulations with a wide range of threshold values T_c. By recording the number of corrections performed for each value of T_c, we obtained MSE as a function of the number of corrections. This allowed us to effectively compare dynamics-informed correction to periodic correction. The range of values used for T_c varied depending on the method being used. Table 6.1 shows the values used for each type of dynamics-informed experiment. These threshold ranges were obtained by hand-tuning to find the values that would produce a fairly complete MSE curve, enabling us to compare the techniques for varying numbers of corrections applied. In the next section, we present the results of these MSE calculations for each of the dynamics-informed strategies described in Section 5, and compare these techniques to periodic correction. The initial comparisons are performed without adding noise to

Method	**Symmetric $\mathbf{T_c}$ [Min,Max]:Inc**	**von Karman $\mathbf{T_c}$ [Min,Max]:Inc**
Norm Change	[0, 200]:0.5	[0, 5]:0.01
Norm Increase	[0, 200]:0.5	[0, 5]:0.01
Norm Decrease	[0, 200]:0.5	[0, 5]:0.01
Norm Thresh	[0.1, 7]:0.01	[0, 0.25]:0.001
Eigenvalues	[0.3, 12.9]:0.02	[0, 0.25]:0.001
Shear	[0.2, 10]:0.01	[0.065, 1]:0.001
RK Steps	[0, 0.038]:0.00005	[0, 0.1]:0.0001

Table 6.1: Thresholds for dynamics-informed techniques. Each entry in this table describes how thresholds were varied for a particular dynamics-informed assimilation experiment. The dynamics-informed technique is listed in the left column and the initial conditions are along the top row. The format of each entry is [minimum Tc, maximum T_c]:T_c increment.

the observations used in the assimilation. Noise-added experiments appear in Section 8.

Chapter 7

Noise-Free Results for von Karman and Symmetric

To study and compare the various dynamics-informed correction techniques presented in Section 5, we performed a comprehensive series of experiments using the data sets from Figures 4.2 and 4.3 . Figure 7.1 displays the results for the von Karman vortex configuration. The plot in the upper left corner displays the MSE results for *all* of the different dynamics-informed and periodic correction techniques. Since this is a little cluttered, we have also plotted a single comparison plot for each dynamics informed correction technique versus periodic correction. Note that each data point in a given plot represents an MSE calculation for a single point-vortex simulation.

There are several interesting results here. First, notice that there are basically two patterns in these plots. Plots (c), (d), (e) and (h) are very similar to each other, as are plots (b), (f), and (g). In the first set of plots—which display the results of the norm increase, norm decrease, norm change, and Runge-Kutta test step methods from Section 5—each dynamics-informed method significantly outperforms periodic correction by several orders of magnitude. This is a significant result: it suggests that we can greatly improve the data-assimilation accuracy by incorporating one of these targeted observing strategies. Since the MSE results for these four techniques are almost identical, the Runge-Kutta test step method is the clear winner because it does not require computation of the velocity gradients.

It is curious that these four methods produce almost identical results. To un-

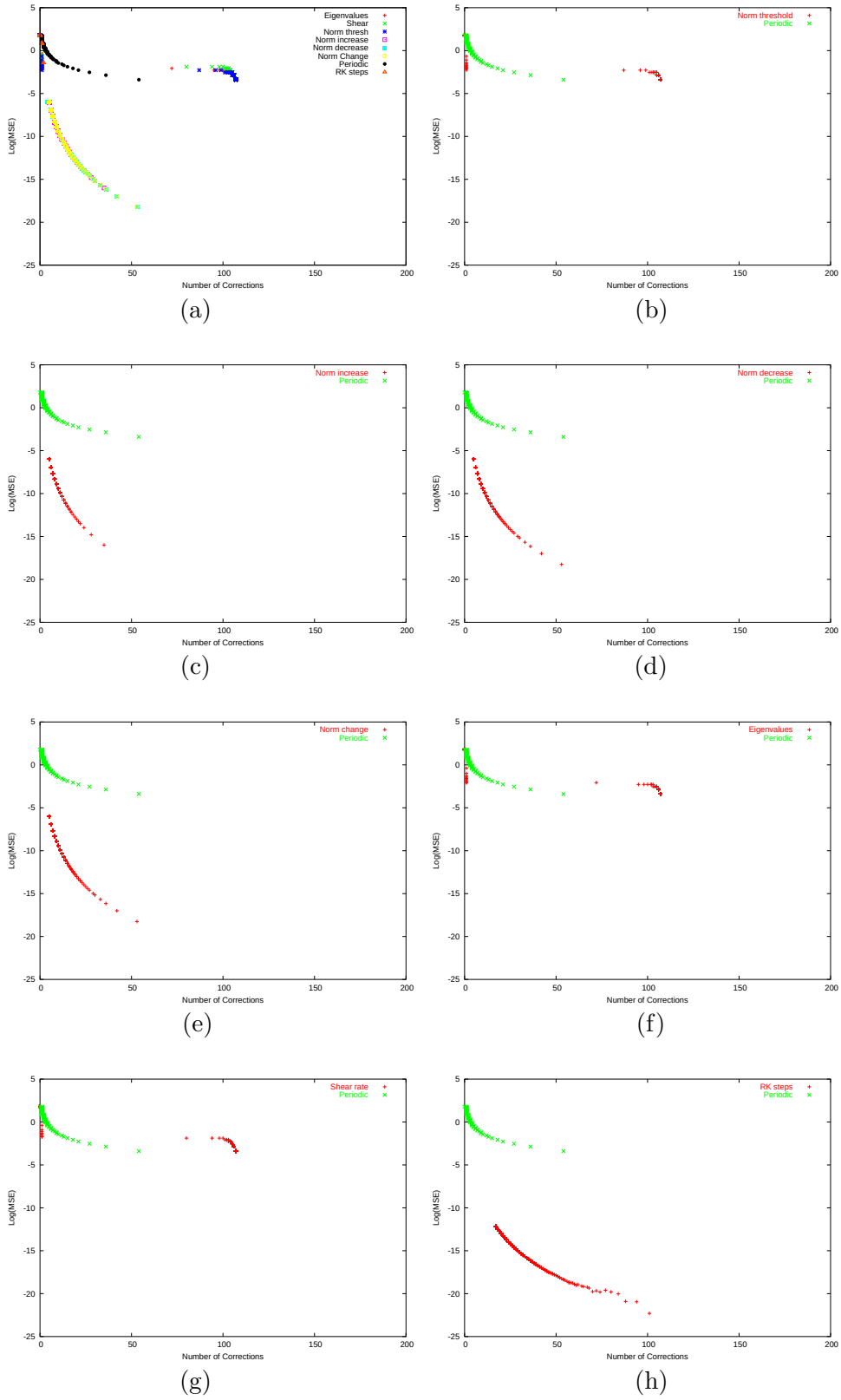


Figure 7.1: Comparison of dynamics-informed and periodic assimilation using the initial conditions in Figure 4.1(a).

derstand this finding, we plotted a time series of the norms of the induced velocity Jacobian for vortices 1, 4, 7, and 10 in a representative simulation (see Figure 7.2). While evaluating these plots, it is useful to recall the behavior of the von Karman simulation, as depicted in Figure 4.2. Initially, the 10 vortices are very close together and induced velocities are changing rapidly. As the vortices move farther apart, they pair up. Eventually, the induced velocity at a particular vortex is dominated by the influence of a single partner vortex. This pattern also clearly manifests itself in the norm of the Jacobian. In the time series plots in Figure 7.2, we see that the gradients are changing rapidly toward the start of the simulation, but then they level out (corresponding to the very stable pair dynamics just described). In the first several timesteps, the Jacobian norm is increasing for some of the vortices and decreasing for others. This explains why the targeted observing techniques that are based on a percentage increase, decrease, or change from one timestep to the next all perform similarly. Each of these techniques applies all of the corrections in the first several time steps and does not correct the simulation when the norms become flat. It is not as obvious why the Runge-Kutta test step method is identical to the other three, but there is a commonality between all four of the methods. They all look at *rates of change*, considering how the system changes from one time step to the next. The test steps in the Runge-Kutta method are moving the system forward by either a half time step or a full time step and measuring the induced velocities at those times. Thus, computing an MSE between these test steps is similar to looking at a percent change in the norm of the Jacobian from one time step to the next.

Plots (b), (f), and (g) in Figure 7.1 display the results of the norm, eigenvalue, and shear rate thresholding techniques. Recall from Section 5 that these techniques essentially perform different arithmetic operations on the velocity gradients at each vortex position. It appears that these different metrics are providing the same information for the von Karman data set. They each provide a measure of how "unstable" the dynamics

are. Surprisingly, for higher numbers of corrections, these techniques do not outperform periodic correction. To understand this, we can again refer to the plots of the norms displayed in Figure 7.2. Consider, for example, the norm thresholding technique. Initially, as we decrease the threshold, we are correcting the simulation at the first several timesteps—where the action is. Once we decrease the threshold value too far, however, the technique subsumes the flat regions in these norm plots. The result is that we apply too many corrections. It is interesting and somewhat counter-intuitive that these static measures of the velocity gradients do not work well and that they are significantly worse than the methods that include a time component in the dynamical analysis. This seems to suggest that, for the von Karman data set, changes in the dynamical stability are more important than instantaneous stability for point-vortex modelling accuracy.

It is interesting to compare these von Karman results to those achieved with the symmetric vortex data sets from Figure 4.3. The MSE results for the symmetric case are presented in Figure 7.3. For correction counts above roughly 25 corrections (1/4 of the time steps in the simulation), six of the dynamics-informed techniqes clearly outperform periodic correction, often by several orders of magnitude. It is encouraging the see the majority of our targeted observing techniques working as designed for this dynamically sentitive vortex configuration. However, there is no single technique that clearly works best. For a few corrections—less than about 30—the strategy of correcting based on decreasing norms outperforms all of the other techniques. The Runge-Kutta test step method is the winner for 30-100 corrections,however, and the norm change technique dominates for > 100 corrections. These are three of the techniques that also performed very well in the von Karman simulations, so we again see the trend that the *rate of change in the dynamical stability is important for simulation accuracy*. It is a bit disconcerting, however, that we cannot point to a single method that works best for all numbers of corrections. At least for both of the initial conditions studied in this section, correcting based on changing Jacobian norms or differences between Runge-Kutta test

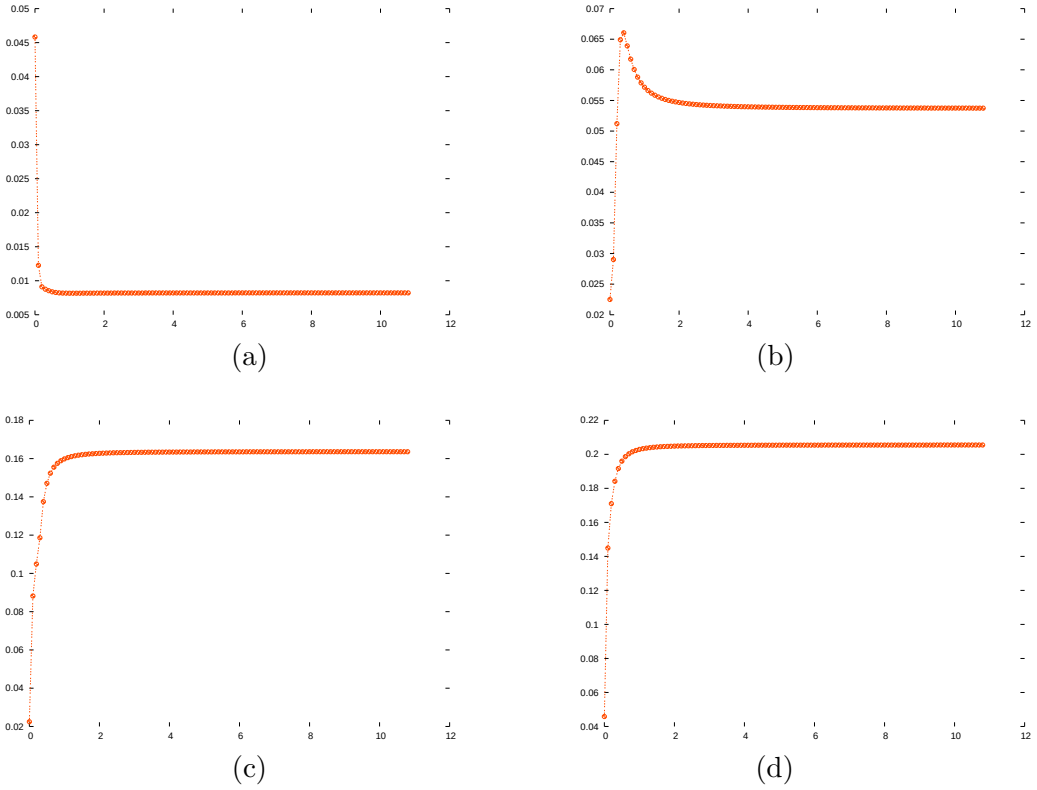


Figure 7.2: Velocity gradient norms as a function of time for vortex (a) 1 (b) 4 (c) 7 (d) 10 for a simulation starting from the initial conditions in Figure 4.1(a).

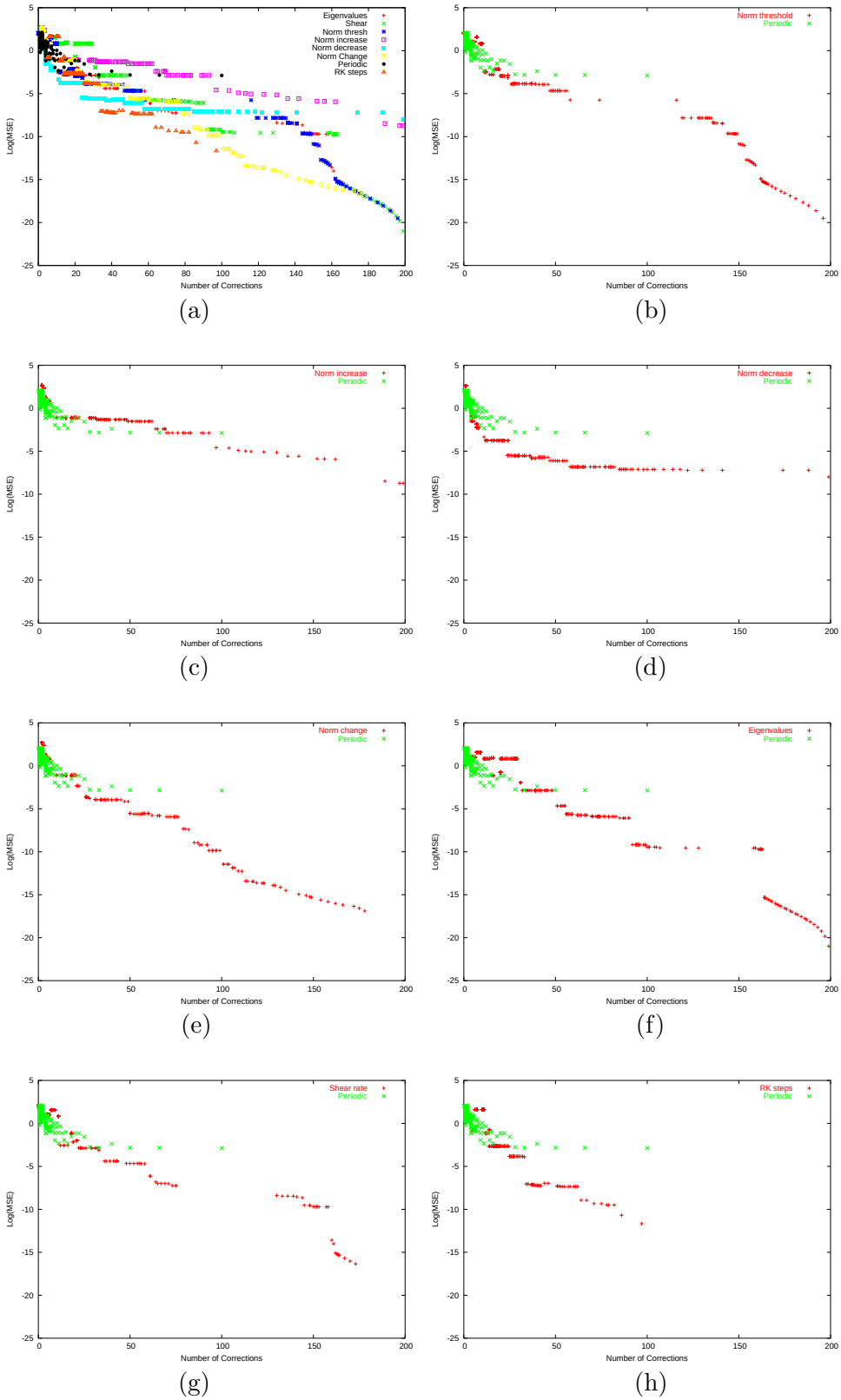


Figure 7.3: Comparison of dynamics-informed and periodic assimilation using the initial conditions in Figure 4.1(b).

steps appears to be a safe bet for improved performance of the assimilation. And, since the dynamical information required by the Runge-Kutta test step method is readily available, there is no reason not to use this method over periodic correction.

In comparing the symmetric results to those for the von Karman case, we again see the pattern that the norm, eigenvalue, and shear rate thresholding techniques (plots (b), (f), and (g), respectively) are qualitatively similar. This corroborates our conjecture that these metrics are providing roughly the same static dynamic stability information. However, the results for the other four cases are different from the von Karman simulations. The norm change and Runge-Kutta time step techniques have a similar trend, so these two seem to be measuring the same rate of change in the dynamical sensitivity of the system. However, correcting based on increasing norms does not outperform periodic correction for the symmetric simulations. To understand this, we can again look at time series of the norms for several vortices in a representative simulation (see Figure 7.4). The norms for the symmetric configuration look quite different than the von Karman case. They are characterized by relatively constant norms that are punctuated with fairly large spikes. These spikes might help to explain why the norm decrease method works better than the norm increase method. Theoretically, the tip of each spike is a dynamically senstive region where error growth is likely. Applying the correction at this timestep (which will occur when we use the norm increase method) will make the model very accurate at this timestep. However, the location is still in a region of high gradients. So, when we integrate forward in time, some errors are inevitably introduced. In the timesteps that follow, the norms are decreasing, so the norm increase method does not trigger a correction here. This analysis suggests that it may be more important to apply the corrections "after" the vortices pass through a high gradient region. This is exactly what the norm decrease method accomplishes for this system that is characterized by spikes in the norms, so its performance dominates the norm increase method.

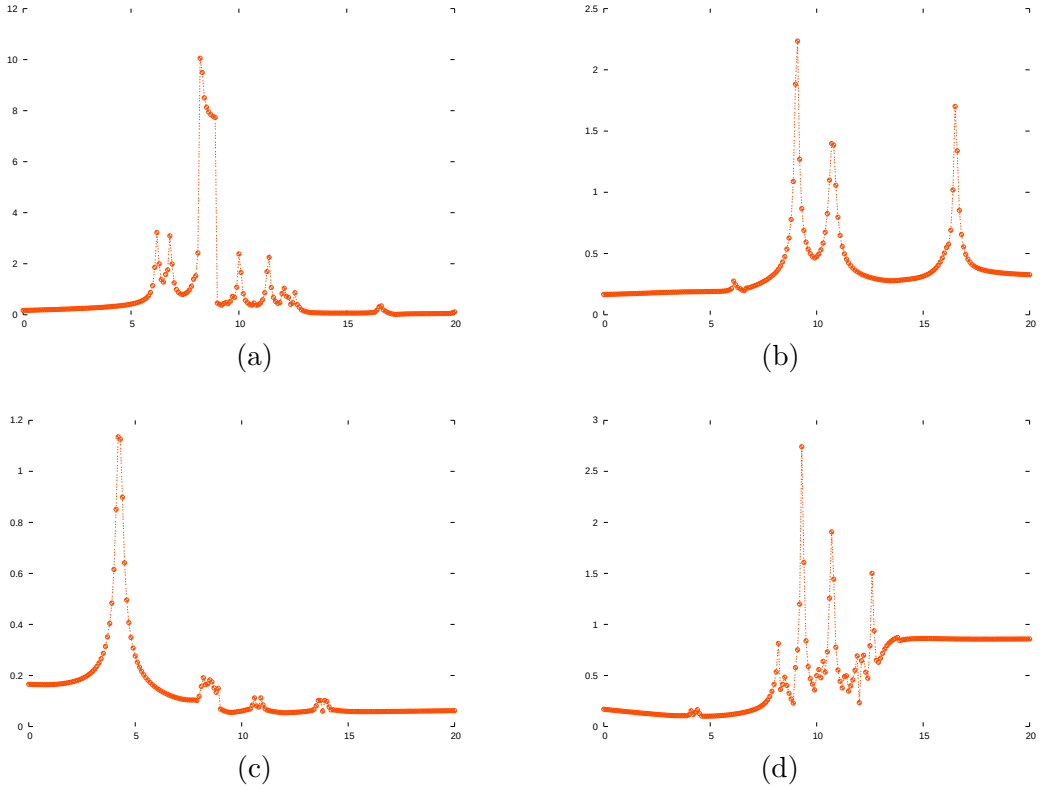


Figure 7.4: Velocity gradient Jacobian norms as a function of time for vortex (a) 1 (b) 4 (c) 7 (d) 10 for a simulation starting from the initial conditions in Figure 4.1(b).

Chapter 8

Adding Observational Noise to von Karman and Symmetric

All of the numerical experiments described thus far were performed without adding any noise to the simulated observations. This is a useful first test, but observations gathered from any real-world experiment will obviously be contaminated with noise. Our correction strategy must accomodate this challenge. To explore this issue, we performed several numerical experiments in which Gaussian-distributed random noise was added to the observations on each assimilation cycle. Figure 8.1 displays the results for the von Karman simulations. Each plot provides the MSE results for an ensemble of experiments corrected with noisy observations with a specific standard deviation, σ, for the additive noise. For example, each point in Figure 8.1(a) represents a simulation in which zero-mean Gaussian noise with $\sigma = 0.001$ was added to the x and y coordinates of the assimilated observations. There are five curves in each plot—the curve outlined by the purple squares displays the results for periodic correction experiments and the other four curves provide the results for the targeted observing techniques that worked well in our noise free experiments (red +=norm increase, green x=norm decrease, blue star=norm change, turquoise square=Runge Kutta test steps). A surprising and discouraging result of this exploration is that any level of noise seems to destroy the benefits of dynamics-informed assimilation for this data set. Periodic correction fares better for low levels of noise ($\sigma = 0.001, 0.01, 0.1$), but as we increase the noise ($\sigma = 0.5$), the MSE values obtained by any of the correction techniques are higher than that of a completely

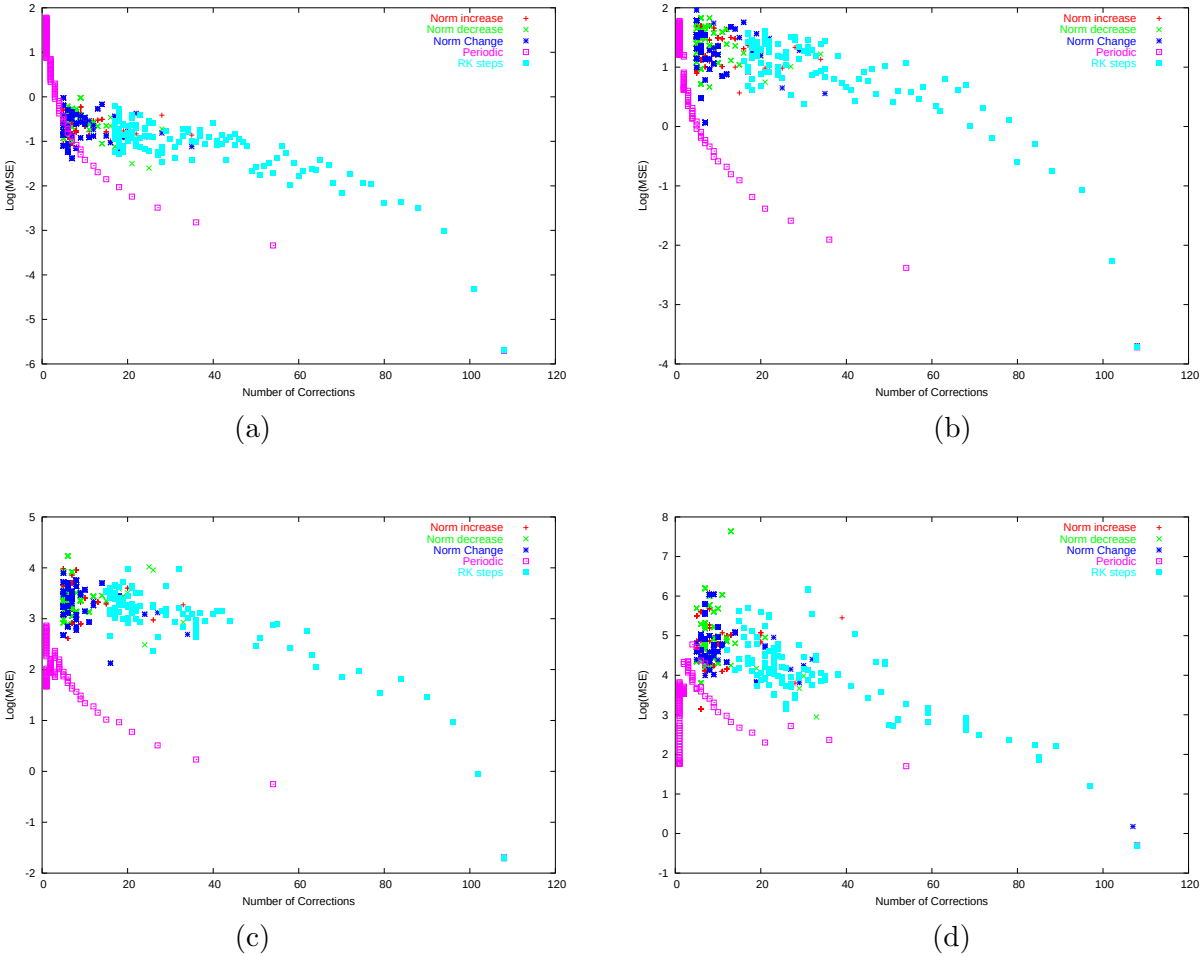


Figure 8.1: Effects of observational noise: simulations starting from the von Karman initial conditions from Figure 4.1(a). Each point in these plots shows the MSE when a single simulation is corrected with noisy observations; the goal is to compare periodic and dynamics-informed correction when the same amount of noise is added. The plots display the MSE results for the four targeted correction techniques that performed best in the first round of experiments: the norm increase, norm decrease, norm change, and Runge-Kutta test step algorithms. The standard deviation of the zero-mean Gaussian noise added to the observations was (a) 0.001 (b) 0.01 (c) 0.1 (f) 0.5

uncorrected simulation (which has log MSE$\approx$ 1.8). This suggests that we will need to employ a more-sophisticated assimilation algorithm to account for the noise in the observations. We explore two such algorithms in Sections 10 and 11.

To understand why the noise is so detrimental for the dynamics-informed techniques, we can look at the corrected paths for some of the vortices in the simulation (see Figure 8.2). Recall from our earlier discussion about the behavior of the vortices in this simulation that all ten vortices are initially close together; after the first few timesteps, they separate into pairs that travel off in different directions. As the simulation continues, the pairs become far enough apart that the motion of a given vortex is dominated by the velocity induced by its partner vortex. As shown in Figure 7.2, this behavior is reflected in the norms, which either increase or decrease rapidly to a steady-state value. As a consequence, the dynamics-informed methods that are based on changes in the norm apply all of their corrections in the first several timesteps, and they will not trigger a correction once the pair dynamics begin to dominate. If the last correction applied contains noise, the subsequent evolution of the vortex trajectories in the simulation will be slightly off track from the true paths. This can be seen in Figure 8.2—the colored paths outlined by the solid dots display the "true" behavior of the system. It is difficult to see in the figure, but each of the five "arms" of the fan shape in the figure actually depicts the paths of two vortices that have paired up. The paths outlined by the pluses in the figure display the vortex trajectories from an imperfect model simulation that is corrected whenever the Jacobian norms change by 3% or more from one timestep to the next. The black squares indicate the times where a correction has been applied. In this simulation, six corrections are applied at the first six timesteps. Since the last correction applied is noisy, we can see that the simulated trajectories diverge from the truth as time goes on; this is most obvious for the green trajectories in the figure. For this data set, it seems that we need to use a more-sophisticated assimilation algorithm that takes observational error into account when applying the correction. In Sections

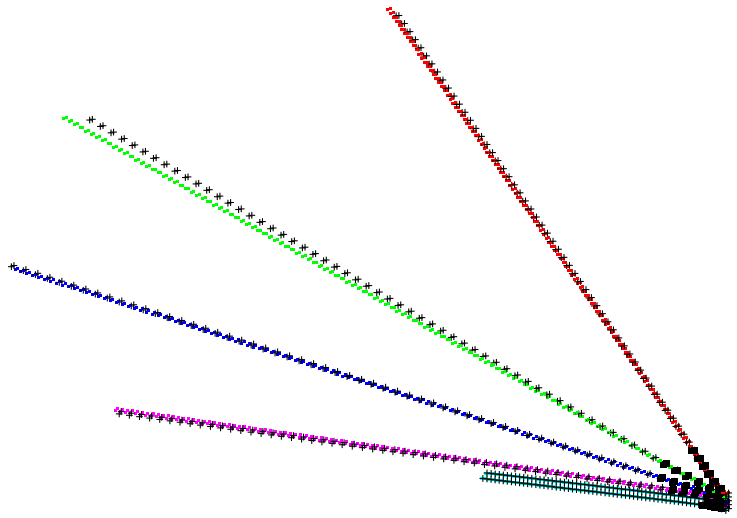


Figure 8.2: Noise destroys dynamics-informed correction: simulation starting from the von Karman initial conditions from Figure 4.1(a) and corrected based on the norm change technique with $T_c = 0.03$. The paths outlined by solid dots are the "true" trajectories for the 10 vortices in the simulation; the plus paths are the corrected trajectories. Corrections are applied at the first 6 timesteps at the positions indicated by the black squares. Corrections are not applied as vortices fly off in pairs, so the noise accumulates.

10 and 11, we present the results of using Newtonian nudging and ensemble filtering to perform this assimilation.

In contrast to the von Karman simulations, data assimilation improves most of the simulations starting from the initial conditions from Figure 4.1(b)[1] . These results are presented in Figure 8.3. Also, the impact of noise on the dynamics-informed techniques is much less drastic. For the lowest level of noise studied ($\sigma = 0.001$), the dynamics-informed techniques outperform periodic correction for correction counts above about 20-30. And, for $\sigma = 0.01$, dynamics-informed correction is at least competitive with periodic correction. It is encouraging that, for this data set, the dynamics-informed techniques are robust in the face of low levels of noise. However, once again we see that periodic correction is the winner for higher levels of noise ($\sigma = 0.1, 0.5$).

It is slightly counter-intuitive that the impact of noise is more deterimental to the von Karman simulations than the symmetric ones, since we might expect noise to have more of an impact in a dynamically sensitive flow. However, based on the noise analysis presented above for the von Karman data set, it is clear that the dynamical stability of the von Karman configuration causes problems in a direct assimilation of noisy data: because the dynamical algorithms do not update the system frequently enough, the noise accumulates. In contrast, the dynamical *sensitivity* of the symmetric case results in more frequent correction by all of the dynamics-informed techniques. And, more frequent correction is necessary in the face of noisy data. In the following sections, we analyze some potential explanations for—and possible improvements upon—the mediocre performance of the dynamics-informed correction techniques in the face of noise.

[1] Note that the log MSE for an uncorrected simulation is roughly 1.9 for this data set.

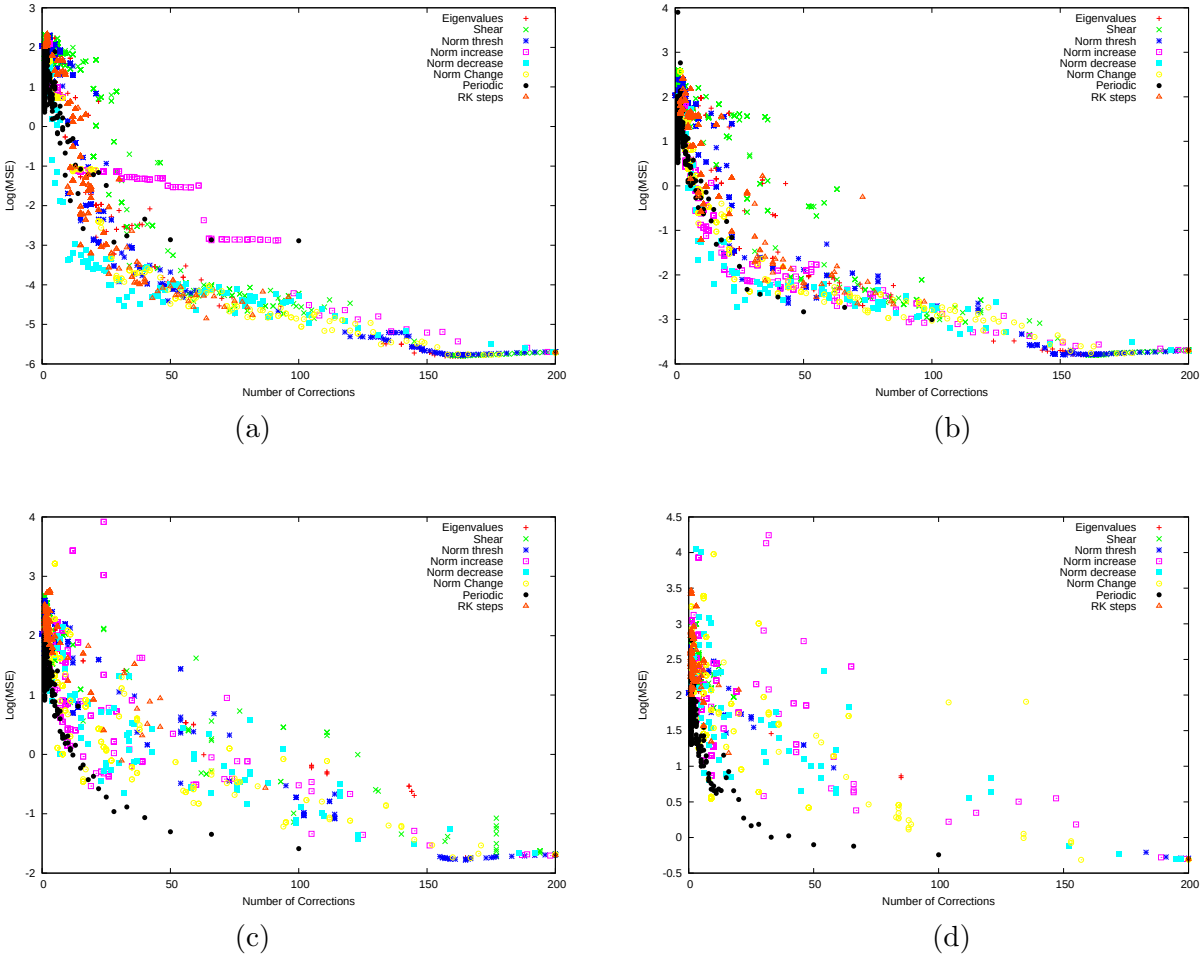


Figure 8.3: Effects of observational noise: simulations starting from the symmetric initial conditions from Figure 4.1(b). Each point in these plots shows the MSE when a single simulation is corrected with noisy observations; the goal is to compare periodic and dynamics-informed correction when the same amount of noise is added. The plots display the MSE results for the four targeted correction techniques that performed best in the first round of experiments: the norm change and Runge-Kutta test step algorithms. The standard deviation of the zero-mean Gaussian noise added to the observations was (a) 0.001 (b) 0.01 (c) 0.1 (f) 0.5

Chapter 9

Random Vortex Configuration

One important question is whether or not the symmetric and von Karman vortex configurations presented thus far are simply too contrived. Is there something critical and unphysical in the perfect nature of the alignment and symmetry of the vortices in these data sets? This initial symmetry is broken when the simulation is corrected with noisy data, which may be the cause of some of the issues in the previous section. To explore this, we ran an imperfect model experiment using a random vortex configuration. Specifically, we randomly positioned 10 vortices in the domain $[0, 1] \times [0, 1]$, with each vortex randomly assigned a strength of either 1 or -1. Figure 9.1 displays this initial condition. Starting from this configuration, we again ran a fine-resolution simulation–in which the timestep of 0.00001 was chosen using a convergence test—as the "truth" simulation. A coarser simulation (with timestep 0.01) was used to represent an inaccurate model run. Vortex trajectories from these two simulations are presented in Figure 9.2; the simulation length in each case was 50 seconds.

We conducted imperfect model experiments on these data sets by correcting the simulation in Figure 9.2(b) with "observations" from the high-resolution simulation in Figure 9.2(a). Using several different correction strategies, we ran an ensemble of experiments to generate plots of MSE versus the number of corrections. In our first set of experiments, we did not add any noise to the observations. The results are presented in Figure 9.3. To produce the "periodic" curve in this figure, the interval in the periodic

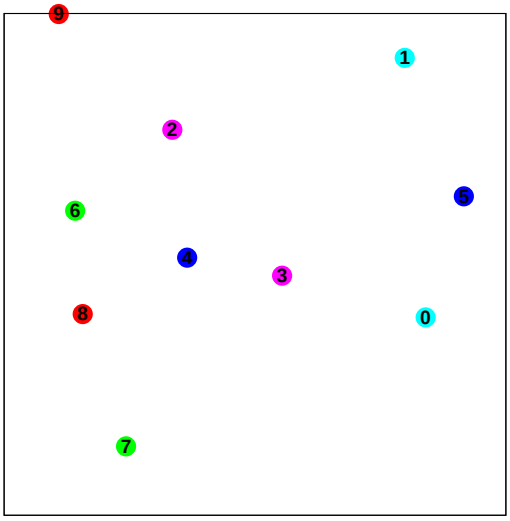


Figure 9.1: Randomly distributed vortices. To produce this initial condition, vortices were randomly placed in $[0,1] \times [0,1]$ and randomly assigned a strength of either -1 or 1. Vortices 0,3,4,5, and 7 have strength -1 and the rest have strength 1. We use this initial condition to explore symmetry-breaking issues.

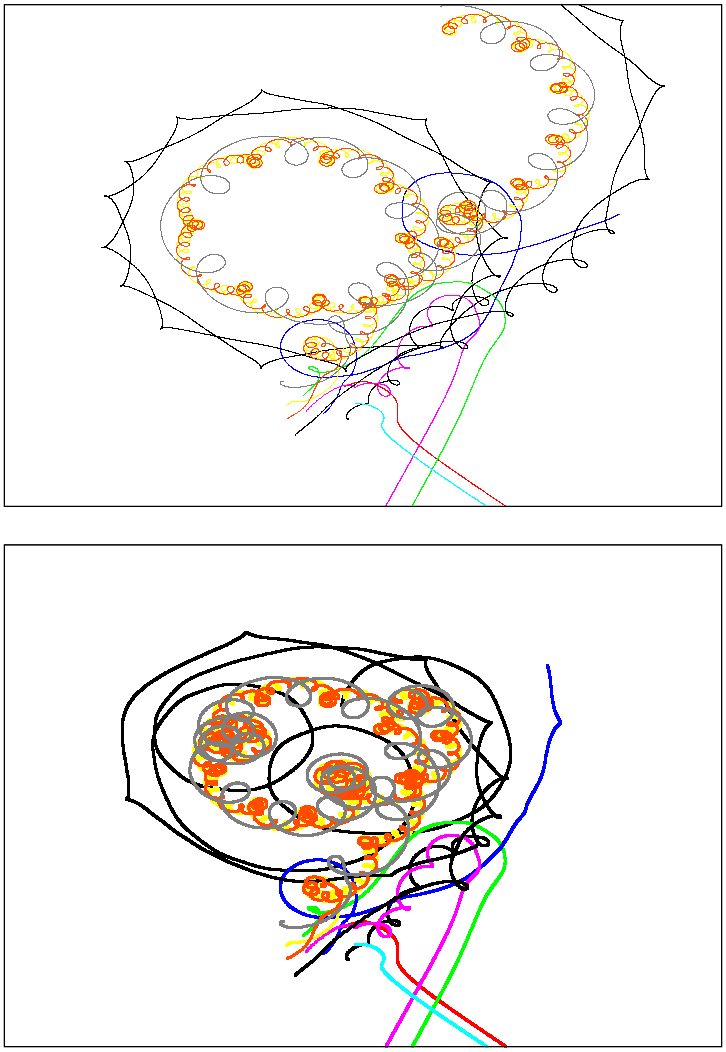


Figure 9.2: Simulations of point-vortex equations starting from the initial condition in Figure 9.1. Both simulations are 50s in length. (a) "Truth" simulation with a 0.00001s timestep (b) Correction simulation with a 0.01s timestep

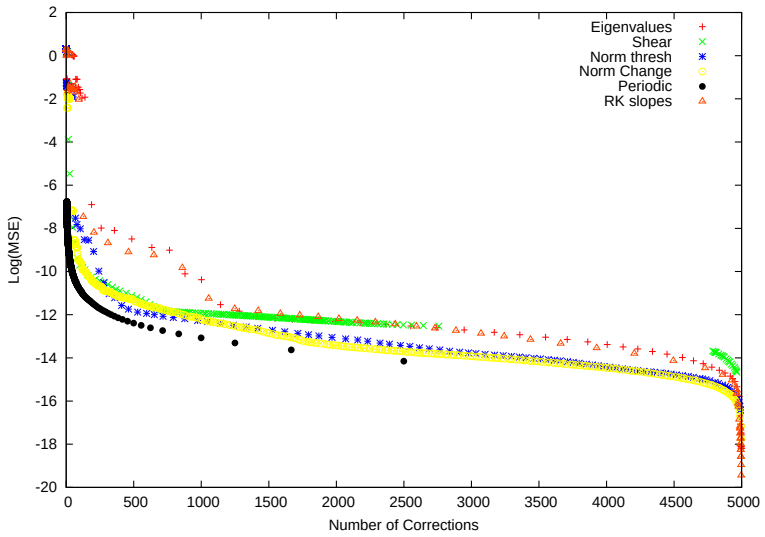


Figure 9.3: Comparison of dynamics-informed and periodic assimilation using the initial conditions in Figure 9.1. Each point in this figure, again, represents a single data-assimilation run; the MSE is plotted as a function of the number of corrections. Each curve is labelled with the correction strategy used to generate the results.

Method	**T_c [Min,Max]:Inc**
Norm Change	[8, 28]:0.05
Norm Thresh	[0, 60]:0.1
Eigenvalues	[10, 16]:0.05
Shear	[1, 20]:0.1
RK Steps	[0.0005, 0.005]:0.00005

Table 9.1: Thresholds for dynamics-informed techniques. Each entry in this table describes how thresholds were varied for a particular dynamics-informed assimilation experiment. The dynamics-informed technique is listed in the left column and the format of each entry is [minimum Tc, maximum T_c]:T_c increment.

correction strategy was varied from 0.01 to 10.01 in increments of 0.01. The other curves are labelled with the dynamics-informed correction strategy that was used to produce them. Recall that for a given dynamics-informed approach, varying the threshold value T_c for the algorithm will result in a different number of corrections. For example, for the norm-change method, T_c represents the correction threshold for a change in the norm of the Jacobian of induced velocity gradients. A higher threshold in this case will usually result in fewer corrections over the course of the simulation. Table 9.1 shows how T_c was varied for each dynamics-informed correction strategy.

The results in Figure 9.3 are quite surprising. Our goal in investigating the random data set was to ensure that the symmetry in the von Karman and symmetric initial conditions did not introduce artificial problems in the assimilation of noisy observations. We hypothesized that breaking the rigid symmetry of these configurations by assimilating noisy observations might have caused the problems in the experiments presented in the previous section. If this were true, we would expect that our dynamics-informed strategies would perform better in simulations with random initial conditions. This is not what happened. In fact, periodic correction outperforms all of the dynamics-informed correction techniques for this data set!

Of the dynamics-informed techniques, the norm-change and norm thresholding techniques work best for this data set. Also, it appears that for the random initial con-

ditions, the behavior of the norm change technique is similar to the norm thresholding technique. And, the eigenvalue and Runge-Kutta steps methods show a similar trend. These similarities are different from the similarities we saw for the symmetric and von Karman initial conditions. In those experiments, techniques based on *changes* in the norms (norm change, Runge-Kutta steps) behaved similarly, and techniques based on a static measure of dynamic stability (eigenvalues, shear, norm thresholding) obeyed a different trend.

Some of these results can be understood by analyzing the induced velocity norms for the vortices in an uncorrected simulation. Time series of these norms for several of the vortices in the simulation are presented in Figure 9.4. Vortices 1 and 8—the two black curves in Figure 9.2(b)—both display a pattern similar to Figure 9.4(a). Vortices 2 through 6—the red, green, blue, purple, and turquoise curves—exhibit a pattern similar to Figure 9.2(b). Notice that four of these vortices are travelling off in pairs similarly to the vortices in the von Karman configuration. And, we see that the resulting norm signature is similar for these vortices to the norm series displayed in Figure 7.2. Vortices 7 and 9 both display a pattern similar to Figure 9.4(c). These are the yellow and orange vortices in Figure 9.2(b) that are tightly oscillating around each other. This tight interaction seems to produce a wild oscillation in the velocity gradient norms. I believe that this norm pattern—which was not observed in either the symmetric or von Karman data sets—is what causes difficulties for our dynamics-informed techniques. For example, we can consider how the norm change technique would handle this norm pattern. If the norm change threshold is too high, we will not correct at any of the timesteps. But, if we lower the threshold, we will essentially begin to correct at all of the times because the norms are oscillating so dramatically. The point is that the dynamical information does not differentiate any of the correction times from the others. The norms displayed in Figure 9.4(d) for Vortex 10—the gray curve in Figure 9.2(b)—shows a similar oscillatory pattern, but it is much less dramatic.

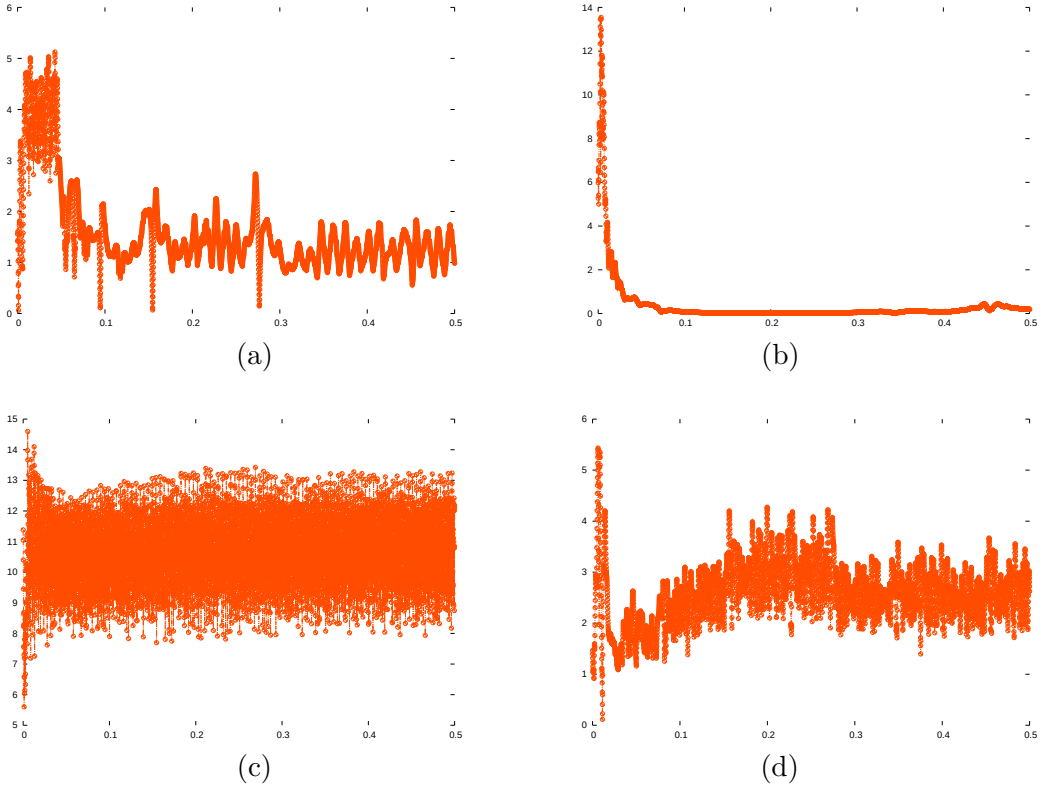


Figure 9.4: Velocity gradient norms as a function of time for vortex (a) 1 (b) 4 (c) 7 (d) 10 for a simulation starting from the initial condition in Figure 9.1.

A preliminary conclusion is that periodic correction is likely to be the best approach for data sets with a strong coupling interaction between the vortices.

Chapter 10

Newtonian Nudging

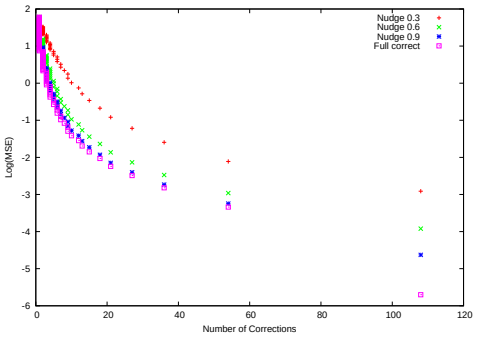
In the experiments described in Section 7, we used direct state variable replacement to correct the point-vortex model. As discussed in Section 2.2, much more sophisticated algorithms are typically used by the meteorological and oceanographic communities. This is primarily because the full-state of the system is not observed and the observations available are often different from the model state variables. Most of these researchers use techniques that attempt to account for the different error characteristics of the model and the observations, pushing the model closer to the observations when they are deemed to be the more-accurate source of information. Newtonian nudging, a common technique in control theory, is a very simplistic approach to this problem that nudges the model a fraction of the distance toward the observations on each correction cycle[80]. We will call this fraction the "nudge factor"; it will be expressed here as the percentage of the distance that the model is nudged toward the observations. For observations with small uncertainty, a larger nudge factor is more appropriate than for observations with larger uncertainty. In perfect-model experiments, in which the true state is known exactly and the observations are noise-free, complete state-variable replacement—i.e., a nudge factor of 100%—is clearly the best approach. However, for noisy observations, we would expect that a smaller nudge factor might produce better results.

Figures 10.1 and 10.2 display the results of the Newtonian nudging experiments

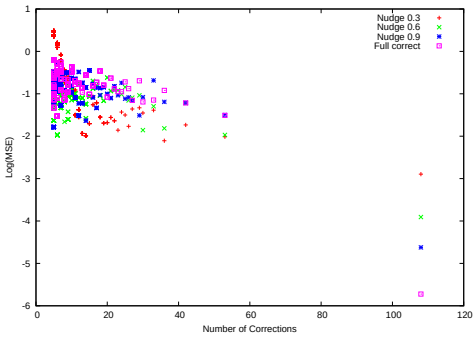
with the von Karman and symmetric initial conditions, respectively. We conducted experiments with various levels of Gaussian noise added to the observations. The σ value shown below each plot in the figure indicates the standard deviation of the zero-mean Gaussian noise added to the observations in the set of experiments that produced the plot. In the left column, we display the MSE results for experiments conducted using standard periodic correction. These can be compared to the results in the right column for the norm change dynamics-informed technique. In each plot there are four curves, each showing the results when a different "nudge factor" is used. The curve labelled "Full correct" represents a nudge factor of 100%; these curves are the same as the ones we saw in Figures 8.1 and 8.3. The curves labelled 0.3, 0.6, and 0.9 each represent an ensemble of experiments in which each state variable in the model was nudged 30%, 60%, or 90% of the distance to the noisy observations, respectively. To produce the curves for the periodic experiments, the correction interval was again varied from 0.1 to 50.1 in increments of 0.01. To produce the norm change curves, we used the same threshold values shown in Table 6.1.

For the periodic curves in the left column of these two figures, the behavior matches our expectations fairly well. For low levels of noise, larger nudge factors outperform smaller ones and a full correction is always the best choice. However, as the noise increases, a smaller nudge factor is better than a full correction. For example, for observations that include Gaussian noise with standard deviation 0.1, a nudge factor of 0.9 often outperforms a full correction; as we increase the noise further to $\sigma = 0.5$, nudge factors of $0.9, 0.6$, and, occasionally, 0.3 yield higher accuracy. There is also a similar trend for the norm change correction of the symmetric data set in the right column of Figure 10.2, although for higher levels of noise, the nudge factor that works best seems to be much less systematic.

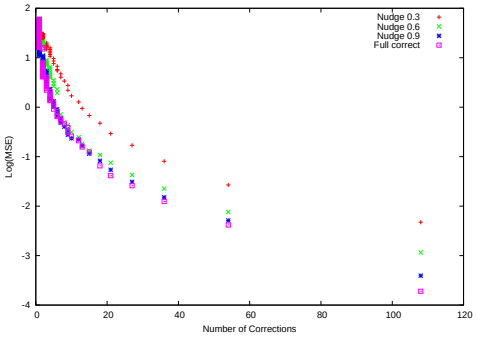
For the von Karman norm change curves in the right column of Figure 10.1, the results are more difficult to interpret. Recall the problem with the von Karman noise



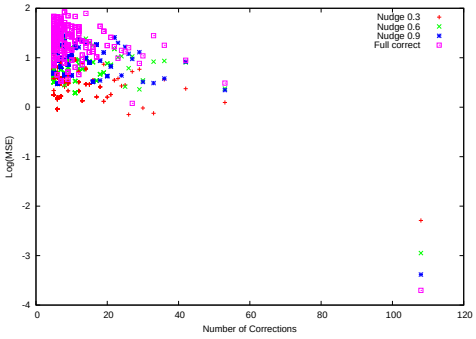
(periodic, $\sigma = 0.001$)



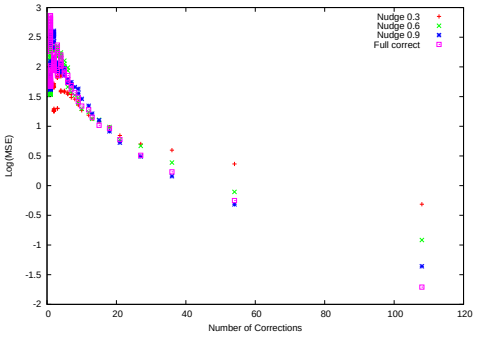
(norm change, $\sigma = 0.001$)



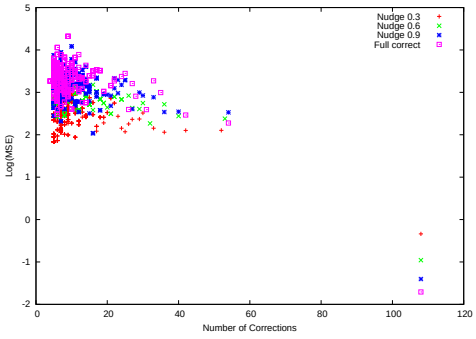
(periodic, $\sigma = 0.01$)



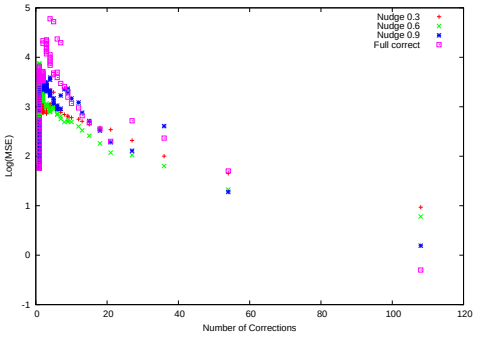
(norm change, $\sigma = 0.01$)



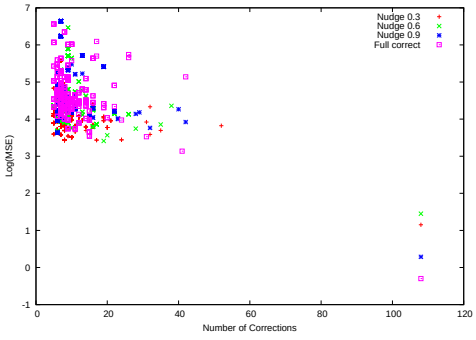
(periodic, $\sigma = 0.1$)



(norm change, $\sigma = 0.1$)

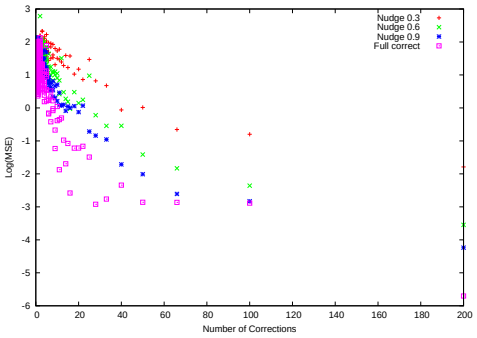


(periodic, $\sigma = 0.5$)

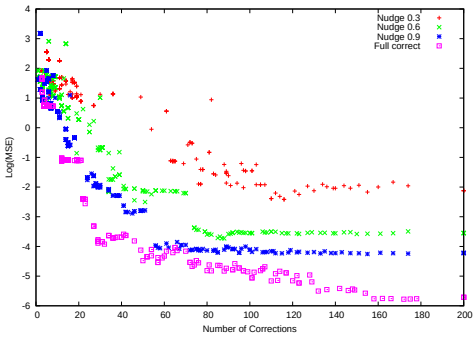


(norm change, $\sigma = 0.5$)

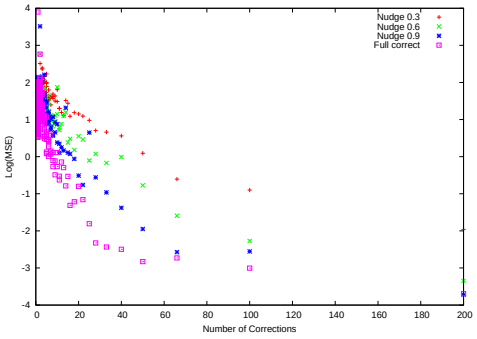
Figure 10.1: Newtonian nudging von Karman: simulations starting from the von Karman initial conditions from Figure 4.1(a).



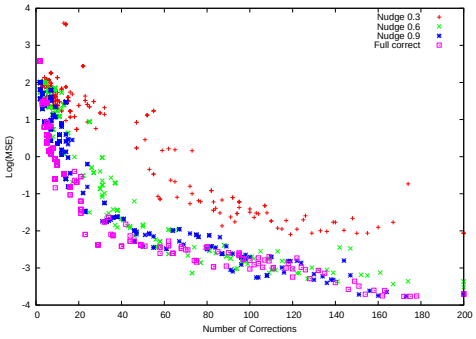
(periodic, $\sigma = 0.001$)



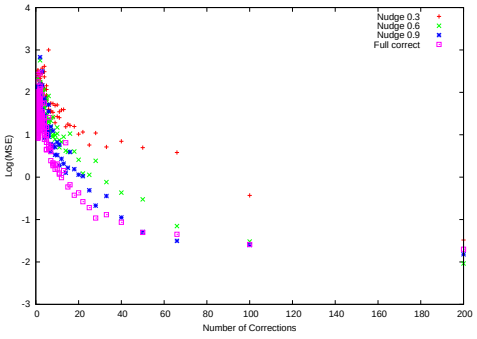
(norm change, $\sigma = 0.001$)



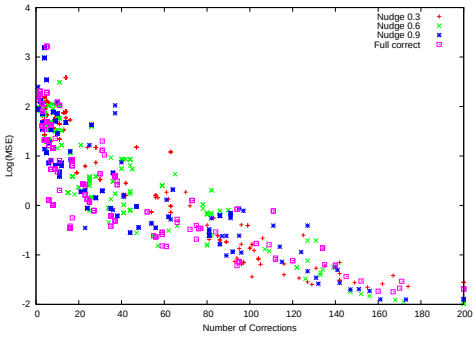
(periodic, $\sigma = 0.01$)



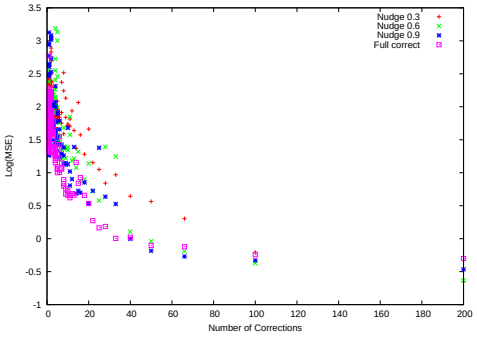
(norm change, $\sigma = 0.01$)



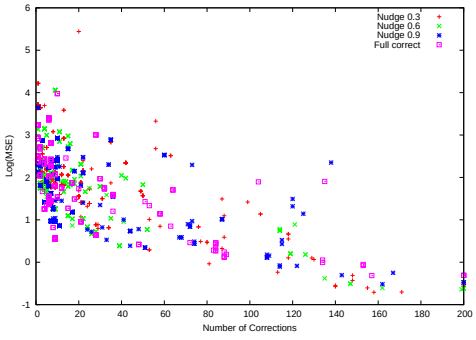
(periodic, $\sigma = 0.1$)



(norm change, $\sigma = 0.1$)



(periodic, $\sigma = 0.5$)



(norm change, $\sigma = 0.5$)

Figure 10.2: Newtonian nudging symmetric: simulations starting from the von Karman initial conditions from Figure 4.1(b).

results from Section 8: there was no correction applied after the first several simulation time steps because the stable pair dynamics of the vortices had materialized. After the last noisy correction was applied early in the simulation, then, the noise accumulated as the simulation proceeded. This situation will not be remedied by using the Newtonian nudging algorithm. Rather, the highest accuracy will be achieved by a simulation that makes that last correction as accurate as possible. So, the best nudge factor will depend on the model accuracy and the observational noise *at this critical correction time*. For low noise levels, lower nudge factors seem to work better, which indicates that the model is more accurate than the observations at the final correction time. For the highest level of noise ($\sigma = 0.5$), the results seem a little less systematic, with low nudge factors winning out occasionally and higher nudge factors being more successful for different correction counts. However, the results for higher levels of noise ($\sigma = 0.1, 0.5$) are all worse than an uncorrected simulation, which has MSE of 1.8, so these results are not worth analyzing further.

The Newtonian nudging algorithm provides a step in the right direction in terms of weighting the impact of the observations based on their noise characteristics. However, the method is fairly crude. There is no way to decide on an appropriate nudge factor given the observational noise. And, a constant nudge factor across the entire simulation is obviously not appropriate. The best nudge factor at a particular timestep depends on the accuracy of the simulation itself in addition to the accuracy of the observations. The Newtonian nudging algorithm does not track any information about the accuracy of the simulation, so there is no mechanism for deciding whether the observations or the model should receive a higher weight. More-sophisticated methods would take all of this information into account in the assimilation step. The ensemble Kalman filter, which was introduced in Section 2.2, is one such method that uses conditional probability and statistical techniques to provide the most accurate state estimate. In the next section, we will explore data assimilation using this technique, in the hopes

that this more-sophisticated algorithm will solve some of the problems with noise in our dynamics-informed assimilation.

Chapter 11

Data Assimilation Research Testbed

Ensemble Kalman filtering (EnKF) is an assimilation technique that has gained popularity in the atmospheric and oceanic communities in recent years. Recall from Section 2.2 that traditional Kalman filtering is based on the Bayesian statistics for approximating the mean and covariance of $p(\mathbf{x}^{\mathrm{t}}(t_i)|\mathbf{O}(t_i))$: the probability of the true state $\mathbf{x}^{\mathrm{t}}(t_i)$ given the available observations $\mathbf{O}(t_i)$ up to time t_i. EnKF circumvents some of the difficulties in the application of the traditional Kalman filter by using Monte-Carlo sampling to estimate the mean and covariance. The basic algorithm works as follows. The initial condition for the simulation is perturbed to produce an ensemble of model states with mean equal to the initial condition and a prescribed variance. All of these "ensemble members" are integrated forward in time to the analysis time. At this point, the mean of the ensemble provides a "prior" estimate of the true system state, and the ensemble spread provides information about the uncertainty in the estimate. After the observations have been assimilated using the algorithm described in [8], these ensemble statistics are referred to as a "posterior" estimate.

It is worth noting that the application of the EnKF algorithm inherently includes some dynamical information that can be exploited in a targeted observing strategy. In particular, the behavior of the ensemble members is an indirect measure of the dynamical sensitivity of the system—if they start close together in the phase space and spread quickly, we know that the model is in a region of high gradients. Thus, the

spread of the ensemble members can be used as an indicator of the error growth and a trigger for correction. The spread is calculated as part of the assimilation process, so on timesteps when we actually assimilate the data, there is no computational expense in using spread to trigger correction. However, in order to use it to trigger correction, the spread must be computed periodically to determine if correction is required. Thus, there is a computational cost incurred to compute the spread on timesteps when no correction occurs. The *spread-based correction* technique is the current state of the art in ensemble-based targeted observing, and it has been found to perform at least as well as (if not better than) periodic correction techniques [83]. However, no one has explored the use of this method in the context of point-vortex models.

Note that the dynamics-informed methods presented in Section 5 and the spread-based method are very similar. They work in different ways to estimate error growth in the simulation, which results from the dynamical sensitivity of the system. The difference is that the dynamics-informed techniques presented in this thesis reason explicitly and proactively to locate the times in the simulation where the error growth *starts*, whereas the spread-based technique reduces the error once the growth has actually been *observed*. The theory behind the dynamics-informed techniques is that making the simulation as accurate as possible in a dynamically sensitive region will minimize the *subsequent* error growth. The spread-based techniques reduce error after the fact. It is not clear which of these techniques will work better, and no one has performed a comparison in the context of point-vortex models. In this section, we present a preliminary study of this, first using the spread-based method to correct the point-vortex simulations starting from the initial conditions in Figure 4.1. We then compare this spread-based correction to the norm change method, which performed well in our experiments with noise-free data.

Ensemble filtering has been very successful and it is widely accepted by the data-assimilation research community. The method works very well for a wide variety of

assimilation problems, and the statistical methods and algorithms employed by the EnKF enable a clear understanding of the data-assimilation process. Another appeal of ensemble techniques is that there is a readily available software package that facilitates their use: the Data Assimilation Research Testbed (DART) [1]. Using this software, the start-up time for researching data-assimilation problems is greatly minimized. Several example models are provided with DART and adding a new model is fairly straightforward. One simply has to implement a model interface to "plug" the new model into the DART infrastructure. We implemented the point-vortex model in this framework; the Fortran90 code for our model is provided in Appendix A.

Using DART, it is relatively simple to conduct the perfect model experiments described in Section 2.2. The standard approach is as follows:

(1) Given an initial condition, run the model for a long time so that the model state converges onto the "attractor" of the dynamical system being modelled. Note that the definition of a "long time" is a tuning parameter that is model-dependent.

(2) Using the spun-up initial state from (1), run the model for a certain number of timesteps to produce a "truth," or "perfect model", simulation.

(3) Collect simulated observations from this truth run and perturb them with Gaussian noise (with variance prescribed by the user).

(4) Construct an ensemble of initial states such that the ensemble mean is equal to state (1) and the variance has some user-specified structure. Note that the initial condition from (1) is not included in the ensemble. This would unfairly pollute the experiments with too much knowledge about the true state.

(5) Run the simulation and correct the ensemble with the observations from the truth run using a Kalman filter update.

Note that there is an assumption in Step (1) that the system being modelled has an attractor. This is not the case for our point-vortex experiments, and the dynamics of interest often occur within in a short period of time before the vortices move apart. So, instead of running the model for a long time, as in step (1), we run the model for a single timestep just to generate required initialization ("restart") files for DART. This is not merely an academic distinction; it has some important implications for the behavior of DART. There are some processes in DART—for example, the adaptive inflation machinery described below—that require some lead time to settle onto the appropriate statistics. We will revisit this difference when we discuss some of the results from our perfect model experiments.

Figure 11.1(a) shows an example DART result for one state variable (x position of one vortex) in a perfect model experiment with the point-vortex model. We can use this time series to make the experimental concepts a little more concrete and show how to analyze results obtained with DART. The blue trajectory in this figure is the true state; the green trajectories display the priors[1] for 20 of the ensemble members in the simulation. The red curve provides the prior mean estimate. Notice the initial time, where we see the ensemble members distributed around the true state. The spread in the ensemble increases as the simulation moves forward, until the first analysis time, where we see the spread decrease significantly. This cycle of increasing spread followed by a huge spread reduction continues through each data-assimilation cycle.

In this example, the EnKF assimilation is not working very well—in the sense that the prior ensemble mean is not consistently tracking the true state. Another important measure for judging the success of a DART assimilation is to consider how often the truth curve lies somewhere within the area spanned by the ensemble members [5]. When this criterion is met, it indicates that the truth can be "explained" by the model's prior estimate of the state (to within the error variance indicated by the ensemble spread). If

[1] value at each analysis time before the observations are assimilated

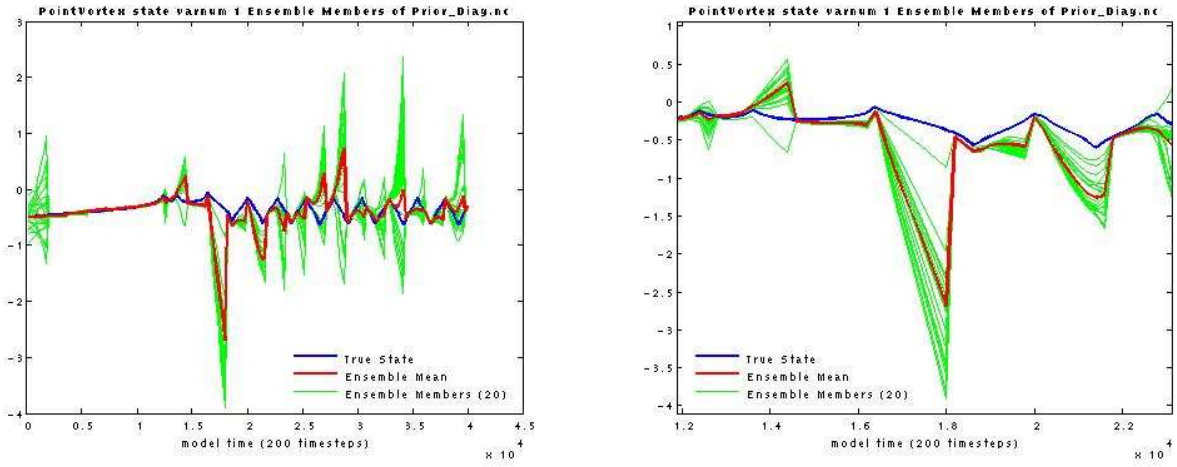


Figure 11.1: Analysis of DART Results. (a) Time series for the x position of a vortex in a representative simulation. The blue trajectory is the true state and the green trajectories display the priors for 20 of the ensemble members. The red curve shows the mean estimate. We can see that the assimilation is not working very well in the sense that the prior mean estimate is not tracking the true state. An example of filter divergence is illustrated in (b), where the mean estimate spikes downward.

this is not the case, the filter has *diverged*. Filter divergence, described in Section 2.2, is a symptom of the finite sample size used to estimate the covariance of the probability distribution. An example of this can be seen in Figure 11.1(b), where the ensemble mean spikes downward. Notice how the prior estimate has diverged significantly from the true value and the truth is not contained within the ensemble spread.

There are several techniques built into DART that can be used to address filter divergence. Perhaps the simplest is to increase the ensemble size to obtain a better statistical sample. For the experiments presented in this Section, we have chosen an ensemble of 100 members. This ensemble size might be considered computationally cost-prohibitive for many research applications, but initially we are not concerned with DART's performance. We simply want to explore targeted observing techniques using an assimilation algorithm that is well-known and respected in the data-assimilation community. Eventually, of course, if we want to use DART for real-time flow control applications, this computational burden will become critical. Fortunately, there are techniques in DART to obtain similar results using fewer ensemble members—namely, covariance localization and inflation. Covariance localization is used to prevent spurious correlations between uncorrelated state variables. Inflation serves to increase the spread to a more representative value; the simplest approach for doing this is to multiply the covariance matrix by a fixed number slightly larger than 1 at each analysis time. There is also a more-sophisticated technique, known as adaptive inflation [6] that incorporates a statistical update of the multiplier at each assimilation time, based on an analysis of the distance between the observations and the ensemble. For the experiments presented in this section, we have chosen to use adaptive inflation with an initial value of 1 and an initial standard deviation of 0.1.

Using this setup, we conducted perfect model experiments for the two sets of initial conditions from Figure 4.1. For each data set, we performed a similar analysis to the one described in Section 4 to compare periodic, spread-based, and dynamics-

Data Set	**Time Step**	**Sim Length**	**Periodic Intervals**	**Spread-based $\mathbf{T}_c$**	**Norm change $\mathbf{T}_c$**
von Karman	0.005	200	[1,200,1]	[0,20,0.2]	[0,20,0.2]
Symmetric	0.005	200	[1,200,1]	[0,2,0.01]	[0,2,0.01]

Table 11.1: Experimental setup for DART. This table displays the time steps, simulation lengths, and correction settings for each of the initial conditions studied with DART. A table entry of the form $[a, b, c]$ indicates that the relevant experimental parameter was varied from a to b in increments of c.

informed correction. The experimental parameters are provided in Table 11.1. For the symmetric experiments, we had to modify the simulation parameters slightly from those used in Section 7, increasing the timestep to 0.005 to avoid a blow-up of the simulation state variables[2] .

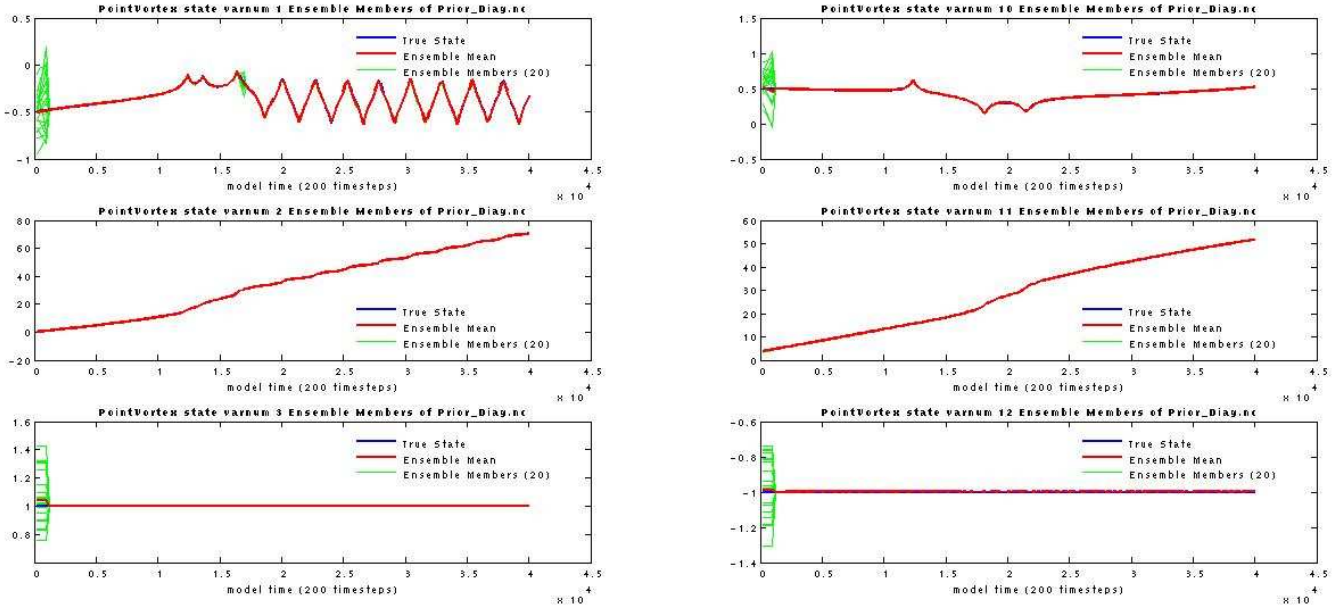
For each correction algorithm, we ran multiple DART experiments to generate a plot of the MSE versus number of corrections. For the periodic algorithm, we varied the correction interval to produce these curves. For spread-based and dynamics-informed correction, we generated observations every 1s. At these observation times, we determined whether to apply a correction based on whether the dynamical criterion was met. In the case of spread-based correction, if the ensemble spread for any state variable was above a pre-specified threshold, T_c, we corrected the model. For dynamics-informed correction, we chose to implement the norm change technique from Section 5: if the norm of the Jacobian of induced velocity gradients at any vortex position changed by a threshold percentage T_c between these 1s analysis times, we corrected the model. For both correction algorithms, T_c was varied as described in Table 11.1 to generate the MSE curves presented in this section.

Before running the ensemble of experiments just described, our first step was to verify that ensemble filtering actually works for the point-vortex model and the data sets we are studying. This is an important contribution of this thesis; no one has investigated

[2] The precise cause of this blow-up was never determined, and we did not explore remedies for this problem because we do not believe that the change in time step affects the results in a meaningful way.

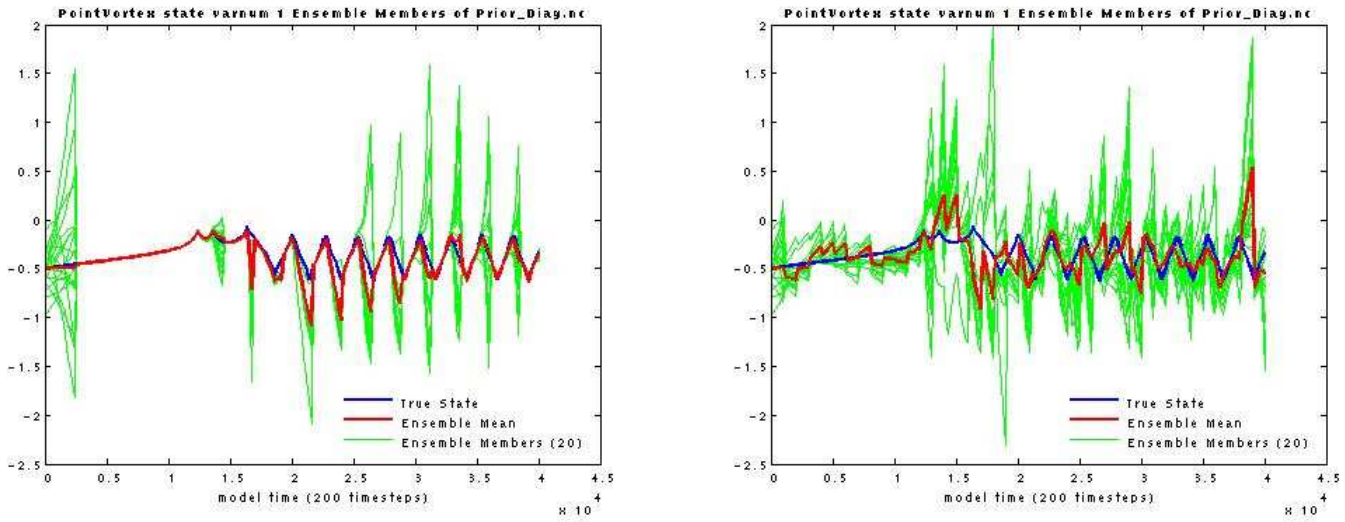
ensemble assimilation techniques with a point-vortex model. In the data-assimilation community, there is significant interest in point-vortex modelling for hurricanes and other planetary and oceanic systems with discrete vortices. Our results provide good news for this group of researchers—ensemble filtering seems to work quite well for the data sets we studied. Figure 11.2 provides some time-series plots similar to the one presented earlier in this section for a single simulation with the symmetric data set. These time series show the assimilation results for all three state variables for two of the vortices in a simulation that was corrected every 5 seconds (40 times in the course of the simulation). The observational error variance was 0.000001. Notice that we cannot distinguish the blue curve in most of these plots because the prior estimate of the mean (red curve) is tracking the true state (blue curve) so well. The assimilation is working beautifully for this test case.

The simulation in Figure 11.2 was corrected fairly frequently with very accurate observations. As expected, things get worse as the number of observations is decreased or the observational error is increased, as shown in Figure 11.3. In this figure, we see two time series for the x coordinate of the same vortex depicted in Figure 11.2(a). Part (a) of the figure displays the results of a simulation that is corrected every $12s$—that is, 16 times over the course of the simulation. The results here are not terrible, but as expected, the prior mean is not tracking the true state as well as it was in the simulation that was corrected more frequently. Part (b) comes from a simulation that was corrected every $5s$ (40 times), but with observations having an error variance of 0.01. It seems that the assimilation in this experiment is not working very well—we can see that the ensemble estimate is not "locking on" to the true state. However, it is also possible that the assimilation *is* working well, but that this is the best we can do given these noisy observations. It is interesting to note that this value of the error variance (0.01), where things seem to break down for the DART assimilation, is the same value that degraded the performance of our dynamics-informed methods below that of periodic correction



(a) (b)

Figure 11.2: DART assimilation is working. This figure shows time series plots for vortices (a) 1 and (b) 4 from a symmetric simulation that was corrected every 5 seconds with observations having error variance 0.000001. The top and middle time series show the evolution of the x and y coordinates, respectively. The bottom plots display the vortex strengths.



(a) (b)

Figure 11.3: Noise level and correction frequency impact accuracy. (a) shows a simulation that was corrected every $12s$ and (b) shows a simulation in which the observations have a larger error variance of 0.01. In both cases, we can see that the ensemble mean is not tracking the true state nearly as well as in Figure 11.2.

in Section 8. This suggests that an error variance of 0.01 or higher ($\sigma > 0.1$) makes successful assimilation much more difficult *in general*.

These considerations regarding noise and correction frequency should be kept in mind when evaluating the MSE versus number of corrections plots in this section. It can be argued that it does not make sense to compare targeted observing techniques based on correction timing when the ensemble assimilation process might not be working well. So, with this in mind, we should give more weight to results obtained for a higher number of corrections and lower observational error. This is one of the limitations in conducting a huge number of experiments with a tuneable software package. Theoretically, it might be possible to tune the DART parameters to obtain better results with each of the simulations in these experiments (even those with higher observational error and less frequent correction). However, this would make it difficult to generalize our results. For the purposes of this thesis, we simply ask the reader to keep these caveats in mind when analyzing the results.

Figures 11.4 and 11.5 display the MSE versus number of correction curves for the three correction techniques we investigated, starting from the initial conditions in Figures 4.1(a) and 4.1(b), respectively. Here, we compare the periodic (blue curve), spread-based (green curve), and norm change (red curve) correction techniques. Each plot provides the results for the ensemble of experiments represented by the parameters in Table 11.1, in which each simulation is corrected with observations having a fixed error variance.

For the von Karman results in Figure 11.4, there is good news. For all levels of noise, the spread-based technique usually significantly outperforms periodic correction. The norm change dynamics-informed technique also does well for the lower levels of noise. However, the spread-based method is the clear winner for this data set, especially when we consider that there is a lower computational cost involved in using this technique. These results are interesting because in Section 8, we found that noise

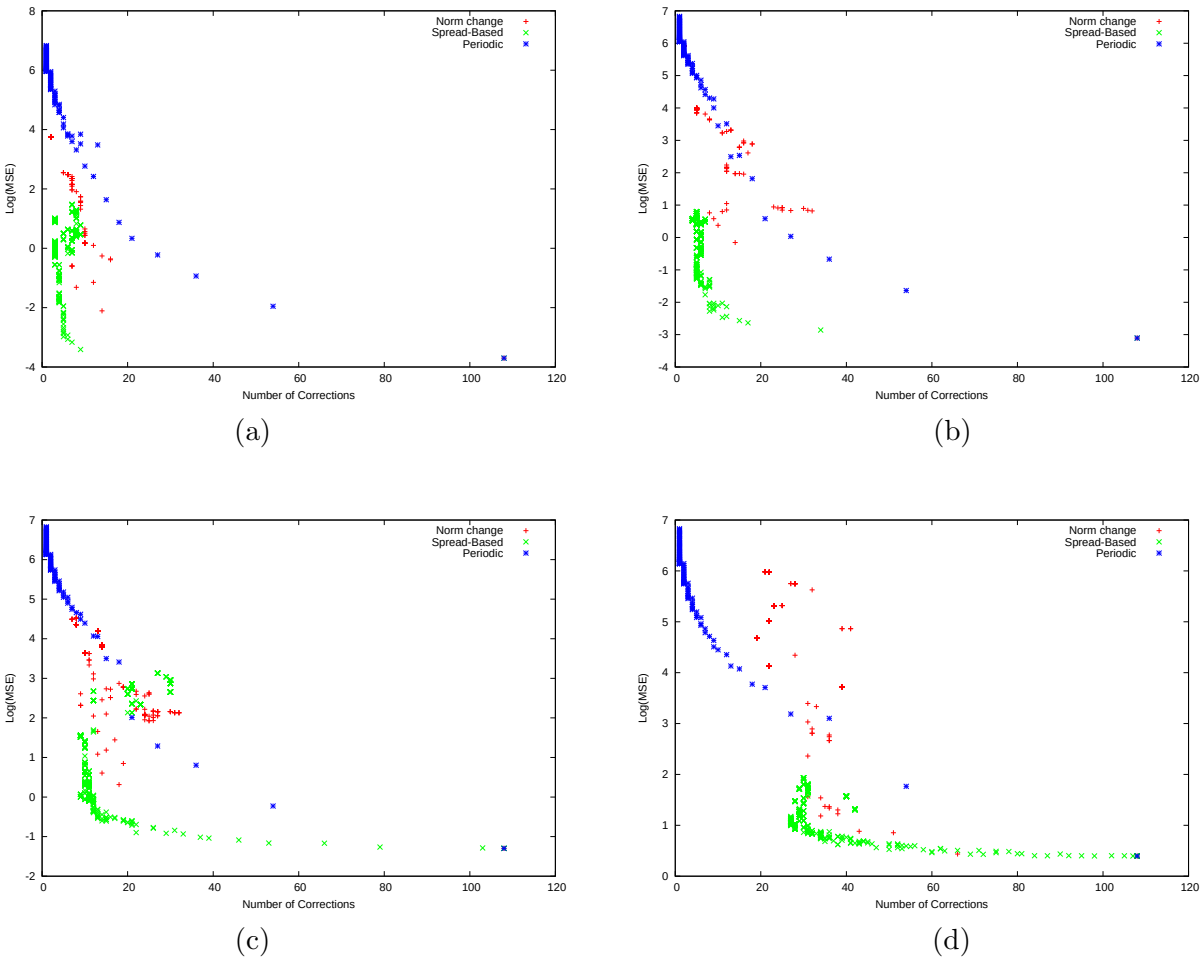


Figure 11.4: DART: simulations starting from the von Karman initial conditions from Figure 4.1(a). Each point in these plots provides the MSE when a single simulation is corrected with noisy observations; the goal is to compare periodic and dynamics-informed correction when the same amount of noise is added. The plots display the MSE results for spread-based, norm change, and periodic correction. The variance of the zero-mean Gaussian noise added to the observations was (a) 0.000001 (b) 0.0001 (c) 0.01 (f) 0.25

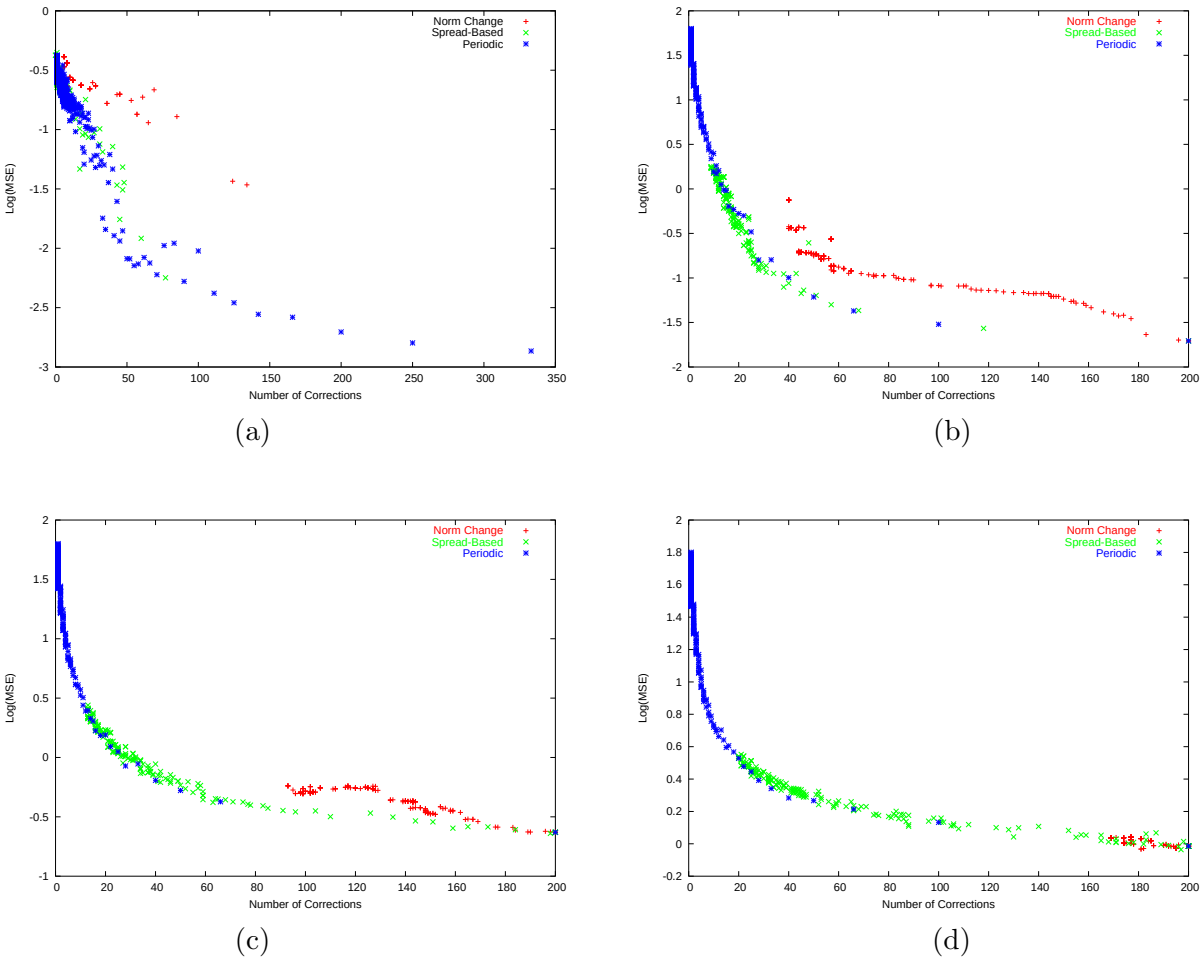


Figure 11.5: DART: simulations starting from the symmetric initial conditions from Figure 4.1(b). Each point in these plots provides the MSE when a single simulation is corrected with noisy observations; the goal is to compare periodic and dynamics-informed correction when the same amount of noise is added. The plots display the MSE results for spread-based, norm change, and periodic correction. The variance of the zero-mean Gaussian noise added to the observations was (a) 0.000001 (b) 0.0001 (c) 0.01 (f) 0.25

completely destroyed the dynamics-informed assimilation. It is clear that for the von Karman data set, an algorithm that incorporates information about the error in the model state estimate is critical.

Figure 11.5 shows that the spread-based technique also works very well for the symmetric initial conditions, achieving performance that is as good as or better than periodic correction. However, the norm change technique is worse than both spread-based and periodic correction even for low levels of noise. For this data set, it appears that our attempt to minimize error growth by producing an accurate state estimate in a region of high gradients is not as successful as reducing the error growth either periodically or only when it is observed.

There are several significant contributions from our research with DART. No one has investigated ensemble Kalman filtering in a point-vortex model. Our confirmation that EnKF is successful in point-vortex assimilation experiments is an important result for the data-assimilation community. Also, prior research has confirmed that the spread-based technique performs well for grid-based models, but no one has used it to correct a point-vortex model. In this section, we have verified that spread-based correction works as well or better than periodic correction for the two point-vortex data sets studied. It is also an interesting finding that triggering correction proactively, based on a measure of dynamical sensitivity, is not as successful as triggering correction after error growth has been observed. In fact, this strategy is often worse than periodic correction. This result is a significant contribution to ensemble-based data-assimilation research in point-vortex systems.

Chapter 12

Initial Conditions Derived from PIV Measurements

The ultimate goal of our research is to determine the best and most efficient scheme for correcting the point-vortex model. Our hope is that a highly effective data-assimilation strategy will boost the accuracy of this reduced-order model, making it practical for real-time modelling and control applications. Recall from Section 3 that a laboratory planar air jet is our testbed for this assimilation research. In the initial stages of this thesis—when experimental data from the jet was not available—we attempted to construct realistic initial conditions for our explorations of data-assimilation dynamics. In the initial conditions we developed—the symmetric, von Karman, and random data sets from the previous sections—the strengths assigned to the vortices were arbitrary. So, the physicality of these vortex configurations is somewhat questionable. Clearly, working with real data gathered from a physical system is highly preferable. The laboratory setup and vortex extraction techniques described in Sections 3 and 3.1 provide the data we need to do just that.

Our first step in working with the experimental data was to repeat the same analysis we conducted for the von Karman, symmetric, and random data sets in order to study the performance of dynamics-informed techniques when the initial conditions are extracted from PIV data. Given the PIV data from Figure 3.3(c), we performed vorticity thresholding to extract point-vortex positions and strengths. The resulting vortex configuration—shown in Figure 12.1—provided the initial condition for our imperfect

model experiments. Starting from this initial condition, we ran two simulations—one with high resolution to represent the "truth" and a second with a larger time step and fairly large error. We use "observations" from the "truth" simulation to correct the coarser one using a simple direct replacement assimilation approach. Both of these simulations were Runge-Kutta integrations of the Biot-Savart equations for $50s$. A timestep of $0.0001s$ was chosen for the "truth" simulation using a convergence test, and a timestep of $0.01s$ was used for the corrected simulation. The resulting vortex trajectories are shown in Figure 12.2.

The vortices in Figure 12.1 are in an antisymmetric configuration similar to the von Karman configuration studied in Section 7. However, there are three important differences in this configuration. First, the vortex spacings are a little different. Recall that for the von Karman case, the ratio a/b depicted in Figure 4.1(a) was 0.281. For this real data initial condition, this ratio is $a/b \approx 0.5$. Second, the von Karman configuration we studied contained an even number of vortices and the real data initial condition contains an odd number. Finally, the vortices in the two configurations are rotating in opposite directions. In the initial configuration in Figure 12.1, the vortices in the left column are rotating in a clockwise direction and those in the right column are rotating counter-clockwise. This is the reverse of the situation for the von Karman vortex configuration. The result of these discrepancies is a significant difference in the dynamical evolution for the two systems. For the von Karman case, the vortices pair up and travel off in the negative x-direction. In Figure 12.2(a), two of the vortices also group together into a pair (the red and green trajectories), but they are rotating around each other instead of travelling off together. This rotation is based on the fact that the circulations of the vortices in this pair have the same sign. It is interesting that the dynamics of this real-data case are so significantly different from both of the data sets–von Karman and symmetric—that we designed to mimic the experiment.

Our next step was to perform a comparison of point-vortex correction strategies

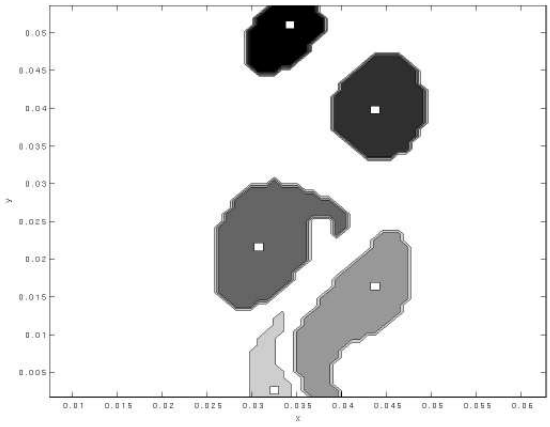


Figure 12.1: Experimental data initial condition. Vortices were extracted from the PIV measurement of the planar air jet shown in Figure 3.3(c). The vorticity thresholding technique was used to determine the vortex positions and strengths. Recall that the white square in each figure is the location where each point vortex was placed by the extraction algorithm. The strengths of the vortices (in top to bottom order) 0.0030, -0.0069, 0.011, -0.012, and 0.0034

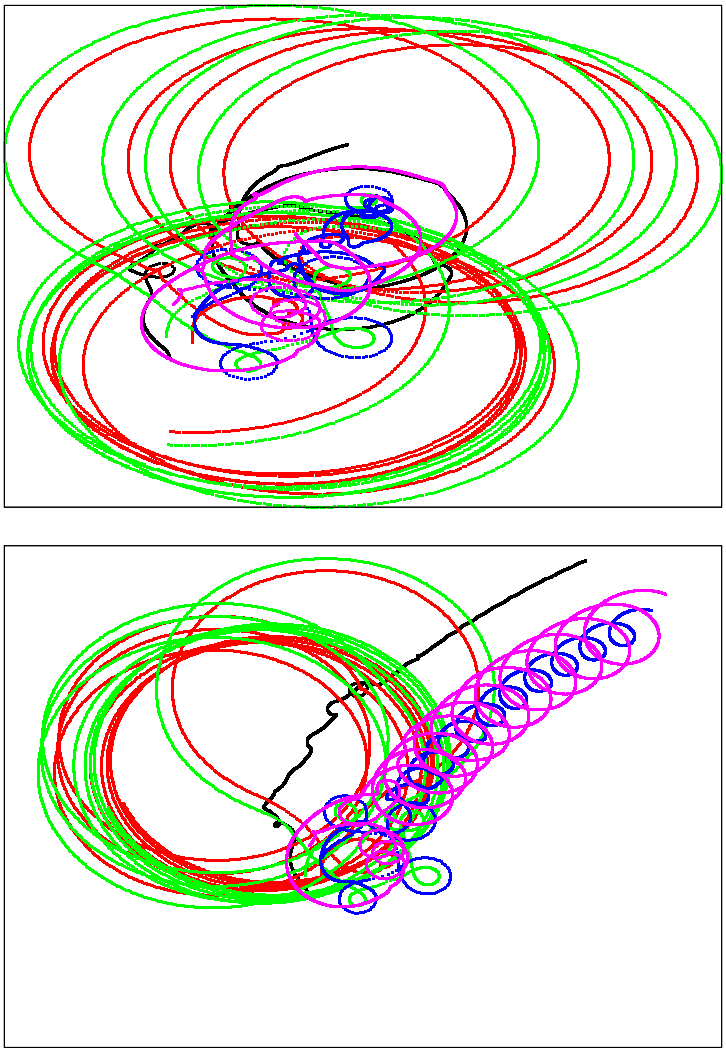


Figure 12.2: Simulations starting from the initial conditions in Figure 12.1. Both simulations are 50s in length (a) "truth" simulation with a 0.0001s timestep (b) correction simulation with a 0.01s timestep

Method	**T_c [Min,Max]:Inc**
Norm Change	[0, 42]:0.1
Norm Thresh	[9, 60]:0.5
Eigenvalues	[9, 60]:0.5
Shear	[0.2, 10]:0.01
RK Steps	[0, 0.0004]:0.0000005

Table 12.1: Thresholds for dynamics-informed techniques. Each entry in this table describes how thresholds were varied for a particular dynamics-informed assimilation experiment. The dynamics-informed technique is listed in the left column and the format of each entry is [minimum Tc, maximum T_c]:T_c increment.

using the data sets from Figure 12.2. We started by conducting noise-free experiments to see how the assimilation performs in this limiting case. We again ran an ensemble of experiments for several different correction strategies, including periodic, norm change, norm thresholding, eigenvalue, shear, and Runge-Kutta test step methods. To generate curves for MSE as a function of the number of corrections, the periodic correction interval was varied from 0.05s to 50.05s in increments of 0.05s. The thresholds used for the various dynamics informed techniques are provided in Table 12.1. The results of these experiments are presented in Figure 12.3. We can see that all of the dynamics-informed techniques outperform periodic correction for correction counts above 500. This is highly encouraging, because we know that this is a realistic vortex configuration. Similar to results for the symmetric and von Karman data sets from Section 7, we again see that the norm change and Runge-Kutta test step methods perform best for this data set.

If we again consider the time series of the induced velocity norms, we can see that our success on this data set is in agreemement with our results thus far on predictable norm patterns. Figure 12.4 displays the time series of induced velocity gradient norms for the five vortices in an uncorrected simulation. Vortex 1 is the black curve in Figure 12.2(b) and vortices 2 and 3 are the red and green curves that are orbiting around each other in large circles. These vortices have opposite rotation direction, but

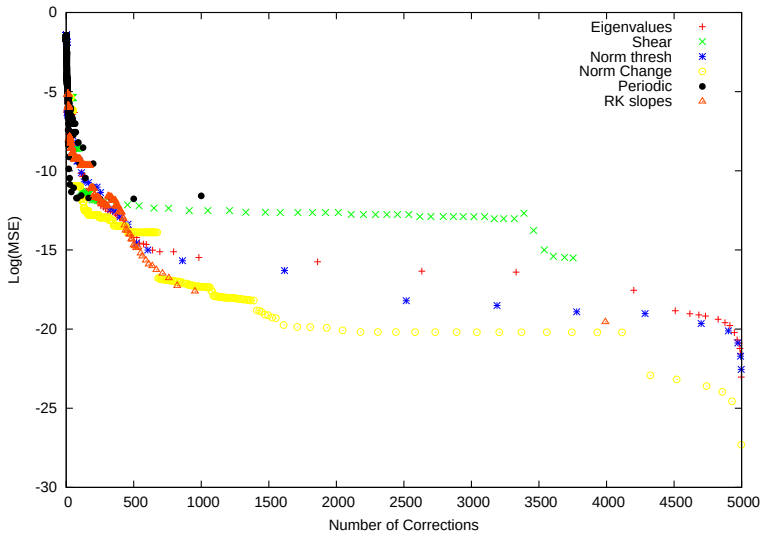


Figure 12.3: Comparison of dynamics-informed and periodic assimilation using the initial conditions in Figure 12.1. Each point in this figure represents a single simulation; the MSE is plotted as a function of the number of corrections. Each curve is labelled with the correction strategy used to generate the results.

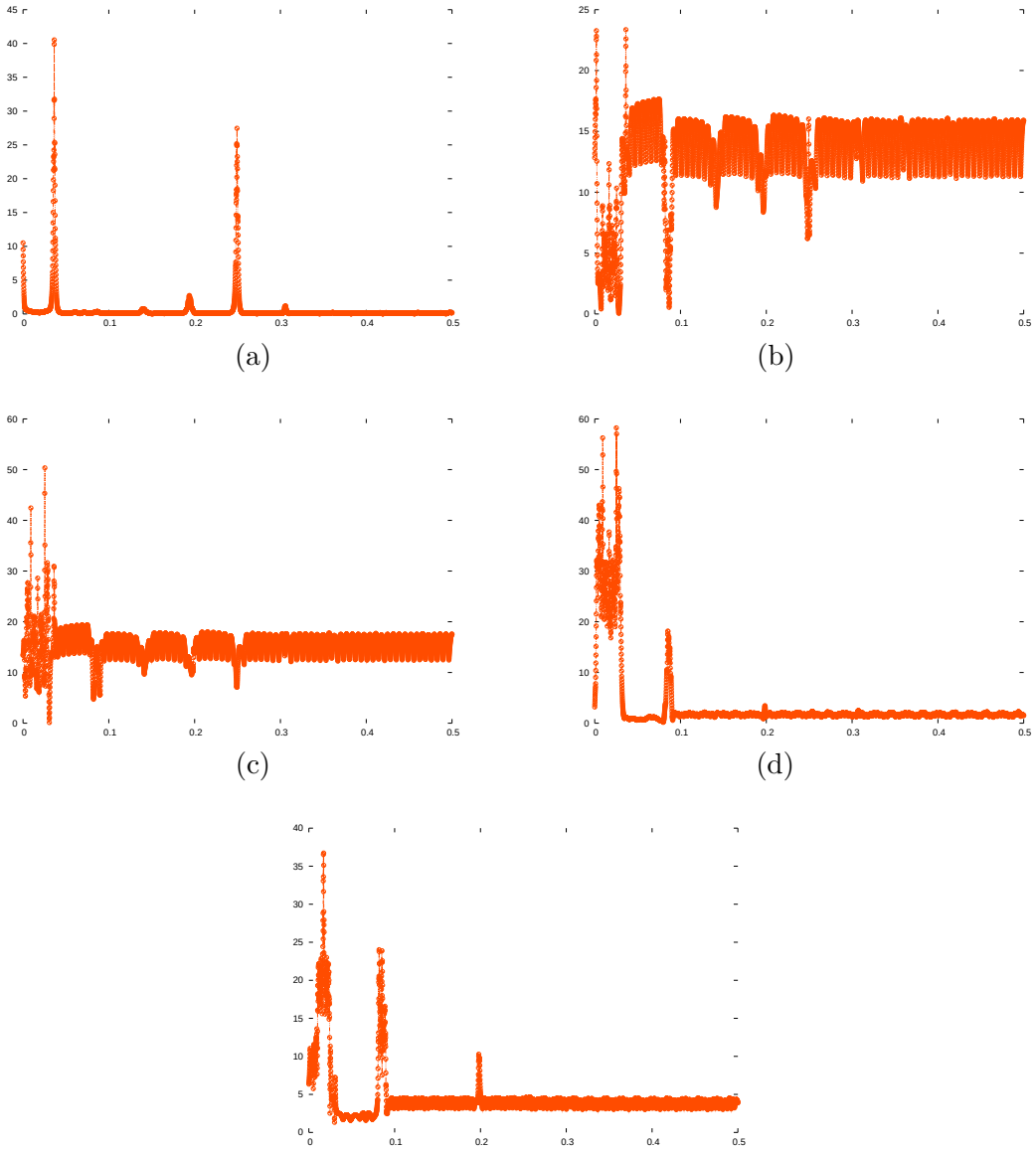


Figure 12.4: Velocity gradient norms as a function of time for vortex (a) 1 (b) 2 (c) 3 (d) 4 (e) 5 for a simulation starting from the initial conditions in Figure 12.1.

are close to the same strength, so the circles traced out by their trajectories are almost the same size. Vortices 4 and 5 are the blue and purple curves in Figure 12.2(b). These vortices are also rotating around each other, but the radius of the circle traced out by each one is different because their strengths—0.012, and 0.0034—differ by almost an order of magnitude. The norms for vortices 1, 4, and 5 look similar in character to those observed for the symmetric data set. They are relatively flat, but are punctuated by spikes at a few locations. This similarity may explain the qualitatively similar performance of the dynamics-informed techniques between the two data sets. The norms for vortices 2 and 3 display some of the oscillatory patterns that caused problems for our random initial conditions. However, the frequency of the oscillation is much lower, and the time series is also punctuated by downward spikes. The oscillation frequency of the norms seems to be related to how tightly the vortices are coupled in the simulation. In this simulation, in which the coupled vortices have larger orbits, the dynamics-informed techniques work quite well.

12.1 Decomposition of the Initial Condition into Smaller Vortices

Recall from our discussion in Section 3.3 that modelling the vorticity field with only five point vortices is a fairly coarse approximation. We explored one strategy for improving the approximation by decomposing each large-scale vortex into smaller vortices of equal strength. This decomposition, as discussed on page 45, enables the point-vortex model to advect the vorticity at a more fine-grained level. The strength of the vortices do not change in a two-dimensional point-vortex simulation, so the ability to move more point-vortices around in the state space could be beneficial. Also, using vortices of equal-strength provides the advantage that we could decrease our model state variables by 1/3 because vorticity is constant [1] . Using the algorithm described in

[1] We would have to keep track of which ones were positive and which negative, however, but this could be done by splitting the state vector into positive and negative vortices.

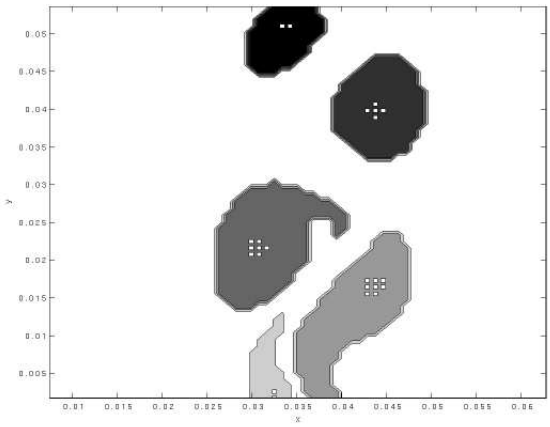


Figure 12.5: Decomposition of large vortices. We divided the large vortices from Figure 12.1 into smaller point vortices of equal strength. Recall that the white squares are the locations where the point vortices were placed by the extraction algorithm. The strength of each point vortex in this initial condition was ± 0.0015.

Section 3.3, we performed a decomposition on the data set in Figure 12.1. The resulting point vortex positioning is depicted in Figure 12.5. The strength of each vortex was chosen such that the weakest vortex would be assigned two point vortices; in this case, the strength was ± 0.0015.

As usual, we performed the imperfect model experiments on this vortex configuration. Figure 12.6(a) depicts the truth simulation—a 20s simulation with a timestep of 0.0001. The time step was chosen with the same convergence test used earlier. Using observations from the "truth" simulation, we corrected the 0.01s timestep simulation in Figure 12.6(b) at periodic intervals ranging from 0.05 to 20.05 in increments of 0.05. We also explored several of the dynamics-informed correction strategies for this data set using the thresholds in Table 12.2. Comparing these thresholds to those in Table 12.1 also corroborates the conjecture that this configuration is highly unstable. The high end of the threshold range is roughly equal to the maximum of each measure over the simulation. For the Runge-Kutta test step method, for example, the maximum difference between test steps over the course of the simulation was about 3.6. For the

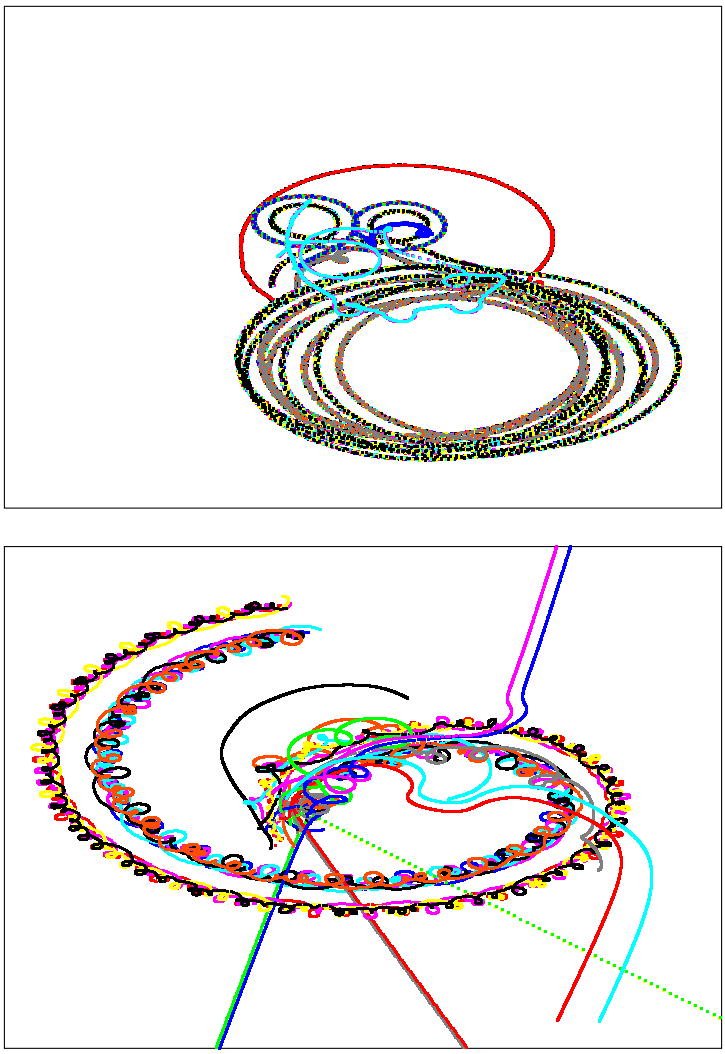


Figure 12.6: Simulations starting from the initial conditions in Figure 12.5. Both simulations are 20s in length (a) "truth" simulation with a 0.0001s timestep (b) correction simulation with a 0.01s timestep

Method	**T_c [Min,Max]:Inc**
Norm Change	[80,300]:1
Norm Thresh	[12, 200]:0.5
Eigenvalues	[12, 200]:0.5
Shear	[12, 200]:0.5
RK Steps	[0, 3.6]:0.01

Table 12.2: Thresholds for dynamics-informed techniques. Each entry in this table describes how thresholds were varied for a particular dynamics-informed assimilation experiment. The dynamics-informed technique is listed in the left column and the format of each entry is [minimum Tc, maximum T_c]:T_c increment.

simulation with only five point vortices, the maximum was roughly 0.0004. This is a difference of four orders of magnitude! Perhaps the strength quantum chosen for this vortex configuration was too large given the small spacing between vortices.

The drastic differences between the simulations in Figures 12.6 and 12.2 are also a concern. In our decomposition of the five vortices into smaller-scale point vortices of equal strength, the goal was to place the smaller vortices such that they more-accurately model the distributed vorticity of the larger-scale vortex. The high-resolution simulation in Figure 12.6(a) has some qualitative similarities to the simulations in Figure 12.2, but the simulation in Figure 12.6(b) is significantly different. This suggests that perhaps our approach for decomposing the vortices is not achieving our desired result. In our future work, we will explore other methods of large-scale vortex decomposition.

In spite of these concerns, it is still worthwhile to look at the MSE results for the imperfect model experiments, which are presented in Figure 12.7. The outcome is quite different than in the large vortex simulation above. For these experiments, periodic correction outperforms all of the dynamics-informed techniques. This is similar to the behavior we saw for the random initial conditions. It is not surprising, because we again see the very tight coupling of two groups of vortices that move out in a spiral shape in Figure 12.6(b). If we look at the norms for some of these vortices in Figure 12.8, we again see the pattern of high-frequency, high-amplitude oscillations similar to the norms for the tightly coupled vortices in Figure 9.4. It seems that for both sets of initial conditions, the tight coupling dynamics for some of the vortices causes a highly unstable dynamical system that is very difficult to predict with the stability measures we are using for our dynamics-informed techniques.

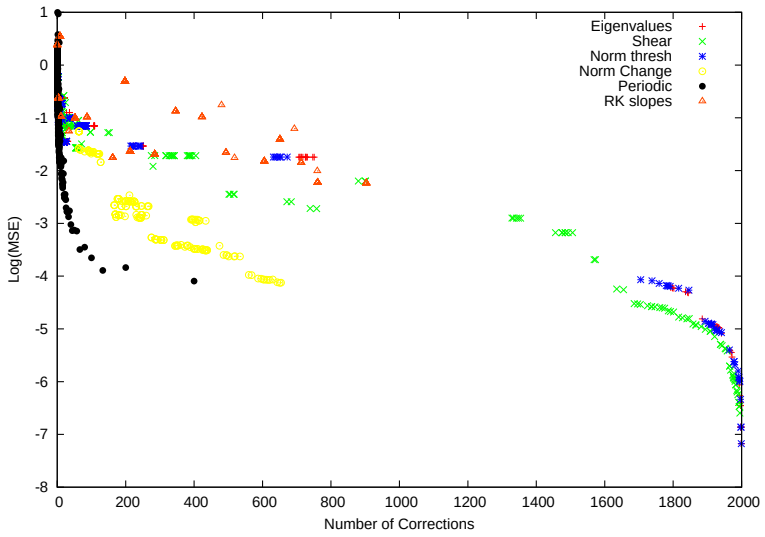


Figure 12.7: Comparison of dynamics-informed and periodic assimilation using the initial conditions in Figure 12.5. Each point in this figure represents a single simulation; the MSE is plotted as a function of the number of corrections. Each curve is labelled with the correction strategy used to generate the results.

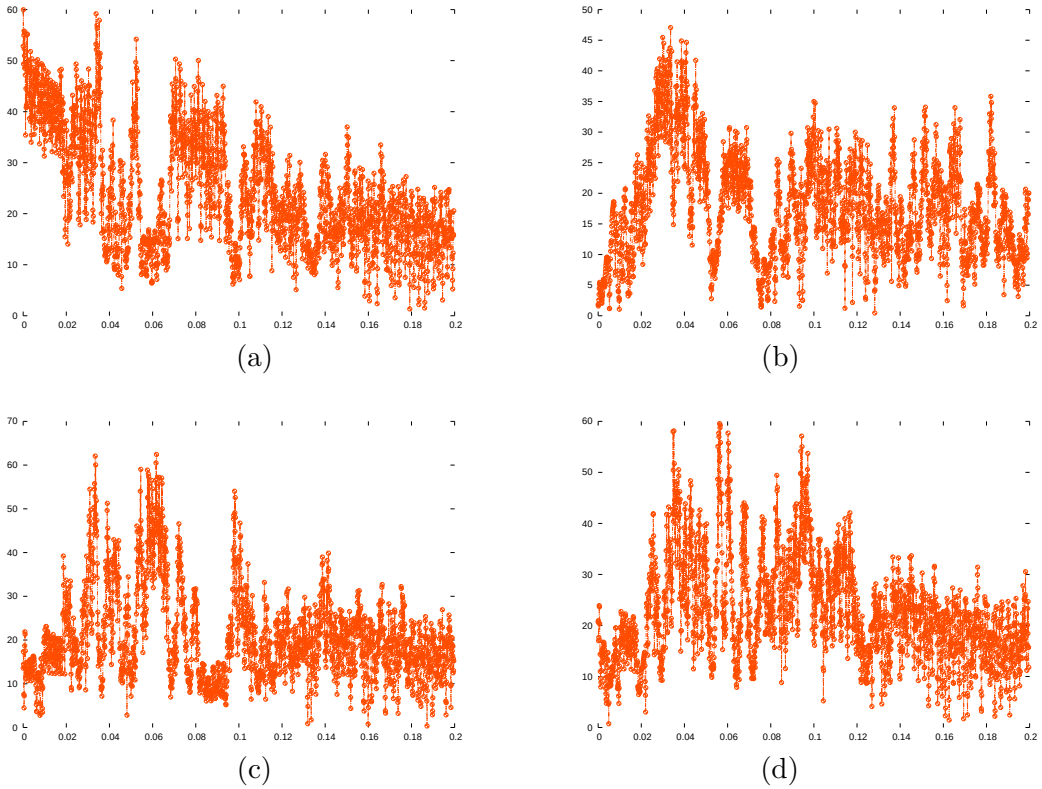


Figure 12.8: Velocity gradient norms as a function of time for vortex (a) 2 (b) 4 (c) 14 and (d) 17 for a simulation starting from the initial conditions in Figure 12.5.

Chapter 13

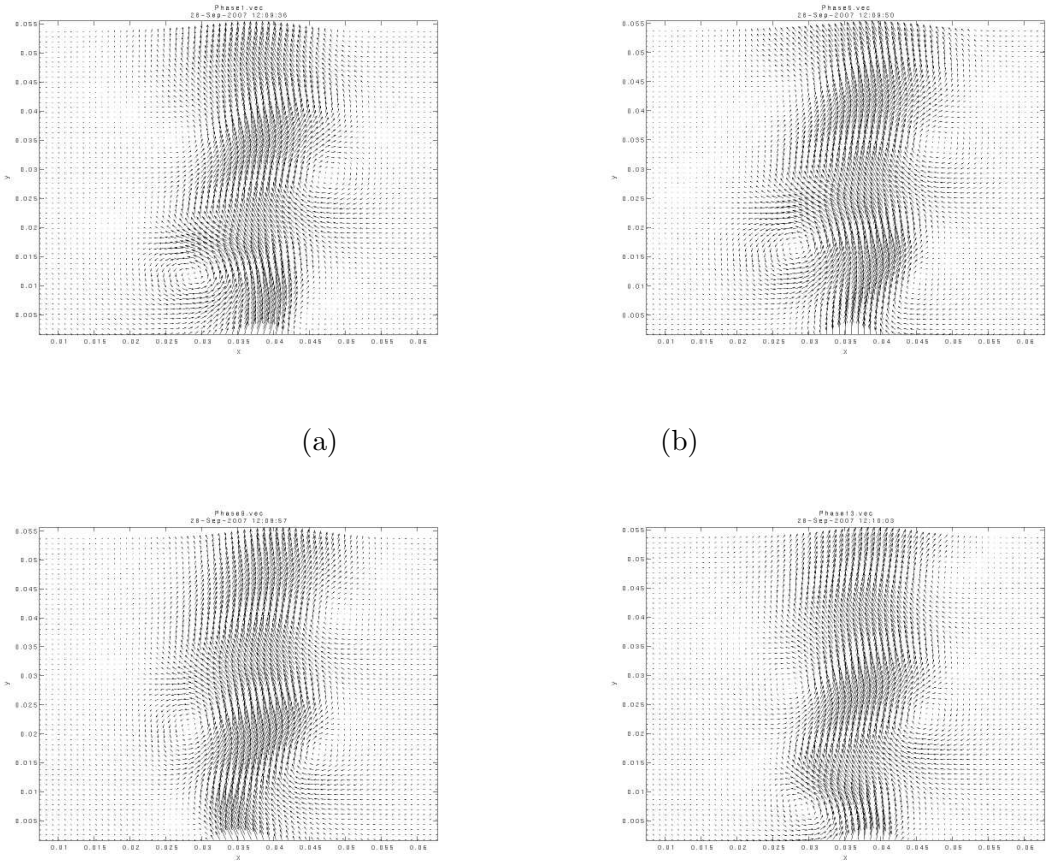
A Real-World Assimilation Experiment

The majority of data-assimilation research relies solely on Observing System Simulation Experiments (OSSEs), numerical experiments in which the model is used to generate synthetic observations for the assimilation. The perfect and imperfect model experiments we have presented thus far in this thesis are classes of OSSEs. While these numerical experiments are a useful first step, it is often the case that the results will not generalize to real-world assimilation problems. Thus, it is critical to take the research one step further and study an assimilation that uses real observations from a physical system.

In our case, we are fortunate to have a controlled laboratory setup of a planar air jet and a particle image velocimetry (PIV) system that can accurately capture the jet's velocity field. More details on the laboratory setup were provided in Section 3. Our approach to conducting a real-world assimilation experiment was as follows. We ran a high resolution simulation with timestep equal to 0.0001s—the same timestep used to produce the "truth" simulation from Section 12. We chose a simulation length equal to 60 ms, which is slighly larger than the 59.4 ms time required for a single cycle of the jet at its natural frequency, 16.83 Hz. Vortices extracted from the PIV observations were then used to correct the model by replacing the model state variables with the observed values. The correction frequency was dictated by the assimilation strategy under investigation.

To produce the observations, we used the PIV equipment described in Section 3 to capture 17 snapshots of the velocity field over a single cycle. Since the PIV capture window is not large enough to measure the velocity field over the entire jet stream, we took the measurements slightly downstream from the jet slit, where the vortices were more developed. It is important to note, however, that there is unmodelled vorticity both upstream and downstream from our PIV capture window. Recall from Section 3 that we had to take phase-averaged measurements because our observing equipment could not keep up with the natural frequency of the jet. Four of these PIV phase-averaged velocity fields are displayed in Figure 13.1.

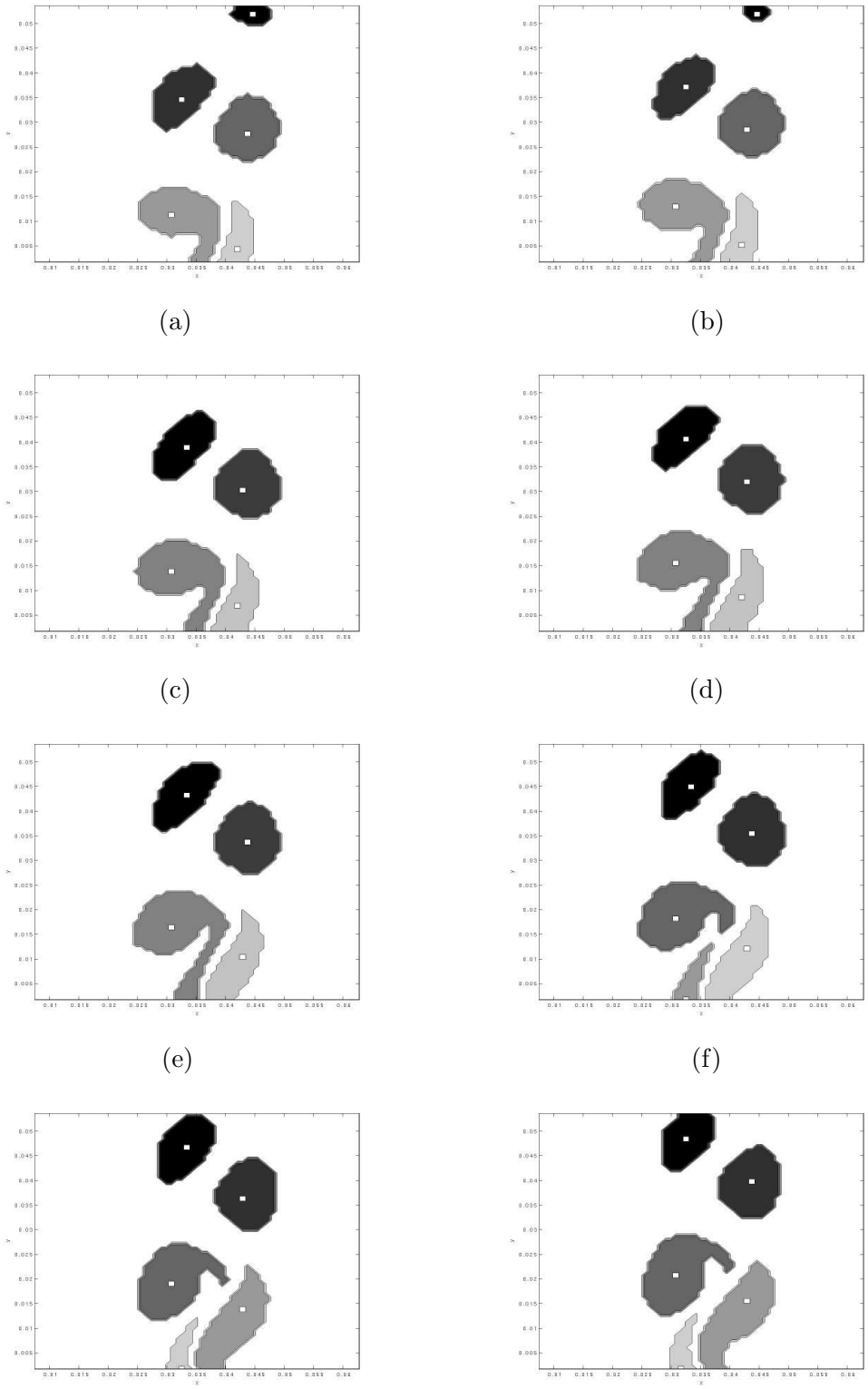
We applied the vorticity thresholding technique described in Section 3.1 to the 17 PIV observations to obtain the point-vortex data sets presented in Figures 13.2 and 13.3. Each of these data sets is an "observation" that can be used to correct the point-vortex model. Since there were 17 observations divided over the jet's 59.4 ms cycle, we had observations available approximately every 3.5 ms. The initial observation in Figure 13.2(a) is used to initialize the model for the assimilation. Notice that there are five vortices present in this observation. However, at timestep 0.0105s, one of the vortices moves downstream from our PIV capture window. Also, at timestep 0.0210s, a new vortex enters the PIV window on the bottom. We had to decide how to handle these types of events in our model. Vortices that leave the PIV capture window remain in our model, but are no longer corrected because they are not observed. It is possible that these vortices dissipate as they move downstream, but dissipation dynamics are not part of the point-vortex model. Also, since the simulation length is so short, it is likely a safe assumption that these vortices remain present in the experiment, even if they decrease in strength as they begin to dissipate in the far-downstream region of the jet. For new vortices that are observed, we simply add a new vortex to the simulation. Thus, although we begin the simulation with five vortices, we eventually include seven because new vortices are introduced at timesteps 0.0210s and 0.0385s.



(a) (b)

(c) (d)

Figure 13.1: PIV measurements of the planar jet at approximately (a) 22.5 ° (b) 90 ° (c) 202.5 ∘ and (d) 292.5 ° into its cycle.



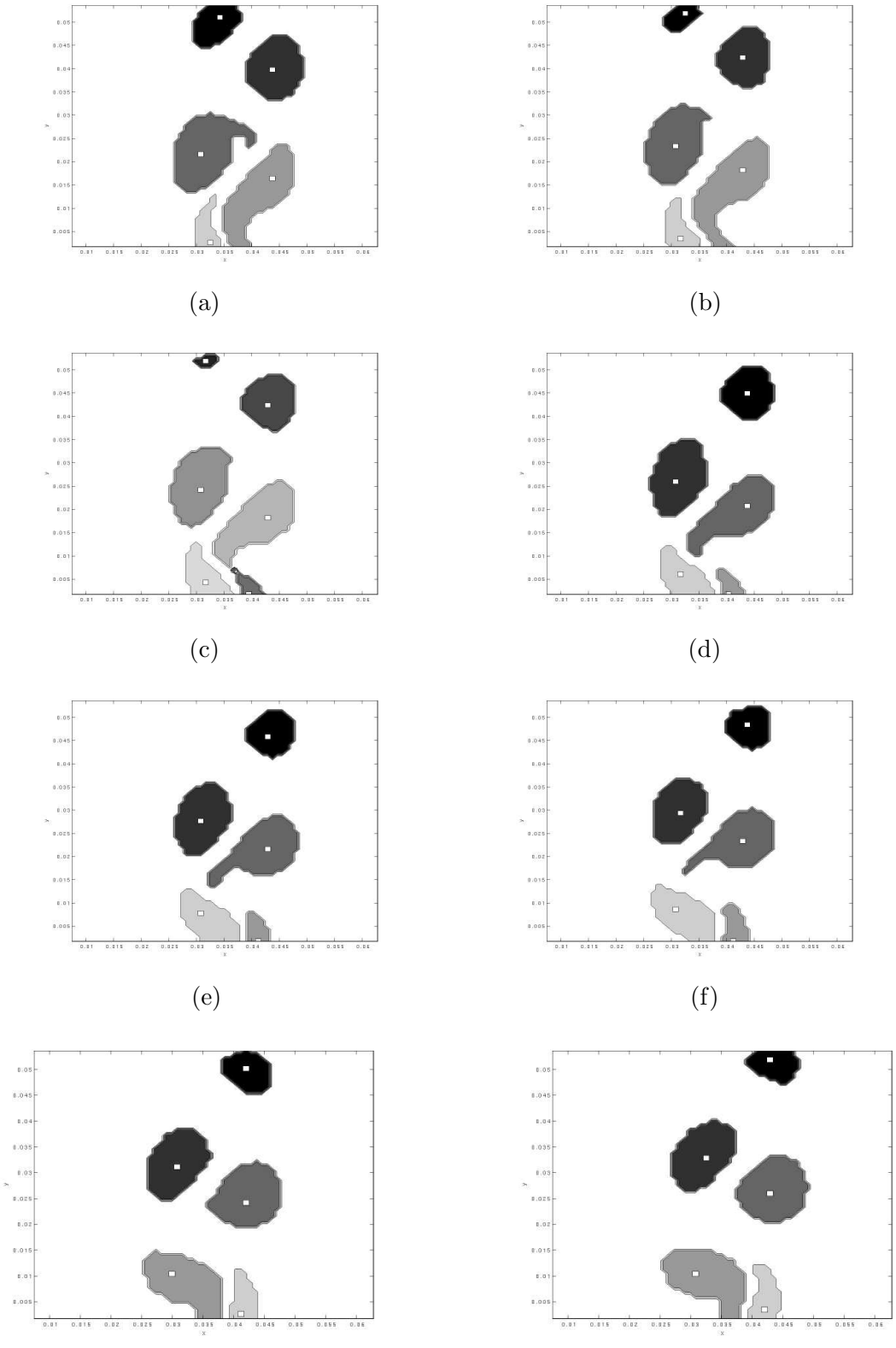
(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 13.2: Vortices extracted from the PIV measurements and used to correct the model at times (a) 0.0035s (b) 0.0070s (c) 0.0105s (d) 0.0140 (e) 0.0175 (f) 0.0210 (g) 0.0245 and (h) 0.0280.



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 13.3: Vortices extracted from the PIV measurements and used to correct the model at times (a) 0.0315s (b) 0.0350s (c) 0.0385s (d) 0.0420 (e) 0.0455 (f) 0.0490 (g) 0.0525 and (h) 0.0560.

Method	**T_c [Min,Max]:Inc**
Norm Change	[0.4, 0.2]:0.001
Norm Thresh	[10, 66]:0.5
Eigenvalues	[10, 60]:0.5
Shear	[9, 28]:0.05
RK Steps	[0, $1E^{-9}$]:$5E^{-12}$

Table 13.1: Thresholds for dynamics-informed techniques. Each entry in this table describes how thresholds were varied for a particular dynamics-informed assimilation experiment. The dynamics-informed technique is listed in the left column and the format of each entry is [minimum Tc, maximum T_c]:T_c increment.

Using these observations and the simulation parameters described above, we ran a collection of experiments using several different correction strategies, including periodic, norm change, norm thresholding, eigenvalue, shear, and Runge-Kutta test step methods. For periodic correction, we ran simulations in which the correction frequency varied from 0.0035s to 0.0630s in increments of 0.0035s. For the dynamics-informed techniques, we used the thresholds displayed in Table 13.1 to generate the set of experiments conducted.

It is difficult to determine the best way to analyze the results of these real-world assimilation experiments. We cannot simply plot our standard MSE versus number of corrections curves for each correction algorithm because we do not know the "truth" in this simulation. We have only our PIV observations of the velocity field, which are inevitably contaminated with noise and uncertainty. The best we can do is to compare our simulations to the observations to see how well each type of simulation reproduces the observations. But, keep in mind that these results do not necessarily translate to how well each simulation reproduces the reality of the physical system. For this analysis, we computed the MSE between the simulation and the observations. Note that this calculation only involved 17 timesteps, because these are the times when observations were available. The results are presented in Figure 13.4. We can see that a periodic correction approach seems to result in the best reproduction of the observations. This

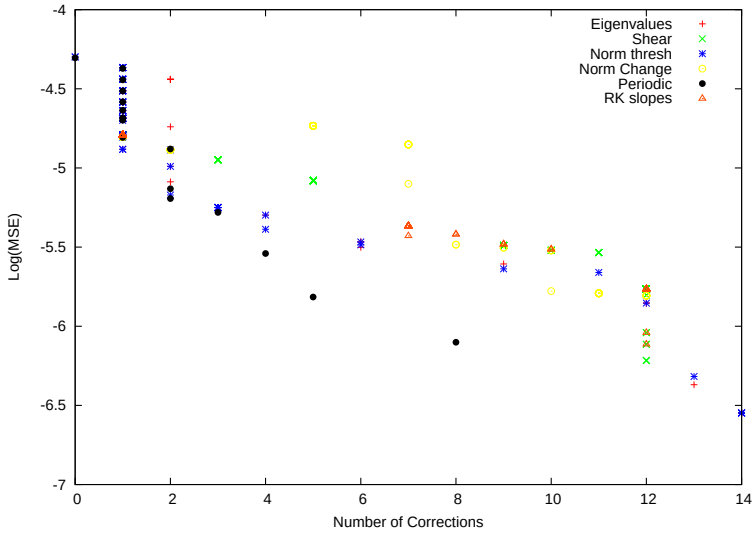
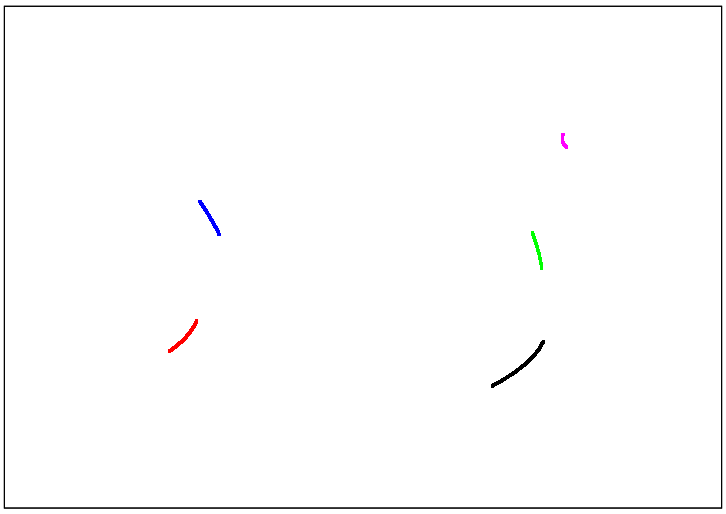


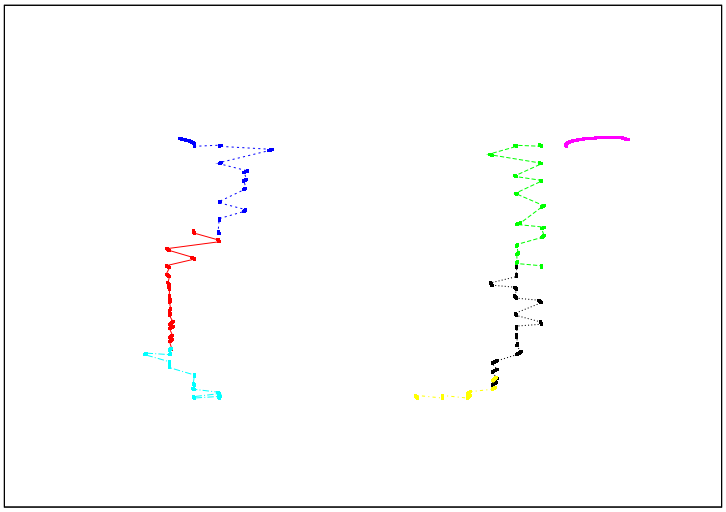
Figure 13.4: Real-World Data Assimilation. Comparison of dynamics-informed and periodic assimilation using the initial conditions in Figure 13.2(a). Each point in this figure represents a single simulation; the MSE between the simulation and the observations is plotted as a function of the number of corrections. Each curve is labelled with the correction strategy used for the assimilation.

is an interesting result, because it is significantly different from what we found in our numerical experiments in Section 12. However, it is important to reiterate that these results only indicate how well the simulation reproduces the observations. So, it is difficult to determine how this correlates with the simulations ability to reproduce the "truth".

We can also perform a qualitative analysis of the vortex trajectories in these simulations. In Figure 13.5(a), we display the vortex trajectories for an uncorrected simulation starting from the initial conditions in Figure 13.2(a); part (b) of this same figure shows a simulation that is corrected whenever observations are available. It is hard to tell conclusively which one of these is a better representation of reality. Clearly in the snapshots displayed in Figures 13.2 and 13.3, we can see that the vortices are slowly moving upward in the y-direction. Both of the simulations in Figure 13.5 show this same general trend. The discontinuous nature of the trajectories in part (b) is due to the assimilation of observations that shift the vortex positions. The point-vortices are jumping back and forth in the x-direction. This unphysical situation is caused by our judgement of the location of each vortex center in the observations in Figure 13.2 and 13.3. A more-sophisticated vortex decomposition scheme in which the point-vortices are distributed to provide a more-realistic representation of the vorticity for each vortex would likely alleviate this problem.



(a)



(b)

Figure 13.5: Point-vortex simulation starting from the initial conditions in Figure 13.2(a). Simulation length in each case was 60 ms with a timestep of 0.0001; (a) was not corrected at all over the course of the simulation and (b) was corrected whenever observations were available, every 3.5 ms.

Chapter 14

Conclusions and Future Work

Real-time fluid modelling and control applications motivate this thesis work. Direct numerical simulation (DNS) techniques are the current state of the art for fluids simulations, but they are usually too slow for these purposes. Several *reduced-order* modelling techniques have been introduced in recent years, including the point-vortex model, which is the focus of our research. So far, the simplifications and approximations necessary to reduce the order of these types of models have compromised their accuracy. Our hope is that correcting the point-vortex model with observations of the physical system using the *data assimilation* techniques introduced in Section 2.2 will sufficiently improve the accuracy without introducing an unmanageable computational burden.

In Section 5, we introduced several *dynamics-informed* targeted observing techniques for the point-vortex model. Our goal was to allow the dynamics of the system to dictate the correction timing for data assimilation. This is in contrast to the traditional data assimilation approach in which observations are assimilated at periodic intervals. Initially, we hoped that one of these dynamics-informed strategies would enable us to improve the accuracy of the point-vortex model at a lower computational cost. The idea was to correct the model only when the correction would have a significant impact in terms of reducing model error, saving on the computational cost of correcting the model when it is not necessary. We found that these dynamics-informed techniques worked well for some data sets and not others, and we have identified some patterns in

the vortex dynamics that indicate when dynamics-informed techniques might fail.

Most of the techniques introduced in Section 5 rely on metrics applied to the induced velocity gradients for each vortex. By analyzing time series of these Jacobian norms, we have developed some preliminary criteria that indicate when dynamics-informed techniques are likely to fail. These results are the culmination of many experiments with five different sets of initial conditions. Initially, we conducted experiments with a simple direct-replacement approach to data assimilation, and we studied the limiting case of noise-free observations. For the von Karman, symmetric, and real data initial conditions from Figures 4.1(a), 4.1(b) and 12.1 respectively, we found that the dynamics-informed techniques significantly outperformed periodic correction. However, for the random and clustered initial conditions from Figures 9.1 and 12.5, the situation was reversed and periodic techniques prevailed. In looking at the norms for these two failing cases, we found that they were characterized by high-frequency, high-amplitude oscillations. Also, the vortex dynamics included several tightly coupled vortices that were orbiting around each other. Our conjecture is that these features make it difficult for our dynamics-informed techniques to pinpoint important time steps, because all of the timesteps have similar dynamics. And, it is likely the case that correcting more often is the best approach.

Clearly these initial conclusions need to be evaluated further in our future work. Also, we would like to try a hybrid correction approach in which periodic corrections are applied in certain types of dynamical regions and dynamics-informed techniques are used in others. Another flavor of hybrid assimilation might be to use the dynamics to vary the periodic correction interval. Hopefully, such techniques might address the difficulties we encountered with dynamics-informed techniques while still enabling us to benefit from their computational savings.

Our second step in the analysis of point-vortex data assimilation with numerical experiments was to add varying levels of Gaussian noise to the assimilated observations.

These results were presented in Section 8 for the von Karman and symmetric data sets. For the von Karman data set, we again discovered some interesting dynamics that resulted in very poor performance of our data assimilation techniques. In particular, the dynamics in later stages of the simulation were "too stable". The result was that after the application of a noisy assimilation early in the simulation, there was nothing to trigger another correction. So, errors accumulated in the incorrect vortex trajectories that resulted. The conclusion in this case was that we needed a more-sophisticated correction algorithm that incorporates information about the observational error when performing the correction. For the symmetric data set, we did not see this type of behavior, because the vortices remain fairly close in the assimilation and the dynamics-informed corrections are triggered throughout the entire simulation. However, we did find that, as the variance in the observational noise increased, periodic correction outperformed dynamics-informed techniques. This is a result that still needs some further analysis in our future work. Additionally, it would be interesting to try other types of noise in the observations to see if the noise type affects the results.

Ensemble Kalman Filtering is one example of a data assimilation algorithm that considers the uncertainty in both the observations and the model when performing the assimilation. The Data Assimilation Research Testbed (DART) is a software framework that enables research with ensemble techniques. We implemented the point-vortex model in DART and conducted some perfect model experiments with the von Karman and symmetric data sets; these results were presented in Section 11. This analysis of ensemble filtering in the point-vortex model is an important contribution of this thesis to the meteorological and geophysical communities. We are the first to use the point-vortex model in DART and verify that ensemble assimilation works well with this model. Additionally, we performed a comparison of the norm change dynamics-informed technique from Section 5 to current state of the art targeted observing technique for ensemble filtering—the spread-based technique described in Section 11. For the von

Karman data set, both the norm change and spread-based correction techniques outperformed periodic correction. And, the assimilation algorithm in DART alleviated the noise issues described in the preceding paragraph. For the symmetric data set, the performance of the spread-based and periodic correction techniques was comparable, but the norm change technique was worse. In our future work, we will conduct some more analysis to determine why the norm change technique does not work well for this data set in the DART framework.

There are also quite a few other areas we would like to address using the DART framework. A more thorough study of several different vortex configurations would enable a better understanding of the dynamics that cause one assimilation timing strategy to outperform another. Also, we have only studied data assimilation of vortex observations; it would be interesting to compare these results to those achievable by assimilating the velocity observations directly. Recall from Section 2.2 that this requires the use of a forward operator to map from the model state space to the observation space. Finally, since one of our interests is ensuring that the computational cost of a simulation that uses the point-vortex model with data assimilation is less than that of a DNS simulation, we need to perform some analysis of the computational costs of using DART for our purposes. This will likely involve some tuning of the DART parameters to achieve the best performance for the lowest cost.

Another significant component of our thesis work was the application of point-vortex data assmilation techniques to real data. Most data assimilation research relies solely on numerical experiments, but we have extended our results to a real-world assimilation of an experimental fluids system. As described in Section 3, we have a controlled laboratory setup of a planar air jet and particle image velocimetry (PIV) system. Our first step in using the PIV data gathered from this system was to develop operational techniques for extracting point-vortex positions and strength from the velocity fields. We employed two standard techniques—vorticity thresholding and Okubo-Weiss—and

augmented them with a connected component algorithm for delineating the vortices. Although this technique is fairly straightforward, we have not seen it used in the vortex extraction literature. We generated one set of observations that assigned one point vortex to each large vortex in the PIV data. After conducting the real assimilation experiment in Section 13, it became clear that we will likely need to distribute more point vortices to more accurately model the vorticity distribution. We explored one technique for doing this in Section 3.3, but the MSE measurements in that section and the numerical experiments in Section 12 seemed to indicate that this technique does not work very well. In future work, we will study other strategies for decomposing the vorticity field into point-vortices, keeping in mind that more point-vortices will increase the computational load for our simulations.

The initial results presented in Section 13 for a real-world assimilation are a useful first step in exploring the practicality of point-vortex data assimilation. In our future work, we would like to run some longer simultions with more obsevations of the system. Also, it might be useful to take some PIV measurements both upstream and downstream to capture more of the dynamics. Once we begin to run longer simulations, it is likely that we will run into problems with the simplicity of our model. In particular, we might need to add boundary conditions that mimic the upstream forcing of the jet and the downstream dissipation of the vortices. Clearly, there are still significant challenges to address before the point-vortex model is practical for real-world modelling of systems like our planar air jet. However, this thesis is a useful first step in analyzing data assimilation in the point-vortex model and determining when dynamics-informed techniques might be useful.

Bibliography

[1] http://www.cgd.ucar.edu/DAI/.

[2] R. Adrian, K. Christiensen, and Z. Liu. Analysis and interpretation of instantaneous turbulent velocity fields. Experiments in Fluids, 29:275–290, 2000.

[3] R.J. Adrian. On the role of conditional averages in turbulence theory. In Proceedings of the Fourth Biennial Symposium on Turbulence in Liquids, volume 44, pages 323–332, Princeton, 1977. Science Press.

[4] A. Airapetov. Motion of an initially point vortex in a flow of viscous fluid. Journal of Applied Mathematics and Mechanics, 54:355–359, 1990.

[5] J. Anderson. A method for producing and evaluating probabilistic forecasts from ensemble model integrations. Journal of Climate, 9:1518–1530, 1996.

[6] J. Anderson. An adaptive covariance inflation error correction algorithm for ensemble filters. Tellus A, 59:210–224, 2007.

[7] Jeffrey L. Anderson. An ensemble adjustment Kalman filter for data assimilation. Monthly Weather Review, 129(12):2884–2903, December 2001.

[8] Jeffrey L. Anderson. A local least squares framework for ensemble filtering. Monthly Weather Review, 131(4):634–642, April 2003.

[9] N.L. Baker. Quality control for the Navy operational atmospheric database. Weather Forecasting, 7:250–261, 1992.

[10] N.L. Baker and R. Daley. Observation and background adjoint sensitivity in the adaptive observation-targeting problem. Quarterly Journal of the Royal Meteorological Society, 126:1431–1454, 2000.

[11] A. Basu, R. Narasimha, and A. Prabhu. Modelling plane mixing layers using vortex points and sheets. Applied Mathematical Modelling, 19:66–75, 1995.

[12] Michael Bergmann, Laurent Cordier, and Jean-Pierre Brancher. Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model. Physics of Fluids, 17:1–21, 2005.

[13] T. Bergot, G. Hello, A. Joly, and S. Malardel. Adaptive observations: A feasibility study. Monthly Weather Review, 127:743–765, 1999.

[14] P. Bergthorsson and B. Doos. Numerical weather map analysis. Tellus, 7:329–340, 1955.

[15] G. Berkooz, P. Holmes, and J. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. Ann. Rev. Fluid Mech., 25:539–575, 1993.

[16] Craig Bishop and Zoltan Toth. Ensemble transformation and adaptive observations. Journal of the Atmospheric Sciences, 56:1748–1765, 1999.

[17] Craig H. Bishop, Brian J. Etherton, and Sharanya J. Majumdar. Adaptive sampling with the ensemble transform Kalman filter. Part I: Theoretical aspects. Monthly Weather Review, 129(3):420–436, March 2001.

[18] P. Boyland, M. Stremler, and H. Aref. Topological fluid mechanics of point vortex motions. Physica D, 175:69–95, 2003.

[19] R. Buizza and T.N. Palmer. The singular-vector structure of the atmospheric global circulation. Journal of the Atmospheric Sciences, 52:1434–1456, 1995.

[20] J.P. Burg and D.G. Luenberger. Estimation of structured covariance matrices. Proceedings of the IEEE, 70:963–974, 1982.

[21] Gerrit Burgers, Peter Jan van Leeuwen, and Geir Evensen. Analysis scheme in the ensemble Kalman filter. Monthly Weather Review, 126(6):1719–1724, June 1998.

[22] R. Camussi. Coherent structure identification from wavelet analysis of particle image velocimetry data. Experiments in Fluids, 32:76–86, 2002.

[23] C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang. Spectral methods in fluid dynamics. Springer, 1988.

[24] P. Chavanis. Kinetic theory of point vortices: Diffusion coefficient and systematic drift. Phys. Rev. E, 64:263091–263092, 2001.

[25] S. Chen, D. Holm, L. Margoin, and R. Zhang. Direct numerical simulations of the Navier-Stokes-α model. Physica D, 133:66–83, 1999.

[26] Y. Chen and C. Snyder. Assimilating vortex position with an ensemble kalman filter. Monthly Weather Review, 135:1828–1845, 2007.

[27] M. Chong, A. Perry, and B. Cantwell. A general classification of three-dimensional flow fields. Physics of Fluids, 2:765–777, 1990.

[28] Stephen E. Cohn. An introduction to estimation theory. Journal of the Meteorological Society of Japan, 75(1B):'257–288, March 1997.

[29] L. Cortelezzi, Y. Chen, and H. Chang. Nonlinear feedback control of the wake past a plate: From a low-order model to a higher-order model. Physics of Fluids, 9:2009–2022, 1997.

[30] P. Courtier, E. Andersson, W. Heckley, J. Pailleux, D. Vasiljevic, M. Hamrud, A. Hollingsworth, F. Rabier, and M. Fisher. The ECMWF implementation of three-dimensional variational assimilation (3D-Var) I: Formulation. Quarterly Journal of the Royal Meteorological Society, 124:1783–1808, 1998.

[31] P. Courtier and O. Talagrand. Variational assimilation of meteorological observations with the adjoint vorticity equation I: Numerical results. Quarterly Journal of the Royal Meteorological Society, 113:1329–1347, 1987.

[32] I. Currie. Fundamental Mechanics of Fluids. McGraw Hill, 1993.

[33] R. Daley. Atmospheric Data Analysis. Cambridge University Press, 1991.

[34] H. Davies and R. Turner. Updating prediction models by dynamical relaxation: An examination of the technique. Quart. J. Roy. Meteor. Soc., 103:225–245, 1977.

[35] A. Deane, I. Kevrekidis, G. Karniadakis, and S. Orszag. Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinders. Physics of Fluids A, 3:2337–2354, 1999.

[36] D. Dee and A. daSilva. Estimating observation error statistics for atmospheric data assimilation. Annales Geophysicae, 11:634–647, 1993.

[37] D. Dee and A. daSilva. Maximum-likelihood estimation of forecast and observation error covariance parameters. Part I: Methodology. Monthly Weather Review, 121:2449–2461, 1998.

[38] J.E. Dennis and J.J. Moré. Quasi-Newton methods, motivation and theory. SIAM Review, 19:46–89, 1977.

[39] John C. Derber. A variational continuous assimilation technique. Monthly Weather Review, 117(11):2437–2446, November 1989.

[40] R. Dickinson and D. Williamson. Free oscillations of a discrete stratified fluid with application to numerical weather prediction. Journal of Atmospheric Science, 29:623–640, 1972.

[41] Francois-Xavier Le Dimet, I.M. Navon, and Dacian N. Daescu. Second-order information in data assimilation. Monthly Weather Review, 130:529–647, 2002.

[42] Francois-Xavier Le Dimet and Olivier Talagrand. Variational algorithms for analysis and assimilation of meteorological observations: Theoretical aspects. Tellus, 38A:97–110, 1986.

[43] E.S. Epstein. Stochastic dynamic prediction. Tellus A, 21:739–759, 1969.

[44] G. Evensen. Using the extended Kalman filter with a multilayer quasi-geostrophic ocean model. Journal of Geophysical Research, 97(C11):17 905–17 924, 1991.

[45] Geir Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. Journal of Geophysical Research, 99(C5):10,143–10,162, May 1994.

[46] M. Farge, K. Schneider, and N. Kevlahan. Non-gaussianity and coherent vortex simulation for two-dimensional turbulence using an adaptive orthogonal wavelet basis. Physics of Fluids, 11(8):2187–2201, August 1999.

[47] M. Farge, K. Schneider, G. Pellegrino, A. Wray, and R. Rogallo. Coherent vortex extraction in three-dimensional homogeneous turbulence: Comparison between CVS-wavelet and POD-Fourier decompositions. Physics of Fluids, 15(10):2886–2896, October 2003.

[48] N. Farrell, N. Ross, T. Peacock, E. Bradley, and J. Hertzberg. Cumetr09092006. Technical report, University of Colorado, 2006.

[49] R.J. Fleming. On stochastic dynamic prediction I: The energetics of uncertainty and the question of closure. Monthly Weather Review, 99:851–872, 1971.

[50] M. Funakoshi. Evolution of vorticity regions of Karman-vortex-street type. Fluid Dynamics Research, 15:251–269, 1995.

[51] L. Gandin. Complex quality control of meteorological observations. Monthly Weather Review, 116:1137–1156, 1988.

[52] P. Gauthier, P. Courtier, and P. Moll. Assimilation of simulated wind LIDAR data with a Kalman filter. Monthly Weather Review, 121:1803–1820, 1993.

[53] A. Gelb, editor. Applied Optimal Estimation. MIT Press, Cambridge, MA, 1974.

[54] J. Gerhard, M. Pastoor, R. King, B.R. Noack, A. Dillman, M. Morzyński, and G. Tadmor. Model-based control of vortex shedding using low-dimensional Galerkin models. In 33rd AIAA Fluids Conference and Exhibit, Orlando, FL, 2003. AIAA Paper 2003-4262.

[55] M. Germano, U. Piomelli, P. Moin, and W. Cabot. A dynamic subgrid scale eddy viscosity model. Phys. Fluids A, 3(7):1760–1765, 1991.

[56] M. Ghil and P. Malanotte-Rizzoli. Data assimilation in meteorology and oceanography. Advanced Geophysics, 33:141–266, 1991.

[57] W. Graham, J. Peraire, and K. Tang. Optimal control of vortex shedding usind low-order models. Part I: Open-loop model development. International Journal of Numerical Methods in Engineering, 44:945–972, 1999.

[58] G. Haller. An objective definition of a vortex. Journal of Fluid Mechanics, 525:1–26, 2005.

[59] T.M. Hamill, J.S. Whitaker, and C. Snyder. Distance-dependent filtering of background error covariance estimates in an ensemble Kalman filter. Monthly Weather Review, 129:2776–2790, 2001.

[60] Ross N. Hoffman. A four-dimensional analysis exactly satisfying equations of motion. Monthly Weather Review, 114:388–397, 1986.

[61] D. Holm, J. Marsden, and T. Ratiu. Euler-Poincaré models of ideal fluids with nonlinear dispersion. Phys. Rev. Lett., 349:4173–4177, 1998.

[62] P.L. Houtekamer, Louis Lefaivre, Jacques Derome, Harold Ritchie, and Herschel L. Mitchell. A system simulation approach to ensemble prediction. Monthly Weather Review, 124(6):1225–1242, June 1996.

[63] P.L. Houtekamer and Herschel L. Mitchell. Data assimilation using an ensemble Kalman filter technique. Monthly Weather Review, 126(3):796–811, March 1998.

[64] P.L. Houtekamer and Herschel L. Mitchell. A sequential ensemble Kalman filter for atmospheric data assimilation. Monthly Weather Review, 129:123–137, 2001.

[65] B. L. Hua, J.C. McWilliams, and P. Klein. Lagrangian accelerations in geostrophic turbulence. Journal of Fluid Mechanics, 366:87–108, 1998.

[66] B.L. Hua and P. Klein. An exact criterion for the stirring properties of nearly two-dimensional turbulence. Physica D, 113:98–110, 1998.

[67] G. Huber and P. Alstrom. Universal decay of vortex density in two dimensions. Physica A, 195:448–456, 1993.

[68] J. Hunt, A. Wray, and P. Moin. Eddies, streams, and convergence zones in turbulent flows. In Proc. of the Summer Program, volume CTR-S88, pages 193–208, 1988. Stanford University Center for Turbulence Research Report.

[69] A K M F Hussain. Coherent structures and turbulence. Journal of Fluid Mechanics, 173:303–356, 1986.

[70] K. Ide, P. Courtier, M. Ghil, and A.C. Lorenc. Unified notation for data assimilation: Operational, sequential, and variational. Journal of the Meteorological Society of Japan, 75(1B):181–189, March 1997.

[71] K. Ide and M. Ghil. Extended Kalman filtering for vortex systems. Part I: Methodology and point vortices. Dynamics of Atmospheres and Oceans, 27:301–332, 1997.

[72] K. Ide, L. Kuznetsov, and C. Jones. Lagrangian data assimilation for point vortex systems. Journal of Turbulence, 3:1–7, 2002.

[73] N.B. Ingleby and A.C. Lorenc. Bayesian quality control using multivariate normal distributions. Quarterly Journal of the Royal Meteorological Society, 119:1195–12251, 1993.

[74] A.H. Jazwinski. Stochastic Processes and Filtering Theory. Academic Press, 1997.

[75] J. Jeong and F. Hussain. On the identification of a vortex. J. Fluid. Mech., 285:69–94, 1995.

[76] B. Jorgensen, J. Sorensen, and M. Brons. Low-dimensional modeling of a driven cavity flos with two free parameters. Theoretical Computational Fluid Dynamics, 16:299–317, 2003.

[77] S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL, 1997.

[78] R.E. Kalman. A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering, 82(Series D):35–45, 1960.

[79] R.E. Kalman and R. Bucy. New results in linear filtering and prediction theory. Transactions of the ASME–Journal of Basic Engineering, 83(Series D):95–108, 1961.

[80] E. Kalnay. Atmospheric Modeling, Data Assimilation, and Predicatability. Cambridge University Press, 2002.

[81] K. Karhunen. Zur spectral theorie stochasticher prozesse. Annales Academiae Scientiarum Fennicae, Mathematica-Physica, 37, 1946.

[82] Christian L. Kepenne. Data assimilation into a primitive-equation model with a parallel ensemble Kalman filter. Monthly Weather Review, 128(6):1971–1981, 2000.

[83] S.P. Khare and J. Anderson. An examination of ensemble filter based adaptive observation methodologies. Tellus A, 58:179–195, 2006.

[84] Sir Horace Lamb. Hydrodynamics. Dover Publications, 1945.

[85] C.E. Leith. Atmospheric predictability and two-dimensional turbulence. Journal of Atmospheric Science, 28:145–161, 1971.

[86] C.E. Leith and R.H. Kraichnan. Predictability of turbulent flows. Journal of Atmospheric Science, 29:1041–1058, 1972.

[87] P.F. Lermusiaux and A. R. Robinson. Data assimilation via error subspaces statistical estimation. Part I: Theory and schemes. Monthly Weather Review, 127:1385–1407, 1999.

[88] M. Lesieur and O. Metais. New trends in large-eddy simulations of turbulence. Ann. Rev. Fluid Mech., 28:45–82, 1996.

[89] J.M. Lewis and J.C. Derber. The use of adjoint equations to solve a variational adjustment problem with advective contraints. Tellus, 37A:309–322, 1985.

[90] M. Loeve. Functiona aleatoire de second ordre. Comptes Rendus Academie des Sciences, Paris, 1945.

[91] A. Lorenc and O. Hammon. Objective quality control of observations using Bayesian methods. theory and practical implementation. Quarterly Journal of the Royal Meteorological Society, 114:515–543, 1988.

[92] A.C. Lorenc. Analysis methods for numerical weather prediction. Quarterly Journal of the Royal Meteorological Society, 112:1551–1556, 1986.

[93] A.C. Lorenc, S.P. Ballard, R.S. Bell, N.B. Ingleby, P.L.F. Andrews, D.M. Barker, J.R. Bray, A.M. Clayton, T. Dalby, D. Li, T.J. Payne, and F.W. Saunders. The Met Office global 3-dimensional variational data assimilation. Quarterly Journal of the Royal Meteorological Society, 126:2991–3012, 2000.

[94] E. Lorenz and K. Emanuel. Optimal sites for supplementary weather observations. Journal of the Atmospheric Sciences, 55:399–414, 1998.

[95] X. Ma and G. Karniadakis. A low-dimensional model for simulating three-dimensional cylinder flow. Journal of Fluid Mechanics, 458:181–190, 2002.

[96] B. Machenhauer. On the dynamics of gravity oscillations in a shallow water model with application to normal mode initialization. Contributions in Atmospheric Physics, 50:253–271, 1977.

[97] R.N. Miller, M. Ghil, and F. Gauthiez. Advanced data assimilation in strongly nonlinear dynamical systems. Journal of Atmospheric Science, 51:1037–1056, 1994.

[98] E. D. Mitchell, S. V. Vasiloff, G. J. Stumpf, A. Witt, M. D. Eilts, J. T. Johnson, and K. W. Thomas. The national severe storms laboratory tornado detection algorithm. Weather and Forecasting, 13:352–366, 1998.

[99] K. Miyakoda and R. Moyer. A method of initialization for dynamical weather forecasting. Tellus, 20:115–128, 1968.

[100] K. Mohseni, B. Kosović, S. Shkoller, and J.E. Marsden. Numerical simulations of the Lagrangian averaged Navier-Stokes (LANS-α) equations for homogeneous isotropic turbulence. Phys. Fluids, 15(2):524–544, 2003.

[101] P. Moin. Progress in large eddy simulation of turbulent flows. Technical Report 1997-0749, AIAA, 1997.

[102] P. Moin and K. Mahesh. Direct numerical simulation: A tool in turbulence research. Annual Review of Fluid Mechanics, 30:539–578, 1998.

[103] R. Morss and K. Emanual. Idealized adaptive observation strategies for improving numerical weather prediction. Journal of the Atmospheric Sciences, 58:210–232, 2001.

[104] T. Nitta and J. Hovermale. A technique of objective analysis and initialization for the primitive forecast equations. Monthly Weather Review, 97:652–658, 1969.

[105] B.R. Noack, K. Afanasiev, M. Morzyński, G. Tadmor, and F. Thiele. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. Journal of Fluid Mechanics, 497:335–363, 2003.

[106] B.R. Noack, G. Tadmor, and M. Morzyński. Actuation models and dissipative control in empirical Galerkin models of fluid flows. In The 2004 American Control Conference, Boston, MA, U.S.A., June 30-July 2, 2004.

[107] A. Okubo. Horizontal dispersion of floatable trajectories in the vicinity of velocity singularities such as convergencies. Deep Sea Research, 17:445–454, 1970.

[108] T. L. Olander and C. S. Velden. The advanced objective dvorak technique (aodt) - continuing the journey. In Conference on Hurricanes and Tropical Meteorology, 26th, Miami, FL, 3-7 May 2004, pages 224–224. Americal Meteorological Society, 2004.

[109] A. Palacios, D. Armbruster, E.J. Kostelich, and E. Stone. Analyzing the dynamics of cellular flames. Physica D, 96(1-4):132–161, 1996.

[110] T. Palmer, R. Gelaro, J. Barkmeijer, and R. Buizza. Singular vectors, metrics, and adaptive observations. Journal of the Atmospheric Sciences, 55:633–653, 1998.

[111] R.L. Panton. Incompressible Flow. John Wiley and Sons, 1996.

[112] David F. Parrish and John C. Derber. The national meteorological center's spectral statistical-interpolation analysis system. Monthly Weather Review, 120(8):1747–1763, August 1992.

[113] T. Peacock, J. Hertzberg, Y.-C. Lee, and E. Bradley. Forcing a planar jet flow using MEMS. Experiments in Fluids, 37:22–28, 2004.

[114] R. Pemberton, S. Turnock, T. Dodd, and E. Rogers. A novel method for identifying vortical structures. Journal of Fluids & Structures, 16:1051–1057, 2002.

[115] D. Peterson and D. Middleton. On representative observations. Tellus, 15:387–405, 1963.

[116] R.W. Preisendorfer. Principal Component Analysis in Meteorology and Oceanography. Elsevier, 1988.

[117] F. Rabier, H. Jarvinen, E. Klinker, J.-F. Mahfouf, and A. Simmons. The ECMWF operational implementation of four-dimensional variational assimilation–Part I: Experimental results with simplified physics. Quarterly Journal of the Royal Meteorological Society, 126:1143–1170, 2000.

[118] M. Raffel, C. Willert, and J. Kompenhans. Particle Image Velocimetry: A Practical Guide. Springer, 1998.

[119] G. Riccardi and R. Piva. Interaction of an elliptical patch with a point vortex. Fluid Dynamics Research, 27:269–289, 2000.

[120] V. Robins, N. Rooney, and E. Bradley. Topology-based signal separation. Chaos, 14:305–316, 2004.

[121] C.W. Rowley and V. Juttijudata. Model-based control and estimation of cavity flow oscillations. In IEEE Conference on Decision and Control, December 2005.

[122] M. Samimy, M. Debiasi, E. Caraballo, J. Malone, J. Little, H. Ozbay, M.O. Efe, P. Yan, X. Yuan, J. DeBonis, J.H. Myatt, and R.C. Camphouse. Three control methods for time-dependent fluid flow. Flow, Turbulence and Combustion, 65:273–298, 2000.

[123] M. Samimy, M. Debiasi, E. Caraballo, J. Malone, J. Little, H. Ozbay, M.O. Efe, P. Yan, X. Yuan, J. DeBonis, J.H. Myatt, and R.C. Camphouse. Exploring strategies for closed-loop cavity flow control. In 42nd AIAA Aerospace Sciences Meeting and Exhibit, January 2004. Stanford University Center for Turbulence Research Report.

[124] Y. Sasaki. Some basic formalisms in numerical variational analysis. Monthly Weather Review, 98:875–883, 1970.

[125] H. Segur. Evolution of a tracer gradient in an incompressible, two-dimensional flow. In IUTAM Symposium on Developments in Geophysical Turbulence, 1998.

[126] A. Seigel and J. Weiss. A wavelet-packet census algorithm for calculating vortex statistics. Physics of Fluids, 9(7):1988–1999, July 1997.

[127] J. Sethian. A brief overview of vortex methods. In Vortex Methods and Vortex Motion. SIAM Press, 1991.

[128] A-M Shineeb, J D Bugg, and R Balachandar. Variable threshold outlier identification in PIV data. Measurement Science and Technology, 15:1722–1732, 2004.

[129] S. Siegel, K. Cohen, and T. McLaughlin. Feedback control of a circular cylinder wake in experiment and simulation. In 33rd AIAA Fluids Conference and Exhibit, Orlando, FL, 2003. AIAA Paper 2003-3571.

[130] L. Sirovich. Turbulence and the dynamics of coherent structures. Quarterly of Applied Mathematics, XLV(3):561–590, 1987.

[131] B.L. Smith and A. Glezer. Jet vectoring using synthetic jets. Journal of Fluid Mechanics, 458:1–34, 2002.

[132] C. Speziale. Analytical methods for the development of Reynolds-stress closures in turbulence. Ann. Rev. Fluid Mech., 23:107–157, 1991.

[133] G. J. Stumpf, A. Witt, E. D. Mitchell, P. L. Spencer, J. T. Johnson, M.D. Eilts, K. W. Thomas, and D. W. Burgess. The national severe storms laboratory mesocyclone detection algorithm for the wsr-88d. Weather and Forecasting, 13:304–326, 1998.

[134] O. Talagrand. Assimilation of observations, an introduction. Journal of the Meteorological Society of Japan, 75(1B):'257–288, March 1997.

[135] O. Talagrand and P. Courtier. Variational assimilation of meteorological observations with the adjoint vorticity equation I: Theory. Quarterly Journal of the Royal Meteorological Society, 113:1311–1328, 1987.

[136] A. Tarantola. Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation. Elsevier, 1987.

[137] C. Temperton. Dynamic initialization for barotropic and multi-level models. Quarterly Journal of the Royal Meteorological Society, 102:297–311, 1976.

[138] H. Vollmers. Detection of vortices and quantitative evaluation of their main parameters from experimental velocity data. Measurement Science and Technology, 12:1199–1207, 2001.

[139] J. Weiss. The dynamics of enstrophy transfer in 2-dimensional hydrodynamics. Physica D, 48:273–294, 1991.

[140] J.S. Whitaker and T.M. Hamill. Ensemble data assimilation without perturbed observations. Monthly Weather Review, 130:1913–1924, 2002.

[141] D. Williamson and R. Dickinson. Free oscillations of the NCAR global circulation model. Monthly Weather Review, 104:1372–1391, 1976.

Appendix A

Point-Vortex model_mod.f90 code for DART

```
! Data Assimilation Research Testbed -- DART
! Copyright 2004, 2005, Data Assimilation Initiative, University Corporation for Atmospheric Research
! Licensed under the GPL -- www.gpl.org/licenses/gpl.html

module model_mod

! <next five lines automatically updated by CVS, do not edit>
! $Source: /home/thoar/CVS.REPOS/DART/models/template/model_mod.f90,v $
! $Revision: 1.3 $
! $Date: 2005/02/26 06:14:24 $
! $Author: thoar $
! $Name: hawaii $

! This is a template showing the interfaces required for a model to be compliant
! with the DART data assimilation infrastructure. The public interfaces listed
! must all be supported with the argument lists as indicated. Many of the interfaces
! are not required for minimal implementation (see the discussion of each
! interface and look for NULL INTERFACE).


! Modules that are absolutely required for use are listed
use        types_mod, only : r8
use time_manager_mod, only : time_type, set_time, write_time, get_time
use     location_mod, only : location_type, set_location, get_location, &
                             LocationDims, LocationName, LocationLName, &
                             get_close_maxdist_init, get_close_obs_init,&
                             get_close_obs, get_dist, query_location,   &
                             operator(==), set_location_missing, &
                             set_location_uninitialized

use    utilities_mod, only : file_exist, open_file, find_namelist_in_file, close_file, &
                             register_module, error_handler, E_ERR, E_MSG, logfileunit, &
                             check_namelist_read
use     obs_kind_mod, only : KIND_V_VELOCITY, KIND_U_VELOCITY, &
                             KIND_VORTEX_X, KIND_VORTEX_Y, KIND_VORTEX_STRENGTH

implicit none
private

public :: get_model_size, &
          adv_1step, &
          get_state_meta_data, &
          model_interpolate, &
          get_model_time_step, &
          end_model, &
          static_init_model, &
          init_time, &
```

```
          init_conditions, &
          model_get_close_states, &
          nc_write_model_atts, &
          nc_write_model_vars, &
          pert_model_state, &
          get_close_maxdist_init, get_close_obs_init, get_close_obs, ens_mean_for_model, &
          comp_gradient_L1_norms, &
          model_corrected

! CVS Generated file description for error handling, do not edit
character(len=128) :: &
source   = "$Source: /home/thoar/CVS.REPOS/DART/models/template/model_mod.f90,v $", &
revision = "$Revision: 1.3 $", &
revdate  = "$Date: 2005/02/26 06:14:24 $"


!================================================================

!  define model parameters

!----------------------------------------------------------------
! Namelist with default values
!
integer  :: numVortices    = 2
real(r8) :: delta_t        = 0.01_r8
integer  :: time_step_days = 1
integer  :: time_step_seconds = 0
real(r8) :: approx_x_min = -5.0_r8
real(r8) :: approx_x_max = 5.0_r8
real(r8) :: approx_y_min = 0.0_r8
real(r8) :: approx_y_max = 200.0_r8

namelist /model_nml/ numVortices, delta_t, time_step_days, time_step_seconds, &
                     approx_x_min, approx_x_max, approx_y_min, approx_y_max
!----------------------------------------------------------------

! Number of state variables
integer :: model_size
integer :: num_corrections

type(location_type), allocatable :: state_loc(:)
logical :: locations_set
type(time_type)     :: time_step, model_time

contains

subroutine static_init_model()
!----------------------------------------------------------------
!
! Called to do one time initialization of the model. As examples,
! might define information about the model size or model timestep.
! In models that require pre-computed static data, for instance
! spherical harmonic weights, these would also be computed here.
! Can be a NULL INTERFACE for the simplest models.

integer  :: vortex, iunit, ierr, io

! Print module information to log file and stdout.
call register_module(source, revision, revdate)

! Read the namelist entry
call find_namelist_in_file("input.nml", "model_nml", iunit)
read(iunit, nml = model_nml, iostat = io)
call check_namelist_read(iunit, io, "model_nml")

! Model size (3 state variables per vortex)
model_size = numVortices * 3.0_r8
```

```
num_corrections = 0

! Create storage for locations
allocate(state_loc(model_size))
locations_set = .false.

! Record the namelist values used for the run ...
call error_handler(E_MSG,'static_init_model','model_nml values are',' ',' ',' ')
write(logfileunit, nml=model_nml)
write(     *     , nml=model_nml)

! open the correction times file
model_time = set_time(0, 0);
open( unit=29, file='./correction_times' )

! The time_step in terms of a time type must also be initialized. Need
! to determine appropriate non-dimensionalization conversion for L93
time_step = set_time(time_step_seconds, time_step_days)

end subroutine static_init_model



subroutine init_conditions(x)
!------------------------------------------------------------------
! subroutine init_conditions(x)
!
! Returns a model state vector, x, that is some sort of appropriate
! initial condition for starting up a long integration of the model.
! At present, this is only used if the namelist parameter
! start_from_restart is set to .false. in the program perfect_model_obs.
! If this option is not to be used in perfect_model_obs, or if no
! synthetic data experiments using perfect_model_obs are planned,
! this can be a NULL INTERFACE.

real(r8), intent(out) :: x(:)

integer  :: i, index
real(r8) :: xPos, yPos, circ

!  read the initial state vector from the file 'initial_conditions'
open(1, file='initial_conditions')

do i = 1, numVortices
   read(1,*) xPos, yPos, circ

   index = 3*(i-1) + 1
   x(index) = xPos
   x(index+1) = yPos
   x(index+2) = circ
end do

close(1)

end subroutine init_conditions


subroutine set_locations(x)
!------------------------------------------------------------------
! subroutine set_locations
!
real(r8), intent(in) :: x(:)

integer  :: index
type(location_type) :: location
```

```
do index=1, model_size-2, 3
   location = set_location( x(index), x(index+1) )
   state_loc(index:index+2) = location
end do

locations_set = .true.

end subroutine set_locations


subroutine model_corrected()
!---------------------------------------------------------------------
! subroutine model_corrected
!
! Keeps track of the number of corrections applied in the assimilation
!
integer :: seconds, days

num_corrections = num_corrections + 1

call get_time( model_time, seconds, days )
write( 29, * ) days*delta_t

end subroutine model_corrected


subroutine adv_1step(x, time)
!---------------------------------------------------------------------
! subroutine adv_1step(x, time)
!
! Does single time step advance for the point-vortex model
! using fourth order Runge Kutta integration

real(r8), intent(inout) :: x(:)
type(time_type), intent(in) :: time

real(r8), dimension(size(x)) :: k1, k2, k3, k4, inter
real(r8) :: half_time
integer :: i

model_time = time;

half_time = delta_t / 2.0_r8

call comp_dx(x, k1)        !  Compute the first intermediate step
inter = x + (half_time * k1);

call comp_dx(inter, k2)    !  Compute the second intermediate step
inter = x + (half_time * k2);

call comp_dx(inter, k3)    !  Compute the third intermediate step
inter = x + (delta_t * k3)

call comp_dx(inter, k4)    !  Compute fourth intermediate step

!  Compute new value for x

x = x + delta_t/6.0_r8 * (k1 + 2.0_r8 * k2 + 2.0_r8 * k3 + k4 )

!print *, "After Advance"
!do i = 1, model_size
!   print *, x(i)
!end do

end subroutine adv_1step
```

```
subroutine comp_gradient_L1_norms( x, norms, dx, dy )
!
! subroutine comp_gradient_L1_norms
!
! Computes the L1 norm of the local velocity gradients
! for each vortex
!
real(r8), intent(in) :: x(:)
real(r8), intent(out) :: norms(:)
real(r8), intent(in) :: dx, dy

real(r8), dimension(model_size) :: d_dx, d_dy
real(r8) :: sum1, sum2
integer :: vortex

call comp_gradients( x, d_dx, d_dy, dx, dy )

! Compute the column sum norm of the matrix
!
! du/dx du/dy
! dv/dx dv/dy
!
do vortex = 1, model_size-2, 3
   sum1 = abs(d_dx(vortex)) + abs(d_dx(vortex+1))
   sum2 = abs(d_dy(vortex)) + abs(d_dy(vortex+1))

!write(*,*) "du_dx, dv_dx, du_dy, dv_dy", d_dx(vortex),d_dx(vortex+1), d_dy(vortex), d_dy(vortex+1)
!write(*,*) "sum1=", sum1, "sum2=", sum2

   if( sum1 > sum2 ) then
      norms((vortex+2)/3) = sum1
   else
      norms((vortex+2)/3) = sum2
   end if
end do

end subroutine comp_gradient_L1_norms



subroutine comp_gradients( x, d_dx, d_dy, dx, dy )
!
! subroutine comp_gradients
!
! Computes the local velocity gradients in the vicinity of
! each vortex
!
real(r8), intent(in) :: x(:)
real(r8), intent(out) :: d_dx(:), d_dy(:)
real(r8), intent(in) :: dx, dy

real(r8), dimension(model_size) :: temp

! compute velocity gradient in the x direction first
call comp_dx_with_offset( x, d_dx, dx, 0.0_r8 )
call comp_dx_with_offset( x, temp, -dx, 0.0_r8 )

d_dx = (d_dx - temp)/(2.0_r8*dx)

call comp_dx_with_offset( x, d_dy, 0.0_r8, dy )
call comp_dx_with_offset( x, temp, 0.0_r8, -dy )

d_dy = (d_dy - temp)/(2.0_r8*dy)

end subroutine comp_gradients
```

```
subroutine comp_dx(x, deriv)
!
! subroutine comp_dx(x, deriv)
!
! Computes the dx/dt for the point vortex model
!
real(r8), intent(in) :: x(:)
real(r8), intent(out) :: deriv(:)

call comp_dx_with_offset(x, deriv, 0.0_r8, 0.0_r8)

end subroutine comp_dx



subroutine comp_dx_with_offset(x, deriv, dx, dy)
!
! subroutine comp_dx_with_offset(x, deriv, dx, dy)
!
! Computes the dx/dt for the point vortex model.
! Adds an offset to the reference vortex position

real(r8), intent( in) :: x(:)
real(r8), intent(out) :: deriv(:)
real(r8), intent(in) :: dx, dy

integer :: reference
type(location_type) :: ref_loc
real(r8), dimension(2) :: induced_velocity

! perform the calculations for each vortex in the system
do reference = 1, model_size-2, 3
   ref_loc = set_location( x(reference), x(reference+1) )
   induced_velocity = find_induced_velocity( x, ref_loc, dx, dy, .true. )

   deriv(reference) = induced_velocity(1)
   deriv(reference+1) = induced_velocity(2)

   ! vorticity remains constant in 2D
   deriv(reference+2) = 0
end do

!   print *, "Deriv", reference, other
!   do i=1,model_size
!      print *, deriv(i)
!   end do


end subroutine comp_dx_with_offset


function find_induced_velocity( state, ref_loc, dx, dy, skip_colocated )
!-------------------------------------------------------------
! subroutine find_induced_velocity()
!
! Finds the induced velocity at a given location
!
real(r8), intent(in) :: state(:)
type(location_type), intent(in) :: ref_loc
real(r8), intent(in) :: dx, dy
logical, intent(in) :: skip_colocated

real(r8), dimension(2) :: find_induced_velocity
real(r8), dimension(2) :: kernel_result
real(r8):: xInfl, yInfl
```

```
real(r8):: ref_x, ref_y
integer :: other
type(location_type) :: other_loc

xInfl = 0.0_r8
yInfl = 0.0_r8

do other = 1, model_size-2, 3
    other_loc = set_location( state(other), state(other+1) )
    if (skip_colocated .AND. other_loc == ref_loc) cycle

    ref_x = query_location( ref_loc, 'x' )
    ref_y = query_location( ref_loc, 'y' )

    kernel_result = kernel_with_offset( state, ref_x, ref_y, state(other), &
                                        state(other+1), dx, dy )

    ! multiply the kernel function times the
    ! vorticity of the other vortex
    xInfl = xInfl + (kernel_result(1) * state(other+2))
    yInfl = yInfl + (kernel_result(2) * state(other+2))
end do

find_induced_velocity(1) = xInfl
find_induced_velocity(2) = yInfl

end function find_induced_velocity


function kernel(state, ref_x, ref_y, other_x, other_y)
!-----------------------------------------------------------------
! function kernel(state, ref_x, ref_y, other_x, other_y)
!
! Kernel function for point vortex integration
!
real(r8), intent(in) :: state(:)
real(r8), intent(in) :: ref_x, ref_y, other_x, other_y

real(r8), dimension(2) :: kernel

kernel = kernel_with_offset( state, ref_x, ref_y, other_x, &
                             other_y, 0.0_r8, 0.0_r8 )

end function kernel



function kernel_with_offset(state, ref_x, ref_y, other_x, other_y, dx, dy)
!-----------------------------------------------------------------
! function kernel_with_offset(state, ref_x, ref_y, other_x, other_y, dx, dy)
!
!  Kernel function for point vortex integration
!  Adds an x offset and y offset to the reference vortex position
!  before computing the kernel
!
real(r8), intent(   in) :: state(:)
real(r8), intent(   in) :: ref_x, ref_y, other_x, other_y
real(r8), intent(in) :: dx, dy

real(r8) :: x, y, normalization, temp
real(r8), parameter :: pi=3.14159265358979323846264338327 95

real(r8), dimension(2) :: kernel_with_offset

x = ref_x+dx - other_x
y = ref_y+dy - other_y
```

```
! Kernel function
! K(x) = 1/2pi * (-x2,x1)/(|x|^2)

! first compute |x|^2
normalization = x*x + y*y;

! multiply by 2pi
normalization = normalization * 2*pi;

! perform swap for (-x2, x1) and div by normalization
temp = x;
x = -y/normalization;
y = temp/normalization;

kernel_with_offset(1) = x;
kernel_with_offset(2) = y;

end function kernel_with_offset


function get_model_size()
!----------------------------------------------------------------------
!
! Returns the size of the model as an integer. Required for all
! applications.

integer :: get_model_size

get_model_size = model_size

end function get_model_size



subroutine init_time(time)
!----------------------------------------------------------------------
!
!! Companion interface to init_conditions. Returns a time that is somehow
! appropriate for starting up a long integration of the model.
! At present, this is only used if the namelist parameter
! start_from_restart is set to .false. in the program perfect_model_obs.
! If this option is not to be used in perfect_model_obs, or if no
! synthetic data experiments using perfect_model_obs are planned,
! this can be a NULL INTERFACE.

type(time_type), intent(out) :: time

! For now, just set to 0
time = set_time(0, 0)


end subroutine init_time



subroutine model_interpolate(x, location, itype, obs_val, istatus)
!----------------------------------------------------------------------
!
! Given a state vector, a location, and a model state variable type,
! interpolates the state variable field to that location and returns
! the value in obs_val. The istatus variable should be returned as
! 0 unless there is some problem in computing the interpolation in
! which case an alternate value should be returned. The itype variable
! is a model specific integer that specifies the type of field (for
! instance temperature, zonal wind component, etc.). In low order
! models that have no notion of types of variables, this argument can
! be ignored. For applications in which only perfect model experiments
```

```
! with identity observations (i.e. only the value of a particular
! state variable is observerd), this can be a NULL INTERFACE.

real(r8),            intent(in) :: x(:)
type(location_type), intent(in) :: location
integer,             intent(in) :: itype
real(r8),           intent(out) :: obs_val
integer,            intent(out) :: istatus

real(r8), dimension(2) :: induced_velocity

! Default for successful return
istatus = 0

induced_velocity = find_induced_velocity( x, location, 0.0_r8, &
                                          0.0_r8, .false. )

select case(itype)
    case (KIND_U_VELOCITY)
        obs_val = induced_velocity(1)
    case (KIND_V_VELOCITY)
        obs_val = induced_velocity(2)
    case DEFAULT
        call error_handler( E_ERR, 'model_interpolate', &
                            'Attempt to interpolate invalid variable type', &
                            source, revision, revdate )
end select

end subroutine model_interpolate



function get_model_time_step()
!----------------------------------------------------------------------
!
! Returns the the time step of the model; the smallest increment
! in time that the model is capable of advancing the state in a given
! implementation. This interface is required for all applications.

type(time_type) :: get_model_time_step

get_model_time_step = time_step

end function get_model_time_step



subroutine get_state_meta_data(index_in, location, var_type)
!----------------------------------------------------------------------
!
! Given an integer index into the state vector structure, returns the
! associated location. A second intent(out) optional argument kind
! can be returned if the model has more than one type of field (for
! instance temperature and zonal wind component). This interface is
! required for all filter applications as it is required for computing
! the distance between observations and state variables.
integer,             intent(in)  :: index_in
type(location_type), intent(out) :: location
integer,             intent(out), optional :: var_type

integer :: vortex
integer :: modulus

if (.NOT. locations_set) then
   write(*,*) 'Locations not set for the point vortex model'
   location = set_location_uninitialized()
end if
```

```
location = state_loc(index_in)

if( present(var_type) ) then
   modulus = MOD(index_in, 3)
   select case( modulus )
      case(1)
         var_type = KIND_VORTEX_X
      case(2)
         var_type = KIND_VORTEX_Y
      case(3)
         var_type = KIND_VORTEX_STRENGTH
   end select
end if

end subroutine get_state_meta_data



subroutine end_model()
!------------------------------------------------------------------
!
! Does any shutdown and clean-up needed for model. Can be a NULL
! INTERFACE if the model has no need to clean up storage, etc.

deallocate(state_loc)

write(*,*) "Writing number of corrections to file...", num_corrections
open( unit=1, file='./num_corrections' )
write(1, *) num_corrections
close(1)

write(*,*) "Closing correction times file..."
close(29)
write(*,*) "Done."

end subroutine end_model



subroutine model_get_close_states(o_loc, radius, inum, indices, dist, x)
!------------------------------------------------------------------
!
! Computes a list of model state variable indices that are within
! distance radius of a given location, o_loc. The units of the radius
! and the metric for computing distances is defined by the location module
! that is in  use. The number of state variables that are within radius
! of o_loc is returned in inum. The indices of each of these is
! returned in indices and the corresponding distance in dist. The model
! state is available in x because it is sometimes required to determine
! the distance (for instance, the current model surface pressure field
! is required to compute the location of state variables in a sigma
! vertical coordinate model). A model can choose to do no computation
! here and return a value of -1 in inum. If this happens, the filter
! will do a naive search through ALL state variables for close states.
! This can work fine in low-order models, but can be far too expensive
! in large models.

type(location_type), intent(in) :: o_loc
real(r8), intent(in) :: radius
integer, intent(out) :: inum, indices(:)
real(r8), intent(out) :: dist(:)
real(r8), intent(in) :: x(:)

type(location_type) :: vortex_loc
real(r8) :: distance
integer :: index, vortex
```

```
index = 1
do vortex=1, model_size-2, 3
    vortex_loc = set_location( x(vortex), x(vortex+1) )

    ! We don't really know the kind of the incoming location,
    ! but it currently doesn't matter in the distance computation
    distance = get_dist( o_loc, vortex_loc, KIND_VORTEX_X, KIND_VORTEX_X )
    if ( distance < radius ) then
       indices(index) = vortex
       indices(index+1) = vortex+1
       indices(index+2) = vortex+2
       dist(index:index+2) = distance
       index = index + 3
    end if
end do
inum = index-1

end subroutine model_get_close_states



function nc_write_model_atts( ncFileID ) result (ierr)
!-----------------------------------------------------------------
! Writes the model-specific attributes to a netCDF file
! TJH Jan 24 2003
!
! TJH 29 July 2003 -- for the moment, all errors are fatal, so the
! return code is always '0 == normal', since the fatal errors stop execution.
!
! For the lorenz_96 model, each state variable is at a separate location.
! that's all the model-specific attributes I can think of ...
!
! assim_model_mod:init_diag_output uses information from the location_mod
!     to define the location dimension and variable ID. All we need to do
!     is query, verify, and fill ...
!
! Typical sequence for adding new dimensions,variables,attributes:
! NF90_OPEN             ! open existing netCDF dataset
!    NF90_redef         ! put into define mode
!    NF90_def_dim       ! define additional dimensions (if any)
!    NF90_def_var       ! define variables: from name, type, and dims
!    NF90_put_att       ! assign attribute values
! NF90_ENDDEF           ! end definitions: leave define mode
!    NF90_put_var       ! provide values for variable
! NF90_CLOSE            ! close: save updated netCDF dataset

use typeSizes
use netcdf

integer, intent(in)  :: ncFileID      ! netCDF file identifier
integer              :: ierr          ! return value of function

!-----------------------------------------------------------------------
! General netCDF variables
!-----------------------------------------------------------------------

integer :: nDimensions, nVariables, nAttributes, unlimitedDimID

!-----------------------------------------------------------------------
! netCDF variables for Location
!-----------------------------------------------------------------------

integer :: LocationVarID
integer :: StateVarDimID, StateVarVarID
integer :: StateVarID, MemberDimID, TimeDimID
```

```
!----------------------------------------------------------------------
! local variables
!----------------------------------------------------------------------

character(len=8)      :: crdate      ! needed by F90 DATE_AND_TIME intrinsic
character(len=10)     :: crtime      ! needed by F90 DATE_AND_TIME intrinsic
character(len=5)      :: crzone      ! needed by F90 DATE_AND_TIME intrinsic
integer, dimension(8) :: values      ! needed by F90 DATE_AND_TIME intrinsic
character(len=NF90_MAX_NAME) :: str1

integer             :: i
type(location_type) :: lctn
ierr = 0                     ! assume normal termination

!----------------------------------------------------------------------
! make sure ncFileID refers to an open netCDF file
!----------------------------------------------------------------------

call check(nf90_Inquire(ncFileID, nDimensions, nVariables, nAttributes, unlimitedDimID))
call check(nf90_sync(ncFileID)) ! Ensure netCDF file is current
call check(nf90_Redef(ncFileID))

!----------------------------------------------------------------------
! Determine ID's from stuff already in the netCDF file
!----------------------------------------------------------------------

! make sure time is unlimited dimid

call check(nf90_inq_dimid(ncFileID,"copy",dimid=MemberDimID))
call check(nf90_inq_dimid(ncFileID,"time",dimid=TimeDimID))

!----------------------------------------------------------------------
! Write Global Attributes
!----------------------------------------------------------------------
call DATE_AND_TIME(crdate,crtime,crzone,values)
write(str1,'(''YYYY MM DD HH MM SS = '',i4,5(1x,i2.2))') &
                  values(1), values(2), values(3), values(5), values(6), values(7)

call check(nf90_put_att(ncFileID, NF90_GLOBAL, "creation_date",str1))
call check(nf90_put_att(ncFileID, NF90_GLOBAL, "model_source", source ))
call check(nf90_put_att(ncFileID, NF90_GLOBAL, "model_revision", revision ))
call check(nf90_put_att(ncFileID, NF90_GLOBAL, "model_revdate", revdate ))
call check(nf90_put_att(ncFileID, NF90_GLOBAL, "model", "PointVortex"))
call check(nf90_put_att(ncFileID, NF90_GLOBAL, "model_numVorts", numVortices ))
call check(nf90_put_att(ncFileID, NF90_GLOBAL, "model_delta_t", delta_t ))


!----------------------------------------------------------------------
! Define the model size, state variable dimension ... whatever ...
!----------------------------------------------------------------------

call check(nf90_def_dim(ncid=ncFileID, name="StateVariable", &
                        len=model_size, dimid = StateVarDimID))

!----------------------------------------------------------------------
! Define the Location Variable and add Attributes
! Some of the atts come from location_mod (via the USE: stmnt)
! CF standards for Locations:
! http://www.cgd.ucar.edu/cms/eaton/netcdf/CF-working.html#ctype
!----------------------------------------------------------------------

!call check(NF90_def_var(ncFileID, name=trim(adjustl(LocationName)), xtype=nf90_double, &
!              dimids = StateVarDimID, varid=LocationVarID) )
!call check(nf90_put_att(ncFileID, LocationVarID, "long_name", trim(adjustl(LocationLName))))
!call check(nf90_put_att(ncFileID, LocationVarID, "dimension", LocationDims ))
!call check(nf90_put_att(ncFileID, LocationVarID, "units", "nondimensional"))
```

```
!call check(nf90_put_att(ncFileID, LocationVarID, "valid_range", (/ 0.0_r8, 1.0_r8 /)))

!----------------------------------------------------------------
! Define either the "state vector" variables -OR- the "prognostic" variables.
!----------------------------------------------------------------

! Define the state vector coordinate variable
call check(nf90_def_var(ncid=ncFileID,name="StateVariable", xtype=nf90_int, &
           dimids=StateVarDimID, varid=StateVarVarID))
call check(nf90_put_att(ncFileID, StateVarVarID, "long_name", "State Variable ID"))
call check(nf90_put_att(ncFileID, StateVarVarID, "units",     "indexical") )
call check(nf90_put_att(ncFileID, StateVarVarID, "valid_range", (/ 1, model_size /)))


! Define the actual state vector
call check(nf90_def_var(ncid=ncFileID, name="state", xtype=nf90_double, &
           dimids = (/ StateVarDimID, MemberDimID, TimeDimID /), varid=StateVarID))
call check(nf90_put_att(ncFileID, StateVarID, "long_name", "model state or fcopy"))

! Leave define mode so we can fill
call check(nf90_enddef(ncfileID))

! Fill the state variable coordinate variable
call check(nf90_put_var(ncFileID, StateVarVarID, (/ (i,i=1,model_size) /) ))

!----------------------------------------------------------------
! Fill the location variable
!----------------------------------------------------------------

! do i = 1,model_size
!   call get_state_meta_data(i,lctn)
!   call check(nf90_put_var(ncFileID, LocationVarID, get_location(lctn), (/ i /) ))
!enddo

!----------------------------------------------------------------
! Flush the buffer and leave netCDF file open
!----------------------------------------------------------------
call check(nf90_sync(ncFileID))

write (*,*)'Model attributes written, netCDF file synched ...'

contains
   ! Internal subroutine - checks error status after each netcdf, prints
   !                       text message each time an error code is returned.
   subroutine check(istatus)
      integer, intent ( in) :: istatus
      if(istatus /= nf90_noerr) call error_handler(E_ERR,'nc_write_model_atts',&
         trim(nf90_strerror(istatus)), source, revision, revdate)
   end subroutine check

end function nc_write_model_atts



function nc_write_model_vars( ncFileID, statevec, copyindex, timeindex ) result (ierr)
!----------------------------------------------------------------
! Writes the model variables to a netCDF file
! TJH 23 May 2003
!
! TJH 29 July 2003 -- for the moment, all errors are fatal, so the
! return code is always '0 == normal', since the fatal errors stop execution.
!
! For the lorenz_96 model, each state variable is at a separate location.
! that's all the model-specific attributes I can think of ...
!
! assim_model_mod:init_diag_output uses information from the location_mod
!     to define the location dimension and variable ID. All we need to do
```

```
!     is query, verify, and fill ...
!
! Typical sequence for adding new dimensions,variables,attributes:
! NF90_OPEN             ! open existing netCDF dataset
!    NF90_redef         ! put into define mode
!    NF90_def_dim       ! define additional dimensions (if any)
!    NF90_def_var       ! define variables: from name, type, and dims
!    NF90_put_att       ! assign attribute values
! NF90_ENDDEF           ! end definitions: leave define mode
!    NF90_put_var       ! provide values for variable
! NF90_CLOSE            ! close: save updated netCDF dataset

use typeSizes
use netcdf

integer,                intent(in) :: ncFileID      ! netCDF file identifier
real(r8), dimension(:), intent(in) :: statevec
integer,                intent(in) :: copyindex
integer,                intent(in) :: timeindex
integer                            :: ierr          ! return value of function

!--------------------------------------------------------------------
! General netCDF variables
!--------------------------------------------------------------------

integer :: nDimensions, nVariables, nAttributes, unlimitedDimID
integer :: StateVarID

!--------------------------------------------------------------------
! local variables
!--------------------------------------------------------------------

ierr = 0                     ! assume normal termination

!--------------------------------------------------------------------
! make sure ncFileID refers to an open netCDF file
!--------------------------------------------------------------------

call check(nf90_Inquire(ncFileID, nDimensions, nVariables, nAttributes, unlimitedDimID))

! no matter the value of "output_state_vector", we only do one thing.

call check(NF90_inq_varid(ncFileID, "state", StateVarID) )
call check(NF90_put_var(ncFileID, StateVarID, statevec,  &
            start=(/ 1, copyindex, timeindex /)))

! write (*,*)'Finished filling variables ...'
call check(nf90_sync(ncFileID))
! write (*,*)'netCDF file is synched ...'

contains
   ! Internal subroutine - checks error status after each netcdf, prints
   !                       text message each time an error code is returned.
   subroutine check(istatus)
   integer, intent ( in) :: istatus
      if(istatus /= nf90_noerr) call error_handler(E_ERR,'nc_write_model_vars',&
         trim(nf90_strerror(istatus)), source, revision, revdate)
   end subroutine check

end function nc_write_model_vars



subroutine pert_model_state(state, pert_state, interf_provided)
!------------------------------------------------------------------
!
! Perturbs a model state for generating initial ensembles.
```

```
! The perturbed state is returned in pert_state.
! A model may choose to provide a NULL INTERFACE by returning
! .false. for the interf_provided argument. This indicates to
! the filter that if it needs to generate perturbed states, it
! may do so by adding an O(0.1) magnitude perturbation to each
! model state variable independently. The interf_provided argument
! should be returned as .true. if the model wants to do its own
! perturbing of states.

real(r8), intent(in)  :: state(:)
real(r8), intent(out) :: pert_state(:)
logical,  intent(out) :: interf_provided

interf_provided = .false.

! May want to do something more intelligent here later...

end subroutine pert_model_state


subroutine ens_mean_for_model(ens_mean)
!------------------------------------------------------------------
! Not used in low-order models

real(r8), intent(in) :: ens_mean(:)

call set_locations(ens_mean)

end subroutine ens_mean_for_model



!============================================================================
! End of model_mod
!============================================================================
end module model_mod
```