UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

KALMAN FILTER BASED TECHNIQUES FOR ASSIMILATION OF RADAR DATA

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

NUSRAT YUSSOUF
Norman, Oklahoma
2010

KALMAN FILTER BASED TECHNIQUES FOR ASSIMILATION OF RADAR DATA

A DISSERTATION APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Dr. S. Lakshmivarahan, Chair

Dr. Sudarshan Dhall

Dr. John K. Antonio

Dr. Thordur Runolfsson

Dr. David J. Stensrud

Acknowledgements

First of all, I thank my employer Cooperative Institute for Mesoscale Meteorological Studies, University of Oklahoma, and NOAA/National Severe Storms Laboratory (CIMMS/OU/NSSL) for supporting me to pursue a Ph.D degree as a "part-time" student while still working full-time.

I thank my graduate research advisor, Dr. S. Lakshmivarahan for his continuous encouragement and support on my path toward a Ph.D. Professor Varahan spent many hours of his time teaching me several techniques on data assimilation that make up the foundation of this work. His Dynamic Data Assimilation book was also very helpful for this research endeavor.

I want to thank my doctoral committee members, Dr. John Antonio, Dr. Sudarshan Dhall, Dr. Thordur Runolfsson, and Dr. David Stensrud for serving on my committee. I am deeply grateful to Dr. Stensrud for his guidance, patience and support throughout this process. He taught me how to conduct research and write manuscripts.

My deepest thanks to my husband, Zahed Siddique and children, Raiyan and Ridwan. Zahed has always been very helpful and this work would not have been possible without his support. Thanks to my parents, Nargis Akhter and M. M. Yussouf for their unconditional love and encouragement.

Finally, and most importantly, I thank Almighty Allah for bringing me this far by His grace.

Table of Contents

List of Tables

List of Figures

Abstract

The Ensemble Square Root Filter (EnSRF) data assimilation technique is applied to examine the impact of assimilating high temporal frequency radar observations over a shorter assimilation period. To reduce the heavy computation cost of assimilating large number of radar observations using EnSRF technique, synthetic radar observations are generated at coarser spatial resolution. Two sets of experiment are conducted with identical settings based on perfect model framework where model error does not play a role. One experiment assimilates radar observations, in which a volume scan is conducted every 5 min, while the other experiment assimilates observations, in which a volume scan is conducted every 1 min. Results indicate that assimilating observations at 1-min intervals over short 15-min period yields significantly better analyses and forecasts than those produced using observations at 5-min intervals. However, the very good performance obtained from perfect model experiments is not expected in real-world experiments where models unavoidably have errors. Therefore to account for model error, another two sets of experiments are conducted using both a perfect and an imperfect model framework and the EnSRF data assimilation technique. In addition, the value of using a range of intercept and density parameters for hydrometeor categories in different ensemble members within the same microphysics scheme also is examined. Results show that the EnSRF system performs reasonably well with the imperfect model assumption. Results also indicate that in the presence of model error, a combination of different hydrometeor density and intercept parameters leads to improved forecasts over experiments that use a constant, hydrometeor intercept and density parameter.

While the EnSRF data assimilation technique shows promise in radar data assimilation, one limitation of EnSRF technique is that it assimilates observations serially, making it computationally very expensive when the number of observations is very large. Thus in an effort to explore efficient data assimilation method, the feasibility of the information filter as an alternate to the EnSRF data assimilation technique when the number of observations is very large is examined. The extended information filter (EIF) is implemented using the Lorenz 96 model and the performance of EIF in assimilating both low and high spatial resolution observations are compared with the benchmark extended Kalman filter (EKF) assimilation technique. Results indicate that both EKF and EIF produce similar results for different spatial resolution observation assimilation. The computational time for the EIF is larger than that of the EKF filter as expected due to the higher computational cost of matrix inversion in EIF technique. However, the increment in computational cost for EIF technique is much smaller than that of EKF technique for increased number of observation assimilation.

Chapter 1

Introduction

1.1 Background and Motivation

The numerical weather prediction (NWP) models are used in meteorology to study a variety of atmospheric processes and to predict the future atmospheric states. The NWP models represents the atmosphere in a three-dimensional grids with n_x, n_y and n_z points in the *x, y* and *z* directions giving rise to $n_g = n_x n_y n_z$ grid points. Moreover, at each grid point there are L physical variables, such as pressure, temperature, wind in three dimension, moisture etc., that must be represented. For a small grid of $n_x=100$, $n_y=100$ and $n_z=50$ points, there is a total of $n_g=5x10^5$ grid points, and thus a total of $n = n_g L$ variables that must be defined in a model. The prediction of future atmospheric states is accomplished as an initial value problem where the "best" initial atmospheric state using dynamic data assimilation techniques. After the initialization, the three-dimensional NWP models are integrated forward in time to make a prediction. Thus the challenge of data assimilation is to find an estimate of the initial atmospheric state based upon an optimal statistical combination of available atmospheric observations and an estimate of atmospheric state provided by a previous model forecasts (also known as background). With the rapid increase in the number and types of atmospheric observations (e.g. remote sensors, fixed and mobile radars, surface in-situ instruments, satellite observations, rawinsondes, and aircraft observations), computationally it is very challenging to assimilate these observations into the model. Numerical methods that are commonly used for this purpose are the Optimal Interpolation or OI method, variational methods in three

and four dimensions (3D-VAR and 4D-VAR), Kalman filtering and the Ensemble Kalman filtering (EnKF) methods.

One of the research goal at the National Severe Storms Laboratory (NSSL) located in Norman, Oklahoma is to improve the accuracy of severe weather forecasts (e.g. hail, tornado, thunderstorms etc.) and to increase the warning lead time of severe weather events. Longer warning lead times are expected to help save lives, reduce damages and injuries, provide improved local flood warnings and positively impact air traffic and surface transportation routing. To increase the warning lead time, it is essential that the model be initialized with a very accurate representation of ongoing convective weather. The only key instrument that observes the 3-D volumetric scans of severe weather events every ~5 minutes at high spatial resolution are the Weather Surveillance Radar – 1988 Doppler (WSR-88D). In recent years, researchers found that assimilating the WSR-88D radar observations using ensemble square-root Kalman (EnSRF), a variant of EnKF data assimilation techniques shows promise in initializing storm-scale[1] NWP models (Snyder and Zhang 2003; Zhang et al. 2004; Dowell et al. 2004a, b; Tong and Xue 2005; Xue et al. 2006; Dowell and Wicker 2009; Aksoy et al. 2009). These studies assimilate either synthetic or real WSR-88D Doppler radar reflectivity or radial velocity observations of thunderstorms. Results indicate that by assimilating radar reflectivity and radial velocity observations, the filter is able to retrieve the unobserved variables, such as temperature and the full three-dimensional wind field, successfully. In contrast, model simulations without radar data assimilation create the thunderstorm in an ad hoc manner using a warm bubble, but are unable to generate the quickly developed observed storm

[1] Storm-scale is a scale of sizes of individual thunderstorms.

characteristics, and often diverge from observations. Indeed, data assimilation is a key element in producing reasonable predictions of observed thunderstorms.

While the EnSRF technique shows promise in storm-scale assimilation, one limitation of EnSRF is that it assimilates observations serially, making it computationally very expensive when the number of observations is very large. Therefore to limit the number of observations, radar observations are either objectively analyzed to coarser resolution (Dowell et al. 2004a, b; Dowell and Wicker 2009, Aksoy et al. 2009) or synthetic radar observations are generated at coarser resolution (Snyder and Zhang 2003; Zhang et al. 2004; Tong and Xue 2005; Xue et al. 2006; Caya et al. 2005). Based on recent studies (Snyder and Zhang 2003; Zhang et al. 2004; Xue et al. 2006; Caya et al. 2005) it appears reasonable to expect that at least 10 radar scans are needed to produce reasonable analyses of storms. However, part of the challenge in using ~5-min radar observations to initialize thunderstorms in numerical models is that a number of storm features evolve on a timescale of minutes and are poorly sampled by ~5-min data. Xue et al. (2006) and Lei et al. (2007) show that the assimilation of synthetic 1-min radar data leads to analyses that more closely approach the truth solution than the analyses created using synthetic 5-min radar data. Moreover, with the advent of the emerging Phased Array Radar (PAR; Forsyth and coauthors 2004;Weber et al. 2007, Yu et al. 2007, Zrnić et al. 2007) technology as a potential replacement candidate of the aging WSR-88D in the next 10-15 years, it is possible to scan a thunderstorm phenomena in less than a minute (Heinselman et al. 2008). Since accurate analyses require approximately 10 radar scans, the amount of time needed to obtain these scans from the WSR-88D is at least 45 min. However, the PAR can produce 10 radar scans in less than 10 min. Thus, it is reasonable

to expect that PAR observations can generate accurate storm analyses very quickly using a shorter assimilation period. A shorter assimilation window also is highly desirable in an operational environment if these analyses are to be used to increase warning lead times for severe weather warnings.

One major sources of error in storm-scale data assimilation and forecasts is the microphysical parameterization scheme used to represent the microphysical characteristics of the storms in the NWP model (Dowell et al. 2004a, b; Gilmore et al. 2004; van den Heever and Cotton 2004; Snook and Xue 2008; Tong and Xue 2008a). Microphysics refers to the model emulation of cloud and precipitation processes that remove excess atmospheric moisture directly resulting from the dynamically driven forecast wind, temperature, and moisture fields. The most commonly used type of microphysical scheme in storm-scale modeling is a single-moment bulk microphysics scheme (Lin et al. 1983; Tao and Simpson 1993; Schultz 1995; Straka and Mansell 2005; Hong and Lim 2006) that uses predefined precipitation particle densities and the intercept parameters (microphysical parameters) and predicts only the particle mixing ratios. The determination of suitable values for the microphysical parameters in storm scale data assimilation is very difficult due to the unavailability of in situ microphysics observations. Several observational studies indicate that the particle densities and the intercept parameters can vary widely among storms and even within a single storm (Gunn and Marshall 1958; Houze et al. 1979, 1980; Mitchell 1988; Pruppacher and Klett 2000; Cifelli et al. 2000; Brandes et al. 2007). Several experimental studies also show that the selection of microphysical parameters in storm-scale modeling has profound impact on the analyses and forecasts of severe weather events, and an arbitrary selection of those

parameters may lead to significant error (Gilmore et al. 2004; van den Heever and Cotton 2004; Snook and Xue 2008). One approach to account for the uncertainty in a storm-scale EnSRF data assimilation system is to vary the microphysical parameters within the same microphysics scheme among the ensemble members. The hope is that by using a variety of realistic precipitation particle parameters, an ensemble is more likely to span the truth.

As mentioned earlier, while the EnSRF data assimilation technique shows promise for radar observation assimilation, numerous challenges exist. From the computational point of view, data assimilation using EnSRF method is efficient when the number of observations is smaller. When the number of observations exceeds the number of model states, the EnSRF method becomes computationally inefficient. Therefore, one major challenges of radar data assimilation is the heavy computational demands of assimilating radar observations in true radar resolution. Moreover, observations of the same storm are available from more than one radar. Xue et al. (2006) examined the impact of assimilating radars observations of the same storm from multiple radars and conclude that it is generally true that the larger the number of observations the better the analyses. Therefore the implementations of EnSRF become exceedingly time consuming as the dimension of the number of the observations increases. The question is how to handle this huge amount of observations efficiently in data assimilation?

A survey of literatures on data assimilation suggests that if the number of observation is very large in dimension compared to the model state, the information form of the filter (Maybeck 1979; Mutambara 1998; Lewis et al. 2006; Simon 2006; Kaminski et al. 1971; Dyer and McReynolds 1969; Bierman 1977) may be computationally more efficient than the traditional Kalman filter. While the traditional Kalman filter calls for

inverting the matrix in observation space, the information filter calls for the inversions of the model space. Moreover, the information filter is algebraically equivalent to Kalman filter. The information filter has been around for years, but to our knowledge the information filter data assimilation method is not yet examined in atmospheric data assimilation. Therefore, another focus of this dissertation is to evaluate the applicability of Information filter as an atmospheric data assimilation technique.

1.2 Outline

Mathematical formulations of the data assimilation techniques implemented in this research are presented in Chapter 2. The standard formulation of the Kalman filter (KF), extended Kalman filter (EKF), information filter (IF), extended information filter (EIF) and the extended square root information filter are given. The framework for the ensemble Kalman filter (EnKF), a suboptimal solution to reduce the huge computational cost for large dimensional problems are discussed and the formulation of the Ensemble Square Root Filter (EnSRF) is presented.

In Chapter 3, the EnSRF data assimilation technique is applied to examine the impact of high temporal frequency observation assimilation over a shorter assimilation period. The synthetic radar observations are generated at a coarser spatial resolution to reduce the computation cost of ingesting the data into the model using EnSRF. A description of the model used for the study is discussed followed by the algorithm developed to create synthetic radar observations. Two sets of experiment are conducted with identical settings based on the assumption of a perfect model in which both the truth simulation and the ensemble data assimilation system use the same microphysics scheme and constant microphysics parameter. One experiment assimilates synthetic WSR-88D

observations, in which a volume scan is conducted every 5 min, while the other experiment assimilates synthetic PAR observations, in which a volume scan is conducted every 1 min. The results obtained from the EnSRF analyses and forecasts are then compared and discussed. This chapter is the basis for the paper Yussouf and Stensrud (2010a).

The experiments conducted in Chapter 3 are based on perfect model framework using constant intercept and density parameters of precipitation particle categories. However, model error is a critical factor and needs to be incorporated into the data assimilation system. Therefore, to examine the potential value of assimilating radar observations using a range of intercept and density parameters across the ensemble members within the same microphysics scheme, two sets of radar observations assimilation experiments are conducted using both perfect and imperfect model framework and are presented in Chapter 4. The WSR-88D radar observations are created at a coarser resolution as in Chapter 3 and assimilated using the EnSRF technique. The results are compared to quantify the value of using different microphysical parameters within the same microphysics scheme. A manuscript, Yussouf and Stensrud, 2010b based on this chapter is currently under review.

Given the high spatial resolution of the radar observations and the advent of new radar and other remote sensing technology, it is highly likely that the observation dimensionality exceeds the dimensionality of the model state vector, indicating that EnSRF data assimilation method may be computationally very expensive; suggesting efficient filter designs need to be tested. In Chapter 5, the feasibility of information filter as a method for high density observations assimilation is examined. Both extended

information filter (EIF) and extended Kalman filter (EKF) are implemented using the 40 dimensional Lorenz 96 nonlinear models and the performance of EIF in assimilating low and high density observations are compared with the benchmark EKF assimilation technique.

A summary of the dissertation and suggestions for future work are contained in Chapter 6.

Chapter 2

Methods of Data Assimilation Based on Kalman Filter

2.1 Introduction

Even though the Kalman filter is developed in early sixties (Kalman 1960; Kalman and Bucy 1961), the meteorological research community started using this method for data assimilation in the 80's. The mathematical equations presented in this chapter are the building blocks of the data assimilation techniques used in this study and mainly follows Lewis et al. (2006); Lakshmivarahan and Stensrud (2009); Mutambara (1998); Simon (2006); Dyer and McReynolds (1969); Kaminski et al. (1971); Maybeck (1979) and Whitaker and Hamill (2002).

Let $x_k \in R^n$ denotes the true state of a dynamical system, e.g., the atmospheric system, at time *k*. It is assumed that x_k is not directly observable but values of z_k (a known function of x_k) called observation is available for some subset of values of *k*. Let

$$F_k = \{z_i \mid 1 \le i \le k\} \tag{2.1}$$

denote the set of all observations during the interval [1, k]. Let $\hat{x}_k$ denote the estimate of x_k at time *k*. The problem of computing (*a*) $\hat{x}_k$ given F_k is called the filtering problem, (*b*) $\hat{x}_k$ given F_N for some $k < N$ is called the smoothing problem and (*c*) $\hat{x}_{k+s}$ given F_k for some s≥1 is called the prediction problem. While the filtering and prediction problem use only the past and present information, smoothing uses all the past, present and the future information.

In this research study, we are interested in the filtering problem. The Kalman filter and the extended Kalman filter formulation is given in Section 2.2 followed by the formulation of information filter, the extended information filter and the extended square root information filter in Section 2.3. The framework for the ensemble Kalman filter and the formulation of the ensemble square-root filter is given in Section 2.4 followed by the summary in Section 2.5.

2.2 The Filtering Problem

There are at least two distinct ways to formulate the filtering problem: first as a linear, unbiased and minimum variance formulation that gives raise to the so called covariance form of the Kalman filters. Second, is using the classical Bayesian formulation with the least square cost function leading to the so called information filters (Lewis et al. 2006, Chapter 17). The numerical accuracy of the covariance form can be increased by using the square-root of the covariance. Moreover, to reduce the computation burden of propagating the covariance forward in time, reduced rank approximation of the filter can be used. The derivation of the filter equations consist of two main steps: the forecast step using the model and the data assimilation step using the model forecast and observations. Here we consider the discrete time, continuous space filtering and prediction problems.

2.2.1 The Kalman Filter

The Kalman filter is a recursive linear minimum variance estimator when the model is linear and the observations are linear functions of the state. Let $M \in R^{nxn}$ denote the state transition matrix and the linear dynamics of evolution of the states is given by

$$x_{k+1} = M_k x_k + w_{k+1}, \tag{2.2}$$

where w_k is the sequence of Gaussian white noise representing model error where $w_k \sim N(0, Q_k)$ and $Q_k \in R^{nxn}$ is a symmetric and positive definite matrix. It is assumed that the initial state is $x_0 \in N(\hat{x}_0, P_0)$ and satisfies the following conditions:

a. x_0 is with known mean vector $E(x_0) = \hat{x}_0 = m_0$ and known covariance matrix $Cov(x_0) = E[(x_0 - m_0)(x_0 - m_0)^T] = P_0$.

b. The model error is unbiased, that is $E(w_k) = 0$ for all *k* and is temporally uncorrelated, that is

$$E[w_k w_j^T] = Q_k \quad \text{if } j = k$$
$$= 0 \quad \text{otherwise.}$$

c. The model error w_k and the initial state x_0 are uncorrelated

$$E[w_k x_0^T] = 0 \text{ for all } k.$$

Let $z_k \in R^m$ denote the observation of the system (2.2) and the observations are linear function of the state:

$$z_k = H_k x_k + v_k, \tag{2.3}$$

where $H \in R^{mxn}$ is a matrix known as the observation operator (or the *H* operator) that maps the model state variables onto the observations, v_k is the sequence of the Gaussian

white noise representing measurement noise $v_k \sim N(0, R_k)$, and $R_k \in R^{mxm}$ is a symmetric and positive definite matrix with the following properties:

a. v_k has mean zero: $E(v_k) = 0$.

b. v_k is temporally uncorrelated:

$$E[v_k v_j^T] = R_k \quad \text{if } j = k.$$

$$= 0 \quad \text{otherwise.}$$

where $R_k \in R^{mxm}$ is a symmetric and positive definite matrix.

c. v_k is uncorrelated with the initial state x_0 and the model error w_k, that is

$$E[x_0 v_k^T] = 0 \text{ for all } k>0.$$

$$E[v_k x_j^T] = 0 \text{ for all } k \text{ and } j.$$

It is assumed that the model noise w_k, the observation noise v_k and the initial condition x_0 are mutually uncorrelated. In the forecast step, starting from an optimal estimate $\hat{x}_{k-1}$ at time *k-1*, the model (2.2) is used to produce a forecast x_k^f at time *k*. In the data assimilation step, this forecast x_k^f is linearly combined with the observation z_k to produce the optimal estimate $\hat{x}_k$.

Given a linear dynamical model, the equations for the Kalman filter are as follows (Chapter 27, Lewis et. al 2006):

Model Forecast Step:

$$x_{k+1}^f = M_k \hat{x}_k \tag{2.4}$$

$$P_{k+1}^f = M_k \hat{P}_k M_k^T + Q_{k+1}. \tag{2.5}$$

Data Assimilation Step:

$$\hat{x}_{k+1} = x_{k+1}^f + K_{k+1}[z_{k+1} - H_{k+1} x_{k+1}^f] \tag{2.6}$$

$$K_{k+1} = P_{k+1}^{f} H_{k+1}^{T} [H_{k+1} P_{k+1}^{f} H_{k+1}^{T} + R_{k+1}]^{-1} \tag{2.7}$$

$$\hat{P}_{k+1} = [I - K_{k+1} H_{k+1}] P_{k+1}^{f} . \tag{2.8}$$

However it is found that round-off errors in the calculation of the covariance matrices P_{k+1}^{f} and $\hat{P}_{k+1}$ from (2.5) and (2.8) of the Kalman filter resulting from large condition number can cause loss of symmetry and/or positive definiteness. This in turn can lead to numerical inaccuracy, filter divergence and instability. Further investigations shows that the effect of round-off errors can be mitigated by performing the filter computations using the square root version of the covariance matrix. The square root filtering increases the numerical precision of Kalman filtering by reducing the condition number of the matrices involved in the computation (Lewis et al. 2006). This in turn can help prevent filter divergence and instability. However, this improved performance from the square root filter is obtained at the cost of greater computational cost.

2.2.2 The Extended Kalman Filter: Ist and 2nd Order

If the dynamic model is nonlinear and the observations are non-linear function of the state, the evolution of the states is given by

$$x_{k+1} = M(x_k) + w_{k+1} , \tag{2.9}$$

and the observation is given by

$$z_k = h(x_k) + v_k . \tag{2.10}$$

The major impediments in formulating the equations for the nonlinear problem are the difficulty of computing $M(\hat{x}_k)$ in the forecast step and $h(\hat{x}_k)$ in the data assimilation step. The only solution is to formulate an approximation to the moment

dynamics. Thus the extended Kalman filter (EKF) is a linear estimate of the nonlinear system obtained from the linearization of the nonlinear state and observations equations. Using the Taylor series expansion, and discarding the second and higher order moments, the first order EKF is formulated as follows (Chapter 29, Lewis et. al 2006):

Model Forecast Step:

$$x_{k+1}^{f} = M(\hat{x}_k) \tag{2.11}$$

$$P_{k+1}^{f} = D_M(\hat{x}_k)\hat{P}_k D_M^T(\hat{x}_k) + Q_{k+1}. \tag{2.12}$$

Data Assimilation Step:

$$\hat{x}_{k+1} = x_{k+1}^{f} + K_{k+1}[z_{k+1} - h(x_{k+1}^{f})] \tag{2.13}$$

$$K_{k+1} = P_{k+1}^{f} D_h^T(x_{k+1}^{f})[D_h(x_{k+1}^{f}) P_{k+1}^{f} D_h^T(x_{k+1}^{f}) + R_{k+1}]^{-1} \tag{2.14}$$

$$\hat{P}_{k+1} = [I - K_{k+1} D_h(x_{k+1}^{f})] P_{k+1}^{f}, \tag{2.15}$$

where $D_M(\hat{x}_k)$ is the Jacobian of M(.) evaluated at $\hat{x}_k$ and $D_h(x_{k+1}^{f})$ is the Jacobian of *h(x)* evaluated at x_{k+1}^{f}.

The second order EKF equations can be obtained by discarding the third and higher order moments (Chapter 29, Lewis et. al 2006):

Model Forecast Step:

$$x_{k+1}^{f} = M(\hat{x}_k) + \frac{1}{2}\partial^2(M, \hat{P}_k) \tag{2.16}$$

$$P_{k+1}^{f} = D_M(\hat{x}_k)\hat{P}_k D_M^T(\hat{x}_k) + Q_{k+1}. \tag{2.17}$$

Data Assimilation Step:

$$\hat{x}_{k+1} = x^f_{k+1} + K_{k+1}[z_{k+1} - h(x^f_{k+1}) - \frac{1}{2}\partial^2(h, P^f_{k+1})] \tag{2.18}$$

$$K_{k+1} = P^f_{k+1} D^T_h(x^f_{k+1})[D_h(x^f_{k+1}) P^f_{k+1} D^T_{h(x^f_{k+1})} + R_{k+1}]^{-1} \tag{2.19}$$

$$\hat{P}_{k+1} = [I - K_{k+1} D_h(x^f_{k+1})] P^f_{k+1}. \tag{2.20}$$

However, due to neglecting the third and higher order statistical moments in the covariance evolution, the EKF data assimilation often encounters unbounded error growth.

2.3 Information Filter: A Dual Formulation of Kalman Filter

The information filter is essentially a Kalman filter that is expressed in terms of measures of information about the states rather than direct state estimates and its covariance. The information filter is constructed from the information[1] space and is algebraically equivalent to the Kalman filter. The two key variables in the information filter are the information matrix and the transformed state vector. The information matrix is defined as the inverse of the covariance matrix, $Y = P^{-1}$ and the transformed state vector is a product of the information matrix and the model state, $y = P^{-1}x = Yx$. Thus while the covariance matrix represents the uncertainty in the state estimate, the information matrix represents the certainty in the information estimate (Mutambara 1998; Simon 2006). However, there are two ways of formulating the information filter using the information matrix: one using the model state space (Simon 2006) and the other one using the transformed state vector (Mutambara 1998). Both formulations are given below.

[1] The term information is employed in the *Fisher* sense, that is, a measure of the amount of information about the random state *x* present in the set of observations *z*, up to time *k*.

2.3.1 The Information Filter

A. State Space Formulation:

The linear Kalman filter can be written in terms of the model state vector and information matrix (Simon 2006) as follows:

Model Forecast Step

$$x_{k+1}^{f} = M_k \hat{x}_k$$

$$P_{k+1}^{f} = M_k \hat{P}_k M_k^T + Q_{k+1}.$$

Applying the matrix inversion lemma to the above equation

$$(A+C\ B\ D^T)^{-1} = A^{-1} - A^{-1}C\left[B^{-1} + D^T A^{-1} C\right]^{-1} D^T A^{-1}$$

and substituting $A = Q_{k+1}$, $B = P_k$ and $C = D = M_k$, the predicted information matrix Y_{k+1}^{f} is:

$$Y_{k+1}^{f} = (P_{k+1}^{f})^{-1} = (M_k \hat{P}_k M_k^T + Q_{k+1})^{-1}$$

$$\Rightarrow Y_{k+1}^{f} = Q_{k+1}^{-1} - Q_{k+1}^{-1} M_k [\hat{Y}_k + M_k^T Q_{k+1}^{-1} M_k]^{-1} M_k^T Q_{k+1}^{-1}. \quad (2.21)$$

Data Assimilation Step

$$\hat{x}_{k+1} = x_{k+1}^{f} + K_{k+1}[z_{k+1} - H_{k+1} x_{k+1}^{f}]$$

$$(\hat{P}_{k+1})^{-1} = (P_{k+1}^{f})^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}.$$

The updated information matrix $\hat{Y}_{k+1}$ is:

$$\Rightarrow \hat{Y}_{k+1} = (\hat{P}_{k+1})^{-1} = Y_{k+1}^{f} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}. \quad (2.22)$$

Again, from the matrix inversion lemma,

$$[A^T B^{-1} A + D^{-1}]^{-1} A^T B^{-1} = DA^T [ADA^T + B]^{-1}.$$

Assuming $D = P_{k+1}^{f\ -1}, A = H_{k+1}, B = R_{k+1}$ we get

$$K_{k+1} = [(P_{k+1}^f)^{-1} + H_{k+1}^T R_{k+1}^{-1} H_{k+1}]^{-1} H_{k+1}^T R_{k+1}^{-1}$$

$$\Rightarrow K_{k+1} = [Y_{k+1}^f + H_{k+1}^T R_{k+1}^{-1} H_{k+1}]^{-1} H_{k+1}^T R_{k+1}^{-1}.$$

Therefore,

$$\Rightarrow K_{k+1} = (\hat{Y}_{k+1})^{-1} H_{k+1}^T R_{k+1}^{-1}. \tag{2.23}$$

While the standard Kalman filter equations require the inversion of an *m x m* matrix, where *m* is the number of measurements, the state space formulation of the information filter equations require at least a couple of *n* x *n* matrix inversions, where *n* is the number of states. Therefore, if *m* >> *n* (i.e., there are significantly more measurements than states variables) it may be computationally more efficient to use the information filter rather than the Kalman filter. However, the *m* x *m* matrix inversion on R_k is common to the standard Kalman filter or the information filter Kalman gain equation. But if R_k is constant, then the inversion can be a part of the initialization process, so the Kalman gain equation may not require this *m* x *m* matrix inversion after all. The same thinking also applies to the inversion of Q_k.

If the initial uncertainty is infinite, the information filter is more mathematically precise since $Y_0 = 0$ can be set for the zero initial certainty case but $P_0 = \infty$ cannot be set. However, if the initial uncertainty is zero, $P_0 = 0$ can be set but $Y_0 = \infty$ cannot be set. This makes the standard Kalman filter more mathematically precise for the zero initial uncertainty case.

B. Information Space Formulation:

The linear Kalman filter can be written in terms of the transformed state vector and the information matrix (Mutambara 1998) as follows:

Model Forecast Step:

$$y_{k+1}^{f} = Y_{k}^{f} M_{k} (\hat{Y}_{k})^{-1} \hat{y}_{k} \tag{2.24}$$

$$Y_{k+1}^{f} = [M_{k} (\hat{Y}_{k})^{-1} M_{k}^{T} + Q_{k+1}]^{-1}. \tag{2.25}$$

Data Assimilation Step:

$$\hat{y}_{k+1} = y_{k+1}^{f} + H_{k+1}^{T} R_{k+1}^{-1} z_{k+1} \tag{2.26}$$

$$\hat{Y}_{k+1} = Y_{k+1}^{f} + H_{k+1}^{T} R_{k+1}^{-1} H_{k+1}, \tag{2.27}$$

where the predicted (updated) transformed state vector is y_{k+1}^{f} ($\hat{y}_{k+1}$) and the predicted (updated) information matrix is Y_{k+1}^{f} ($\hat{Y}_{k+1}$)

2.3.2 The Extended Information Filter

The extended information filter (EIF) is the extension of the linearized estimation algorithm for the nonlinear systems similar to EKF. The EIF predicts and estimates the information state and its associated information matrix for the nonlinear dynamic model and observations.

A. State Space Formulation:

The EIF can be written in terms of the model state vector and information matrix (Simon 2006) as follows:

Model Forecast Step

$$x_{k+1}^{f} = M(\hat{x}_k)$$

$$P_{k+1}^{f} = D_M(\hat{x}_k)\hat{P}_k D_M^T(\hat{x}_k) + Q_{k+1}$$

$$Y_{k+1}^{f} = (P_{k+1}^{f})^{-1} = [D_M(\hat{x}_k)\hat{P}_k D_M^T(\hat{x}_k) + Q_{k+1}]^{-1}.$$

$$\Rightarrow Y_{k+1}^{f} = Q_{k+1}^{-1} - Q_{k+1}^{-1} D_M(\hat{x}_k)[\hat{Y}_k + D_M^T(\hat{x}_k) Q_{k+1}^{-1} D_M(\hat{x}_k)_k]^{-1} D_M^T(\hat{x}_k) Q_{k+1}^{-1} \qquad (2.28)$$

Data Assimilation Step

$$\hat{x}_{k+1} = x_{k+1}^{f} + K_{k+1}[z_{k+1} - h(x_{k+1}^{f})]$$

$$(\hat{P}_{k+1})^{-1} = (P_{k+1}^{f})^{-1} + D_h^T(x_{k+1}^{f}) R_{k+1}^{-1} D_h(x_{k+1}^{f}).$$

The updated information matrix $\hat{Y}_{k+1}$ is:

$$\Rightarrow \hat{Y}_{k+1} = (\hat{P}_{k+1})^{-1} = Y_{k+1}^{f} + D_h^T(x_{k+1}^{f}) R_{k+1}^{-1} D_h(x_{k+1}^{f}) \qquad (2.29)$$

$$K_{k+1} = [(P_{k+1}^{f})^{-1} + D_h^T(x_{k+1}^{f}) R_{k+1}^{-1} D_h(x_{k+1}^{f})]^{-1} D_h(x_{k+1}^{f}) R_{k+1}^{-1}.$$

$$\Rightarrow K_{k+1} = [Y_{k+1}^{f} + D_h^T(x_{k+1}^{f}) R_{k+1}^{-1} D_h(x_{k+1}^{f})]^{-1} D_h^T(x_{k+1}^{f}) R_{k+1}^{-1}.$$

Therefore,

$$\Rightarrow K_{k+1} = (\hat{Y}_{k+1})^{-1} D_h(x_{k+1}^{f}) R_{k+1}^{-1}. \qquad (2.30)$$

B. Information Space Formulation:

The extended Information filter can be written in terms of the transformed state vector and the information matrix (Mutambara 1998) as follows:

Model Forecast Step:

$$y_{k+1}^f = Y_k^f M(\hat{x}_k) \tag{2.31}$$

$$Y_{k+1}^f = [D_M(\hat{x}_k)\hat{Y}_k^{-1} D_M^T(\hat{x}_k) + Q_{k+1}]^{-1}. \tag{2.32}$$

Data Assimilation Step:

$$\hat{y}_{k+1} = y_{k+1}^f + D_h^T(x_{k+1}^f)R_{k+1}^{-1}[z_k - h(x_{k+1}^f) + D_h(x_{k+1}^f)x_{k+1}^f] \tag{2.33}$$

$$\hat{Y}_{k+1} = Y_{k+1}^f + D_h^T(x_{k+1}^f)R_{k+1}^{-1}D_h(x_{k+1}^f). \tag{2.34}$$

Remarks: A closer look into the above equations reveals that to implement the EIF in transformed space, we need to compute the Jacobians $D_M(\hat{x}_k)$ and $D_h(x_{k+1}^f)$. Therefore we also need the knowledge of $\hat{x}_k$ and x_{k+1}^f to implement the Jacobians in model space indicating that we must implement EKF in parallel to compute Y_{k+1}^f, $\hat{Y}_{k+1}$, $\hat{y}_{k+1}$ and y_{k+1}^f. This is a serious limitation of implementing the EIF in transformed space.

2.3.3 The Square-Root Information Filter

Let (*x*, *P*) be given where $x \in R^n$ is a random vector and $P \in R^{nxn}$ is its covariance. It is assumed *P* is always symmetric positive definite (SPD). Let $P = SS^T$ be the cholesky factorization of *P*, where $S \in R^{nxn}$ is a lower triangular matrix called the square root of *P*. We know that the matrix $Y = P^{-1}$ is called the information matrix. Clearly *Y* is SPD since *P* is. Let $P = \Lambda\Lambda^T$ be the cholesky square root factorization of *Y*. It can be verified that $(S^T)^{-1} = (S^{-1})^T = S^{-T}$ and the inverse of a lower triangular matrix is lower triangular. Hence $\Lambda = S^{-1}$. We also know that $y = P^{-1}x = Yx$ is the transformed state vector. It is

well known that there are several equivalent ways to express the classical linear Kalman filter equations:

(1) Covariance form using (x, P): Kalman (1960), Jazwinski (1970), Sorenson (1976), Lewis et al. (2006), Maybeck (1979), Simon (2006), and Mutambara (1998).

(2) Square-root covariance form using (x, S): Maybeck (1979), Lewis et al. (2006), Bierman (1977).

(3) Information form using (x, Y): Maybeck (1979), Simon (2006).

(4) Square-root information form using (x, Λ): Dyer and McReynolds (1969), Bierman (1977).

(5) Transformed state information form using (y, Y): Kaminski et al. (1971), Mutambara (1998).

In this section we concentrate on the square root version of the information filters using (x, Λ). Golub (1965) was the first to use the notion of square root information and orthogonal transformation to solve the linear least squares problem.

Golub's method: Let $z \in R^m$ be the observations. Let

$$z = Hx + v, \tag{2.35}$$

where $H \in R^{mxn}$ and $v \sim N(0, R)$. Then the least squares estimate is obtained by minimizing

$$f(x) = \|z - Hx\|^2_{R^{-1}}. \tag{2.36}$$

Let $\Gamma \in R^{mxm}$ be an orthogonal transformation and let $R^{-1} = \Lambda^T(z)\Lambda(z)$ be the square factorization. Thus, it can be verified that

$$\begin{aligned} f(x) &= \|\Lambda(z)(z - Hx)\|^2 \\ &= \|b - \Lambda(z)Hx\|^2 \quad \text{where } b = \Lambda(z)z \end{aligned}$$

$$= \left\| \Gamma(b - \Lambda(z)Hx) \right\|^2 . \tag{2.37}$$

Our goal is to choose Γ such that

$$\Gamma\Lambda(z)H = \begin{bmatrix} \overbrace{\tilde{\Lambda}}^{n} \\ \cdots \\ 0 \end{bmatrix} \begin{matrix} n \\ \\ m-n \end{matrix} \quad \text{and} \quad \Gamma b = \begin{bmatrix} \hat{b} \\ \cdots \\ b_1 \end{bmatrix} \tag{2.38}$$

where $\tilde{\Lambda} \in R^{nxn}$ is an upper triangular matrix. Thus

$$f(x) = \left\| \begin{bmatrix} \hat{b} \\ \cdots \\ b_1 \end{bmatrix} - \begin{bmatrix} \tilde{\Lambda} \\ \cdots \\ 0 \end{bmatrix} x \right\|^2$$

$$= \left\| \hat{b} - \tilde{\Lambda} x \right\|^2 + \left\| b_1 \right\|^2 . \tag{2.39}$$

Hence the least square estimate is given by the solution of the upper triangular system

$$\hat{\Lambda} x = \hat{b} \text{ or } \hat{x} = (\hat{\Lambda})^{-1} \hat{b} . \tag{2.40}$$

Solution of (2.40) is obtained by back substitution which takes $O(n^2)$ steps. Now consider

$$\begin{bmatrix} \tilde{\Lambda}^T & \vdots & 0 \end{bmatrix} \begin{bmatrix} \tilde{\Lambda} \\ \cdots \\ 0 \end{bmatrix} = \left[\Gamma\Lambda(z)H \right]^T \left[\Gamma\Lambda(z)H \right]$$

$$= H^T \Lambda^T(z) \Gamma^T \Gamma \Lambda(z) H$$

$$= H^T \Lambda^T(z) \Lambda(z) H$$

$$= H^T R^{-1} H . \tag{2.41}$$

That is,

$$\tilde{\Lambda}^T \tilde{\Lambda} = H^T R^{-1} H \,. \tag{2.42}$$

Hence the covariance of $\hat{x}$ is given by

$$(\tilde{\Lambda}^T \tilde{\Lambda})^{-1} = (H^T R^{-1} H)^{-1} \tag{2.43}$$

which matches the standard result (Lewis et al. (2006), Chapter 14).

Thus the above method obtains $(\hat{x}, \hat{\Lambda})$, the estimate and the square root of its information matrix from the input $(z, H, \Lambda(z))$.

Remark: The orthogonal transformation Γ can be realized in one of two ways: using Gramm-Schmidt orthogonalization or Householder's transformation. In our computations, we use the Householder's transformation (Lakshmivarahan and Dhall 1990).

Dyer and McReynolds (1969) extended Golub's idea to include the model and derived the square root information version of the Kalman filter. In the following, we extend Dyer and McReynolds algorithm to derive the extended square root information version of the filter using (x, Λ) that is suitable for nonlinear systems.

Remark: Extended information from using (*x, Y*) is given in Simon (2006) and extended transformed state information filter using (*y, Y*) is given in Mutambara (1998).

2.3.3.1 The Extended Square-Root Information Filter:

Let

$$x_{k+1} = M(x_k) + \Lambda_{k+1}^{-1}(Q) w_{k+1} \tag{2.44}$$

be the nonlinear dynamic model where $w_k \sim N(0, I_n)$, $Q_k^{-1} = \Lambda_k^T(Q)\Lambda_k(Q)$ is the square factorization of Q_k^{-1} and Q_k is the covariance of the model noise at time *k*.

Let

$$z_k = H_k x_k + \Lambda_k^{-1}(z) v_k, \tag{2.45}$$

where $v_k \sim N(0, I_m)$ and $R_k^{-1} = \Lambda_k^T(z)\Lambda_k(z)$ be the square root factorization of R_k^{-1} where R_k is the covariance of the observation noise at time *k*.

Clearly,

$$\begin{aligned} \operatorname{cov}[\Lambda_k^{-1}(Q) w_k] &= E[(\Lambda_k^{-1}(Q) w_k)(\Lambda_k^{-1}(Q) w_k)^T] \\ &= \Lambda_k^{-1}(Q) E(w_k w_k^T) \Lambda_k^{-T}(Q) \\ &= [\Lambda_k^T(Q)\Lambda_k(Q)]^{-1} = Q_k \end{aligned} \tag{2.46}$$

and

$$\begin{aligned} \operatorname{cov}[\Lambda_k^{-1}(z) v_k] &= E[(\Lambda_k^{-1}(z) v_k)(\Lambda_k^{-1}(z) v_k)^T] \\ &= \Lambda_k^{-1}(z) E(v_k v_k^T) \Lambda_k^{-T}(z) \\ &= [\Lambda_k^T(z)\Lambda_k(z)]^{-1} = R_k . \end{aligned} \tag{2.47}$$

Cox (1964) proved that the estimation of x_k given $\{z_1, z_2, ...z_k\}$ is equivalent to minimizing

$$\hat{J}_k = J_k^f + \|v_k\|^2 \tag{2.48}$$

where

$$J_k^f = \sum_{i=1}^{k-1} \|v_i\|^2 + \sum_{i=1}^{k} \|w_i\|^2 + \left\| \hat{\Lambda}_1 [x_1 - \hat{x}_1] \right\|^2 \tag{2.49}$$

where $(\hat{x}_1, \hat{\Lambda}_1)$ is the prior information on x_1, the initial state.

Assume recursively that J_k^f is expressed as

$$J_k^f = \left\| \Lambda_k^f x_k - d_k^f \right\| + \left\| e_k^f \right\|^2 \tag{2.50}$$

much like as in Golub's algorithm where $x_k^f = (\Lambda_k^f)^{-1} d_k^f$ is the forecast and Λ_k^f is the square root of the information matrix of x_k^f. That is, (x_k^f, Λ_k^f) is given.

Analyis phase: Given (x_k^f, Λ_k^f) and a new observation $(z_k, H_k, \Lambda_k^{-1}(z))$, from (2.48) we now obtain

$$\hat{J}_k = \left\| \Lambda_k^f x_k - d_k^f \right\| + \left\| \Lambda_k^{-1}(z)(z_k - H_k x_k) \right\| + \left\| e_k^f \right\|^2 \tag{2.51}$$

with

$$v_k = \Lambda_k^{-1}(z)[z_k - H_k x_k]. \tag{2.52}$$

obtained from (2.45). Setting $\Lambda_k^{-1}(z) z_k = b_k^f$,

$$\hat{J}_k = \left\| \begin{pmatrix} \Lambda_k^f \\ -\Lambda_k^{-1}(z) H_k \end{pmatrix} x_k - \begin{pmatrix} d_k^f \\ b_k^f \end{pmatrix} \right\|^2 + \left\| e_k^f \right\|^2. \tag{2.53}$$

Let $\hat{\Gamma}_k$ be the orthogonal transformation such that

$$\hat{\Gamma}_k \begin{bmatrix} \Lambda_k^f \\ \cdots \\ -\Lambda_k^{-1}(z) H_k \end{bmatrix} \begin{matrix} n \\ \\ m \end{matrix} = \begin{bmatrix} \hat{\Lambda}_k \\ \cdots \\ 0 \end{bmatrix} \begin{matrix} n \\ \\ m \end{matrix} \quad \text{where } \hat{\Lambda}_k \in R^{nxn} \text{ is a upper triangular} \tag{2.54}$$

and

$$\hat{\Gamma}_k \begin{bmatrix} d_k^f \\ \cdots \\ b_k^f \end{bmatrix} = \begin{bmatrix} \hat{d}_k \\ \cdots \\ b_k \end{bmatrix} \tag{2.55}$$

In view of (2.54) and(2.55), rewrite (2.53) as

$$\hat{J}_k = \left\| \begin{pmatrix} \hat{\Lambda}_k \\ 0 \end{pmatrix} x_k - \begin{pmatrix} \hat{d}_k \\ b_k \end{pmatrix} \right\|^2 + \left\| e_k^f \right\|^2$$

$$= \left\| \hat{\Lambda}_k x_k - \hat{d}_k \right\| + \left\| \hat{e}_k \right\|^2 \tag{2.56}$$

where $\left\| \hat{e}_k \right\|^2 = \left\| b_k \right\|^2 + \left\| e_k^f \right\|^2$.

Clearly $(\hat{x}_k, \hat{\Lambda}_k)$ is obtained from (2.56) by solving an upper triangular system

$$\hat{\Lambda}_k x_k = \hat{d}_k \tag{2.57}$$

from

$$\begin{aligned}
\left(\hat{P}_k \right)^{-1} &= (\hat{\Lambda}_k)^T \hat{\Lambda}_k \\
&= \begin{bmatrix} (\hat{\Lambda}_k)^T & \vdots & 0 \end{bmatrix} \begin{bmatrix} \hat{\Lambda}_k \\ \cdots \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} (\hat{\Lambda}_k)^T & \vdots & (\Lambda_k(z) H_k)^T \end{bmatrix} \hat{\Gamma}_k \hat{\Gamma}_k \begin{bmatrix} \Lambda_k^f \\ \cdots \\ -\Lambda_k(z) H_k \end{bmatrix} \\
&= (\Lambda_k^f)^T \Lambda_k^f + (\Lambda_k(z) H_k)^T \Lambda_k(z) H_k \\
&= (\Lambda_k^f)^T \Lambda_k^f + H_k^T \Lambda_k^T(z) \Lambda_k(z) H_k \\
&= \left(P_k^f \right)^{-1} + H_k^T R_k^{-1} H_k
\end{aligned}$$

$$\hat{P}_k = \left[\left(P_k^f \right)^{-1} + H_k^T R_k^{-1} H_k \right]^{-1} \tag{2.58}$$

which proves that $(\hat{x}_k, \hat{\Lambda}_k)$ is the correct analysis and its square root information matrix (Lewis et al. 2006: Chapter 17) is the Bayesian formulation.

Forecast phase: Given $(\hat{x}_k, \hat{\Lambda}_k)$ our goal is to obtain $(x_{k+1}^f, \Lambda_{k+1}^f)$. Consider

$$J_{k+1}^f = \hat{J}_k + \|w_{k+1}\|^2$$

$$= \left\| \hat{\Lambda}_k x_k - \hat{d}_k \right\|^2 + \|w_{k+1}\|^2 + \left\| \hat{e}_k \right\|^2 . \tag{2.59}$$

But from the model equations (2.44) we get

$$x_k = M^{-1}(x_{k+1} - \Lambda_{k+1}^{-1}(Q)w_{k+1}) . \tag{2.60}$$

Adding and subtracting $\hat{x}_k$, we get

$$x_k = M^{-1}(x_{k+1} - \Lambda_{k+1}^{-1}(Q)w_{k+1} + \hat{x}_k - \hat{x}_k)$$

$$= M^{-1}(\hat{x}_k + (x_{k+1} - \hat{x}_k) - \Lambda_{k+1}^{-1}(Q)w_{k+1}) . \tag{2.61}$$

Using the first-order Taylor expansion around $\hat{x}_k$, we can approximate x_k in (2.60) as

$$x_k \approx M^{-1}(\hat{x}_k) + D_{M^{-1}}(\hat{x}_k)(x_{k+1} - \hat{x}_k) - D_{M^{-1}}(\hat{x}_k)(\Lambda_{k+1}^{-1}(Q)w_{k+1}) \tag{2.62}$$

Assuming that the model *M* is smooth, it can be verified that

$$D_{M^{-1}}(x) = D_M^{-1}(x) \tag{2.63}$$

that is, the Jacobian of the inverse of *M* is the inverse of the Jacobian of *M*, provided that the latter is non-singular. Hence, using (2.63) in(2.62), we get

$$x_k \approx M^{-1}(\hat{x}_k) + D_M^{-1}(\hat{x}_k)(x_{k+1} - \hat{x}_k) - D_M^{-1}(\hat{x}_k)(\Lambda_{k+1}^{-1}(Q)w_{k+1}) . \tag{2.64}$$

Now substituting (2.64) in (2.59) and simplifying, we get

$$J_{k+1}^f = \left\| \hat{\Lambda}_k D_M^{-1} \left[x_{k+1} - \Lambda_{k+1}^{-1}(Q)w_{k+1} \right] - \hat{d}_k \right\|^2 + \|w_{k+1}\|^2 + \left\| \hat{e}_{k+1} \right\|^2 \tag{2.65}$$

where $D_M^{-1} = D_M^{-1}(\hat{x}_k)$ for simplicity in notation, and

$$\left\| \hat{e}_{k+1} \right\|^2 = \left\| \hat{e}_k \right\|^2 + \left\| \hat{\Lambda}_k \left[M^{-1}(\hat{x}_k) - D_M^{-1} \hat{x}_k \right] \right\|^2, \tag{2.66}$$

which is independent of x_{k+1} and w_{k+1}. We can rewrite (2.65) as

$$J_{k+1}^f = \left\| \begin{bmatrix} I_n & \vdots & 0 \\ \cdots & \cdots & \cdots \\ -\hat{\Lambda}_k D_M^{-1} \Lambda_{k+1}^{-1}(Q) & \vdots & \hat{\Lambda}_k D_M^{-1} \end{bmatrix} \begin{bmatrix} w_{k+1} \\ x_{k+1} \end{bmatrix} - \begin{bmatrix} 0 \\ \cdots \\ \hat{d}_k \end{bmatrix} \right\|^2 + \left\| \hat{e}_{k+1} \right\|^2 \tag{2.67}$$

Now, let Γ^f be such that

$$\Gamma^f \begin{bmatrix} I_n & \vdots & 0 \\ \cdots & \cdots & \cdots \\ -\hat{\Lambda}_k D_M^{-1} \Lambda_{k+1}^{-1}(Q) & \vdots & \hat{\Lambda}_k D_M^{-1} \end{bmatrix} = \begin{bmatrix} A & \vdots & B \\ \cdots & \cdots & \cdots \\ 0 & \vdots & \Lambda_{k+1} \end{bmatrix} \tag{2.68}$$

where $A \in R^{nxn}$ is non-singular and

$$\Gamma^f \begin{bmatrix} 0 \\ \cdots \\ \hat{d}_k \end{bmatrix} = \begin{bmatrix} b_{k+1} \\ \cdots \\ d_{k+1} \end{bmatrix} \tag{2.69}$$

Combining (2.68) and (2.69) we readily see that

$$J_{k+1}^f = \left\| A w_{k+1} + B x_{k+1} - b_{k+1} \right\|^2 + \left\| \Lambda_{k+1} x_{k+1} - d_{k+1} \right\|^2 + \left\| e_{k+1} \right\|^2 \tag{2.70}$$

Since *A* is non-singular, we can set for any x_{k+1}

$$w_{k+1} = A^{-1} \left[B x_{k+1} + b_{k+1} \right] \tag{2.71}$$

there by annihilating the first term in (2.70). The optimal forecast is then obtained by solving

$$\Lambda_{k+1} x_{k+1} = d_{k+1} \tag{2.72}$$

A little reflection reveals that Λ_{k+1} in (2.72) is not in general an upper triangular matrix. This can be easily obtained by multiplying both sides of (2.72) by an orthogonal matrix Γ_{k+1} such that $\Gamma_{k+1}\Lambda_{k+1} = \Lambda^f_{k+1}$, upper triangular matrix and $\Gamma_{k+1}d_{k+1} = d^f_{k+1}$.

Hence (2.72) becomes

$$\Lambda^f_{k+1}x^f_{k+1} = d^f_{k+1} \tag{2.73}$$

Hence $x^f_{k+1} = \left(\Lambda^f_{k+1}\right)^{-1} d^f_{k+1}$ can be easily obtained by back substitution. This gives us the pair (x^f_{k+1}, d^f_{k+1}) and the cycle continues.

2.4 The Ensemble Kalman Filter

The implementation of the Kalman filter or the EKF in large dimensional meteorological problems is not feasible due to the huge computational cost and also due to the unbounded error growth caused by the closure problem (Evensen 1992). The alternative is to implement reduced rank approximation of the full-rank covariance matrix. There are two types of reduced-rank approximations. The first is the explicit reduced-order filter (Lewis et al. 2006; Evensen 2007; Lakshmivarahan and Stensrud 2009). If at time $k = 0$, $\hat{x}_0$ is the initial estimate of the unknown atmospheric state and $\hat{P}_0$ is the covariance of the estimate $\hat{x}_0$. Then

$$\begin{aligned}\hat{P}_0 &= [V(1:r)\Lambda^{1/2}(1:r)][V(1:r)\Lambda^{1/2}(1:r)]^T \\ &= \hat{S}_0(1:r)\,\hat{S}_0^{\,T}(1:r),\end{aligned}$$

where $\hat{S}_0(1:r) = \sum_{i=1}^{r} \lambda_i^{1/2} v_i$ Then $\hat{S}_0 = \hat{S}_0(1:r) \in R^{nxr}$ is the rank r square root of $\hat{P}_0$. Thus $\hat{S}_0$ is called a full rank if $r = n$ or a reduced rank square root if $r < n$. Rank r square root of a matrix contains r largest eigenvalues and vectors of $\hat{S}_0$ to get the maximum spread of the ensemble. The second class of implicit reduced-order filters for nonlinear problems where in the forecast x_{k+1}^f, the estimate $\hat{x}_{k+1}$ and their covariances P_{k+1}^f and $\hat{P}_{k+1}$ respectively are computed using the standard Monte Carlo framework as the sample moments of an ensemble of size N much smaller compared to n, the dimension of the state space of the model. The following section describes the creation of the ensemble using the Monte Carlo framework.

(a) Creation of Initial Ensemble

To create an initial ensemble given ($\hat{x}_0$, $\hat{S}_0$), we compute the initial (one-sided) ensemble of size N where $1 < N < n$. The i^{th} member of the initial ensemble (Lewis et al. 2006) is given by

$$\hat{\xi}_0(i) = \hat{x}_0 + \hat{S}_0 \, y_0(i), \tag{2.74}$$

where $y_0(i) \in R^r, y_0(i) \sim N(0, I)$ for $1 \le i \le N$, and $r < N$ Clearly, $\hat{x}_0$ is the mean of this initial ensemble $\{\hat{\xi}_0(i) : 1 \le i \le N\}$. Let

$$\hat{a}_0(i) = \hat{\xi}_0(i) - \hat{x}_0 = \hat{S}_0 \, y_0(i), \tag{2.75}$$

for $1 \le i \le N$ denote the anomaly associated with the i^{th} ensemble member $\hat{\xi}_0(i)$ and let

$$\hat{A}_0 = [\hat{a}_0(1), \hat{a}_0(2), ... \hat{a}_0(N)] \in R^{nxN} \tag{2.76}$$

denote the $n \times N$ matrix of anomaly vectors. In ensemble filtering, the information about the state and its covariance are extracted from the pair $(\hat{x}_0, \hat{A}_0)$. Let $1 = (1,1.....1)^T \in R^N$ denote a column vector all of whose elements are 1s. Since $\hat{x}_0$ is the ensemble mean, it can be verified that $\hat{A}_0$ **1** = *0* and

$$\frac{\hat{A}_0(\hat{A}_0)^T}{N-1} = \hat{S}_0(\hat{S}_0)^T . \tag{2.77}$$

That is, $\hat{S}_0 = (N-1)^{-1/2} \hat{A}_0$ this is the scaled anomaly matrix. This relation shows that there is a natural relation between ensemble methods and reduced rank filtering.

(b) Creation of Forecast Ensemble

To create a forecast ensemble at time k, we assume inductively that the pair $(\hat{x}_k, \hat{A}_k)$ is given, where $\hat{A}_k$ **1** = *0*. For $1 \le i \le N$, compute a deterministic forecast ensemble

$$\xi^f_{k+1}(i) = M(\hat{\xi}_k(i) = M(\hat{x}_k + \hat{a}_k(i)), \tag{2.78}$$

using the nonlinear model $M(\cdot)$ where $\hat{a}_k(i)$ is the i^{th} column of the anomaly matrix $\hat{A}_k$. Expanding (2.36) in a first-order Taylor series yields

$$\xi^f_{k+1}(i) \approx M(\hat{x}_k) + D_M(\hat{x}_k)\hat{a}_k(i), \tag{2.79}$$

where $D_M(x)$ is the Jacobian of $M(x)$ with respect to x. Combining this expression with the relation $\hat{A}_0 \mathbf{1} = 0$ indicates that the sample average of this forecast ensemble is given by

$$x_{k+1}^f = M(\hat{x}_k). \tag{2.80}$$

Define A_{k+1}^f as the forecast anomaly matrix whose i^{th} column $a_{k+1}^f(i)$ is given by

$$a_{k+1}^f(i) = \xi_{k+1}^f(i) - x_{k+1}^f(i), \quad \mathbf{1} \le \boldsymbol{i} \le \boldsymbol{N}. \tag{2.81}$$

Then, the pair (x_{k+1}^f, A_{k+1}^f) constitutes the forecast ensemble at time (k+1). It can be verified that $A_{k+1}^f \mathbf{1} = 0$ and

$$(N-1)^{-1/2} A_{k+1}^f = S_{k+1}^f,$$

which is the rank q square root of P_{k+1}^f

(c) Creation of Analysis Ensemble

Given an ensemble forecast (x_{k+1}^f, A_{k+1}^f) and a new observation z_{k+1}, the data assimilation step computes the new analysis ensemble $(\hat{x}_{k+1}, \hat{A}_{k+1})$. All the known algorithms for ensemble filtering essentially differ in the details of this data assimilation step and can be classified into two groups – stochastic and deterministic methods.

2.4.1 Stochastic Method

Earlier studies of the application of ensemble filtering in geophysical problems (Evensen 1994) indicates that if the same observation z_{k+1} are assimilated in each of the forecast ensemble members, the resulting covariance and the spread of the ensemble is

less than the theoretical values dictated by the Kalman filter. As the filter evolves in time, this reduction in covariance leads to the collapse of the ensemble and the spread shrinks rapidly. However, assimilating perturbed observations can compensate this deficiency. This scheme is first implemented by Houtemaker and Mitchell (1998) and later clarified by Burgers et al. (1998). In this type of method, the i^{th} member of the analysis ensemble at time (k+1) is computed as

$$\hat{\xi}_{k+1}(i) = \xi^{f}_{k+1}(i) + K_{k+1}[z_{k+1}(i) - H_{k+1}\xi^{f}_{k+1}(i)], \tag{2.82}$$

where K is the Kalman gain in the square root form and $z_{k+1}(i)$ is the i^{th} perturbed observation given by

$$z_{k+1}(i) = z_{k+1} + v_{k+1}(i). \tag{2.83}$$

The covariance of the analysis ensemble generated using (2.41) matches the theoretical value given by the Kalman filter as $N \to \infty$ (Houtemaker and Mitchell 1998; Burgers et al. 1998; Lewis et al. 2006). While the use of perturbed observations improves the performance of the ensemble Kalman filter, side effects can occur due to sampling errors, especially when N is small. Since ensemble sizes used in meteorological applications typically are small, other strategies to compensate for the underestimation of the analysis covariance are needed.

2.4.2 Deterministic Method and its Variants

Ensemble filtering approaches designed without perturbed observations belongs to the deterministic method. The ensemble square root filter (EnSRF) (Whitaker and Hamill 2002), ensemble transform Kalman filter (ETKF) (Bishop et al. 2001) and the

ensemble adjusted Kalman filter (EAKF) (Anderson 2001) are examples of deterministic method that do not require the observations to be perturbed. All these method have one unifying theme that they all exploit combinations of ideas from square root of covariance matrices along with the reduced rank approximations resulting from the small ensemble size. The general idea behind this type of method is that given $A_{k+1}^f \in R^{nxN}$, find a transformation T such that

$$\hat{A}_{k+1} = T(A_{k+1}^f), \tag{2.84}$$

where $\hat{A}_{k+1}$ satisfies $\hat{A}_{k+1} 1 = 0$ and

$$\frac{\hat{A}_{k+1}(\hat{A}_{k+1})^T}{N-1} \to P_{k+1}^f,$$

as the actual value as N increases. All the known algorithms of this type realize $T(\cdot)$ as a linear transformation. Accordingly, $\hat{A}_{k+1}$ is obtained from A_{k+1}^f either by a left multiplication by a matrix $T_L \in R^{nxn}$ or by a right multiplication by $T_R \in R^{NxN}$. The ETKF computes

$$\hat{A}_{k+1} = A_{k+1}^f T_R, \tag{2.85}$$

whereas EAKF computes

$$\hat{A}_{k+1} = T_L A_{k+1}^f. \tag{2.86}$$

Since $T_R \in R^{NxN}$ and $T_L \in R^{nxn}$, ETKF requires less computations compared to EAKF unless the EAKF is implemented in the sequential least squares framework in which case the computational requirements are the same. However, $\hat{A}_{k+1}$ generated by EAKF

automatically satisfies $\hat{A}_{k+1}1 = 0$. Clearly, one needs to impose additional conditions on T_L to ensure that $\hat{A}_{k+1}1 = 0$.

2.4.2.1 The Ensemble Kalman Square-Root Filter

The EnSRF proposed by Whitaker and Hamill (2002) is used extensively in storm-scale radar data assimilation studies (Snyder and Zhang 2003; Zhang et al. 2004; Dowell et al. 2004; Tong and Xue 2005; Xue et al. 2006; Aksoy et al. 2009). This method also is used in this study to assimilate radar observations (see Chapters 3 and 4).

Following Whitaker and Hamill (2002), the EnSRF algorithm uses nonlinear forecast model and the observations also are a nonlinear function of the state while the assimilation of observations is a linear. The observations are assimilated serially, one observations after another, which is an approximation based on the assumption that observation errors are uncorrelated in space and time. Therefore, the observation error covariance matrix R_{k+1} reduce to scalar (variance) each time an observation $z_{j,k+1} \in z_{k+1}$ is assimilated and so does the matrix $H_{k+1}P_k^f H_{k+1}^T$. We know, the Kalman gain equation (2.7) is

$$K_{k+1} = P_{k+1}^f H_{k+1}^T \left[H_{k+1} P_{k+1}^f H_{k+1}^T + R_{k+1} \right]^{-1}.$$

Now

$$P_{k+1}^f H_{k+1}^T \cong Cov(x_{k+1}^f, x_{k+1}^f) H_{k+1}^T = Cov[x_{k+1}^f, H_{k+1}(x_{k+1}^f)],$$

and for one observation assimilation, (2.7) reduces to

$$K = Cov\left[x_{k+1}^f, H_{k+1}(x_{k+1}^f) \right] \left[Var[H_{k+1}(x_{k+1}^f)] + Var(z_{j,k+1}) \right]^{-1}. \qquad (2.87)$$

Here $H_{k+1}(x_{k+1}^f)$ is the conversion of the model variables to the observation type. Now the numerator (or the background error covariances) of the Kalman gain *K* is estimated from the forecast ensemble as follows:

$$P_{k+1}^f H_{k+1}^T = Cov[x_{k+1}^f, H_{k+1}(x_{k+1}^f)]$$

$$= \frac{1}{N-1}\sum_{i=1}^{N}[x_{k+1}^f(i) - x_{k+1}^f(N)][H_{k+1}(x_{k+1}^f(i)) - \overline{H_{k+1}(x_{k+1}^f)}]. \quad (2.88)$$

Here $\overline{H_{k+1}(x_{k+1}^f)}$ is the ensemble mean of the model variables converted to the observation type i.e. $\overline{H_{k+1}(x_{k+1}^f)} = \frac{1}{N}\sum_{i=1}^{N} H_{k+1}x_{k+1}^f(i)$. The denominator is estimated as:

$$H_{k+1}P_{k+1}^f H_{k+1}^T = Var[H_{k+1}(x_{k+1}^f)]$$

$$= \frac{1}{N-1}\sum_{i=1}^{N}[H_{k+1}(x_{k+1}^f(i)) - \overline{H_{k+1}(x_{k+1}^f)}]^2, \quad (2.89)$$

where

$$x_{k+1}^f(N) = \frac{1}{N}\sum_{i=1}^{N} x_{k+1}^f(i). \quad (2.90)$$

The covariance calculation in the numerator of *K* tends to be small at large distances from the scalar observation and likely contains considerable sampling error due to relatively small ensemble sizes (Houtekamer and Mitchell 2001). To overcome this problem, an observation is allowed to update only state variables at nearby grid points by multiplying *K* by a weight *W* that is a function of distance from the observation (following Gaspari and Cohn 1999). This covariance localization function that defines *W* decreases smoothly from 1 at the observation location to 0 at the edge of an elliptical influence region of a particular radius.

To account for the unperturbed observations, a α factor also is included in the equation (Whitaker and Hamill 2002):

$$\alpha = \left[1 + \sqrt{R_{k+1}(H_{k+1}P_{k+1}^{f}H_{k+1}^{T} + R_{k+1})^{-1}} \right]^{-1}. \tag{2.91}$$

The ensemble mean and the members are updated according to the following equations:

$$\hat{x}_{k+1}(N) = x_{k+1}^{f}(N) + WK\left[z_{j,k+1} - \overline{H_{k+1}(x_{k+1}^{f})} \right] \tag{2.92}$$

$$\hat{x}_{k+1}(i) = \hat{x}_{k+1}(N) + (x_{k+1}^{f}(i) - x_{k+1}^{f}(N)) + WK\alpha\left[\overline{H_{k+1}(x_{k+1}^{f})} - H_{k+1}(x_{k+1}^{f}(i)) \right], \tag{2.93}$$

where over bar indicates the ensemble mean and *i* is an index to identify a particular ensemble member. Again, H_{k+1} is the observation operator that maps model state to the observation type and locations. The ensemble $\hat{x}_{k+1}(i)$ calculated from (2.93) then becomes the prior ensemble for the assimilation of the next observation and the algorithm continues until all observations are processed at time $k+1$.

To summarize, the data assimilation procedure for the EnSRF is as follows:

1) First create an initial ensemble of model states at time $k = 0$.
2) Advance the ensemble to a make forecasts to the first observations time $k = 1$.
3) Assimilate observations serially using (2.92) and (2.93) until all the observations valid at time $k = 1$ are assimilated.
4) Advance the ensemble to make a forecast to the next observations time $k = 2$.
5) Repeat step *3* and *4* until all available observations are assimilated.

2.5 Summary

This chapter presents the mathematical formulation of the data assimilation techniques that are implemented in Chapters 3, 4 and 5 of this research endeavor. The chapter begins with the mathematical formulations of the traditional Kalman filter and information filter data assimilation technique followed by the derivations of the variants: EKF and EIF. While the traditional Kalman filter operates by updating the mean and its covariance, the ensemble Kalman filter approach approximates a finite number of members and the filtering algorithm is applied to every ensemble members from which the required mean and the variance are computed as the standard sample moments. One variant of the ensemble Kalman filter, which is widely used for large scale data assimilation, is the EnSRF data assimilation technique and the formulation of the EnSRF is presented at the end of this chapter. The EnSRF is implemented in Chapter 3 and Chapter 4 of this dissertation to assimilate radar observations in NWP model. However, the technique assimilates observations serially as shown earlier. Therefore, while the EnSRF data assimilation technique shows promise for radar observation assimilation, the technique is computationally very expensive when the number of observations to assimilate increases. To answer this question, the EIF is implemented in Chapter 5 of this dissertation using the simple Lorenz model and compared against the benchmark EKF data assimilation technique.

Chapter 3

Data Assimilation using Ensemble Square-Root Filter: Impact of High Temporal Frequency Observations

3.1 Introduction

In this chapter, the EnSRF data assimilation technique described in Chapter 2 is used to assimilate high temporal frequency synthetic radar observations. The EnSRF data assimilation technique has shown great promise in assimilating radar observations into NWP model and is widely used by the storm-scale data assimilation research community (Snyder and Zhang 2003; Dowell et al. 2004a, b; Tong and Xue 2005; Jung et al. 2008a and b). However, the EnSRF technique assimilates observations serially, therefore the technique is computationally feasible when the size of observation vector m is less than the size of model state n ($m < n$). Past literature (Snyder and Zhang 2003; Zhang et al. 2004; Dowell et al. 2004; Xue et al. 2006) also suggests that reasonable analyses of an ongoing severe weather event can be incorporated into the NWP model from assimilating approximately 10 volume scans of WSR-88D radar observations. However, part of the challenge in using ~5-min WSR-88D radar observations to initialize thunderstorms in numerical models is that a number of storm features evolve on a timescale of minutes and are poorly sampled by ~5-min data. Since accurate analyses require approximately 10 radar scans, the amount of time needed to obtain these scans from the WSR-88D is at least 45 min. However, the PAR can produce 10 radar scans in less than 10 min. Thus, it is reasonable to expect that PAR observations can generate accurate storm analyses very quickly using a shorter assimilation period.

Therefore, in an attempt to evaluate the value of PAR observation assimilation for a shorter period of time, a set of observing system simulation experiments (OSSEs; Lord et al. 1997) within a perfect model framework are conducted using the EnSRF data assimilation technique. However, to reduce the computational expense of assimilating huge number of observations using EnSRF, synthetic radar observations are generated at a coarser 1 km range resolution instead of the 0.25 km interval available from the operational radars. One experiment assimilates 3 volumes of WSR-88D radar observations and another experiment assimilates 15 volumes of PAR observations during the short 15-min period. The analyses and the forecasts from WSR-88D and PAR observation assimilations are then compared to determine the accuracy of the storm represented in the analyses and forecast. This chapter is the basis for the paper Yussouf and Stensrud (2010a).

Description of the storm-scale model used in this study is given in Section 3.2 for the generation of the observations in Section 3.3. The experimental design is described in Section 3.4. Section 3.5 presents the results obtained from the EnSRF analyses and forecasts, followed by a summary in Section 3.6.

3.2 Description of the COMMAS Model

The NWP model used for this study is the Collaborative Model for Multiscale Atmospheric Simulation (COMMAS; Wicker and Wilhelmson 1995) model. The COMMAS is a three-dimensional model which was developed in the early 1990s to study the dynamics of supercells and tornados. Over the years the model has changed considerably in terms of equation set and numerical algorithms. The prognostic variables

for this model include the three velocity components (*u, v*, and *w)*, pressure in the form of the perturbation Exner function (π), potential temperature[1] (θ), mixing coefficient (k_m), water-vapor mixing ratio (q_v), cloud-water mixing ratio (q_c), and hydrometeor mixing ratio ($q_{1.....r}$), where *r*, the number of hydrometeor categories depends on which of several options in COMMAS are chosen for the precipitation-microphysics scheme. For this study, the Gilmore et al. (2004) version of the Lin et al. (1983) precipitation-microphysics scheme is used. This scheme includes one rain category and three ice classes: thus $q_{1.....4} = q_r$ (rain), q_i (cloud ice crystals), q_s (snow), and q_h (hail graupel). The moist processes represented in the model are cloud condensation, cloud and rain evaporation, autoconversion of cloud to rain, ice-crystal initiation, vapor deposition and sublimation for ice species, freezing, melting, accretion, aggregation, rain shedding by wet hail/graupel and precipitation fallout (Gilmore et al. 2004).

This model is a three-dimensional grid with n_x, n_y and n_z points in the *x, y,* and *z* directions giving rise to $n_g = n_x n_y n_z$ grid points. At each grid points, all these prognostic variables are represented as dependent variables of *x, y* and *z*. The integration process can be denoted by $x_{k+1} = M(x_k)$ where $x_k \in R^n$ denotes the *n* real vector called the model state at time $k = 0, 1, 2, 3....$ and the *n*-dimensional Euclidean space R^n is called the model space. The mapping $M : R^n \to R^n$ denoted by $M(x_k) = (M_1(x_k), M_2(x_k), ... M_n(x_k))^T$ is a vector valued nonlinear function of the vector x_k defining the state transition rule of the model. When the PDEs are approximated using the finite differencing schemes, errors are introduced resulting from the finite grid

[1] The temperature a parcel of dry air would have if brought adiabatically (i.e., without transfer of heat or mass) to a standard pressure level of 1000 mb.

length, truncation in the spectral expansion and other approximations and simplifications. These errors introduce an additional term into the equations $x_{k+1} = M(x_k) + w_{k+1}$. It is generally assumed that $w_k \in R^n$ has mean $E(w_k) = 0$ and covariance $\text{cov}(w_k) = Q_k$. It is a sequence of Gaussian white noise representing model error where $w_k \sim N(0, Q_k)$ and $Q_k \in R^{nxn}$.

The PDEs in this model include three momentum equations, the pressure and thermodynamic equations, six moisture and water equations, the turbulent kinetic energy (TKE) as well as the Smagorinsky mixing scheme equations (Wicker and Scamarock 2002; Coniglio et al. 2006).

The momentum equations are

$$\frac{du_i}{dt} = -C_p \bar{\theta} \frac{\partial \pi}{\partial x_i} - \in_{ijk} f_j (u_k - \bar{u_k}) + \delta_{i3} B + D_i \tag{3.1}$$

$$B = g \left[\frac{\theta - \bar{\theta}}{\bar{\theta}} + 0.61 \left(q_v - \bar{q_v} \right) + \sum_l q_l + \sum_i q_i \right] \tag{3.2}$$

$$D_i = \frac{\partial}{\partial x_i} \left[K_m \left(\frac{\partial u_i}{\partial x_i} + \frac{\partial u_j}{\partial x_i} \right) + \frac{2}{3} \delta_{ij} E \right]. \tag{3.3}$$

The u_i *(i* = 1, 2, 3*)* are the velocities *u, v* and *w* respectively, θ is the potential temperature, π is the perturbation Exner function used for the pressure (which is the deviation of pressure from the initial unperturbed state $\bar{\pi}$), q_l, q_i and q_v are the mixing ratios of liquid, ice and vapor hydrometeors, C_p is the specific heat at constant pressure.

Bars over the individual variables refer to the initial undisturbed state which is function of z only. These equations include the Coriolis force, with *f* being the Coriolis parameter. The operator *d/dt* denotes the substantial derivative given by

$$\frac{d}{dt} \equiv \frac{\partial}{\partial t} + u_i \frac{\partial}{\partial x_i}.$$

Buoyancy effects (*B*) from hydrometeor loading are accounted for in the summations over the liquid and ice hydrometeor mixing ratios, respectively. The terms denoted by D_i represent the subgrid turbulent mixing, K_m is the momentum eddy mixing coefficient and *E* is the subgrid-scale kinetic energy.

The equation for thermodynamic θ and hydrometeor equations are given by

$$\frac{d\theta}{dt} = D_\theta + M_\theta, \tag{3.4}$$

where M_θ refers to microphysical terms and D_θ to turbulence terms.

$$D_\theta = \frac{\partial}{\partial x_i}\left(K_m \frac{\partial \theta}{\partial x_i} \right)$$

$$\frac{dq_i}{dt} = -\frac{1}{\bar{\rho}} \frac{\partial(\bar{\rho} V_{T_i} q_i)}{\partial z} + D_{qi} + M_{qi}, \tag{3.5}$$

where $q_i = q_v, q_c, q_r, q_i, q_s, q_h$ and $D_{qi} = \frac{\partial}{\partial x_i}\left(K_m \frac{\partial q_i}{\partial x_i} \right)$.

The first term on the right hand side of (3.4) represents the hydrometeor fallout and the M_{qi} represent the microphysical terms. The term V_T represents terminal velocity. The model includes multiple options for precipitation microphysical schemes. The LFO scheme used in this study has a terminal velocity for all liquid and ice hydrometeors, except cloud water.

The pressure equation is

$$\frac{\partial \pi}{\partial t} + \frac{C_s^2}{C_p \bar{\rho} \bar{\theta}_v} \frac{\partial (\bar{\rho} u_i)}{\partial x_i} = F_\pi . \tag{3.6}$$

As in most cloud-scale models the TF_π term is set to zero as it primarily changes the mean pressure within the domain that impacts the dynamical solution minimally.

The parameterization of the turbulent mixing coefficient is represented using a prognostic turbulent kinetic energy (TKE) equation to represent the energy associated with the subgrid scale eddies. The TKE equation represents the effects of buoyancy, shear, diffusion and dissipation and is expressed as:

$$\frac{dE^{1/2}}{dt} = \frac{C_m l}{2}[shear] - \frac{P_r C_m l}{2}[buoy] + \frac{\partial}{\partial x_i}\left[2K_m \frac{dE^{1/2}}{\partial x_i}\right] - \frac{C_e E}{2l} \tag{3.7}$$

$$K_m = C_m E^{1/2} l .$$

Here K_m is the momentum eddy mixing coefficient, E is the subgrid-scale kinetic energy and $l = (\Delta x \Delta y \Delta z)^{1/3}$. The length scale is computed as the cube root of the computational grid volume or as a function of distance from the lower boundary.

The numerical integration scheme closely follows that of Wicker and Skamarock (2002). The 3rd order Runge-Kutta time-split (RK3) scheme is chosen (Wicker and Skamarock, 2002) for time integration while 5th and 3rd order finite difference approximations are used for the spatial derivatives in horizontal and vertical directions, respectively. If the model equation is $x_{k+1} = M(x_k)$, the RK3 integration takes the form of 3 steps to advance a solution x_k to $x_{k+\Delta k}$:

$$x_k^* = x_k + \frac{\Delta k}{3} M(x_k)$$

$$x_k^{**} = x_k + \frac{\Delta k}{2} M(x_k^*)$$

$$x_{k+\Delta k} = x_k + \Delta k M(x_k^{**}),$$

where Δk is the model timestep. The model incorporates a vertically stretched grid and supports open and periodic lateral boundary conditions.

The COMMAS modeling system software is written in Fortran-77, Fortran-90 and Python scripts. The model outputs are written in netCDF format. The model supports VIS5D, NCAR graphics and NCL for graphics.

3.3 Observations

3.3.1 The Truth Simulation

The model domain for the truth simulation is 100 km long in the horizontal (x and y) and 18 km tall in the vertical (z) direction. The resolution in the horizontal direction is $\Delta x = \Delta y = 1$ km and the domain is vertically stretched with Δz =100 m vertical spacing at the bottom to Δz =700 m vertical spacing at the domain top. Thus $n_x = n_y = 100$ and n_z = 45 grid points. Hence $n_g = n_x\, n_y\, n_z = 4.5 \times 10^5$ and the number of variables L = 10. The origin of the Cartesian coordinates (x, y, z) is at the lower left corner (southwest) corner of the domain. The model is initialized with the classic Weisman-Klemp analytic sounding (Weisman and Klemp, 1982) in a horizontally homogeneous environment. This initial sounding (vertical u, v , θ, and q_v profiles) with a high shear produce split storms which are the model equivalent of the observed supercells. An ellipsoidal thermal bubble (temperature perturbations) with 10 km radius in the horizontal direction and 1.4 km radius in the vertical direction is placed at the center of the domain to initiate a supercell thunderstorm at t = 0 min. A temperature excess of 2.5 K is specified at the center of the bubble and decreases gradually to 0 K at the edge. The model time step (Δk) for the simulation is 6 sec. The simulation is allowed to evolve through 2 hours of model time. The ellipsoidal thermal bubble develops into a convective cell within the first 30 minutes of the simulation and the first echo is seen by the radar emulator at around k = 25 min. Over the next 30 min, the convective cell splits into two cells, one moving right towards the southeast and the other moving towards the northwest. During the second hour of the simulation, the right-moving cell tends to dominate the system with a few short lived smaller cells developing in between the two main cells. Since the storms naturally move

out of the 100 x 100 km domain in this time period, the domain grid is translated at u = 17 and v = 7 ms^{-1} to keep the main storm near the center of the model domain.

The simulated truth is run on a SGI machine (Altix 3700BX2 system) with 64 processors, 64 GB of RAM and operates on Linux environment (SUSE Linux Enterprise Server 10). The time required to run the truth is 1 hour.

3.3.2 Radar Emulator Design and Observations Generation

The most significant difference between the PAR and the WSR-88D is the antenna. While the phased array antenna forms a beam electronically by controlling the phase of 4,352 transmit/receive elements, the WSR-88D's parabolic antenna forms a beam from a feedhorn. Thus the steering of the beam in PAR is done electronically from a stationary pane while it is accomplished mechanically, by rotating and elevating the antenna for WSR-88D. The electronic steering of the beam from PAR provided higher temporal frequency observations than the WSR-88D. Thus while PAR takes less than a minute to scan a complete volume of the severe weather events, the conventional WSR-88D takes about 5 minutes to scan the same event. Therefore while 88D takes 5 minutes to get a complete picture of the atmosphere, PAR captures 5 snapshots of the developing weather events during the same 5-min period The PAR and WSR-88D antennas share three similarities: wavelength (S-band: 9.4-cm *vs.* 10-cm, respectively), range resolution (both 250 m), and the PAR can mimic WSR-88D VCP scan.

A radar emulator is created in this study using Fortran-90 programming language to generate artificial WSR-88D and PAR observations from the truth run. Simulation of radar observations also are done in several studies (Xue et al., 2006; Tong and Xue,

2008a and b; Jung et al. 2008a and b; Lei et al. 2008). The input to the radar emulator is the 3-D gridded model variables from the simulated supercell storm which is stored in netCDF format. The model output of the truth is in Cartesian coordinates and the emulator scans the data in a conical surface in spherical coordinates. To allow for PAR and WSR-88D radar antenna, the behavior of the radar emulator is controlled by specifying radar parameters (eg. beamwidth, range and azimuthal intervals, etc) and scanning strategy (Table 3.1). While in reality the radar reflectivity and radial velocity observations are generated from averaging radar pulses, the radar emulator in this study constructs the reflectivity and radial velocity values by averaging reflectivity and wind components from the three-dimensional, gridded model data within the beamwidth area using a simplified version of a volume averaging technique (Wood et al. 2009). The radar observations are created on a spherical coordinate system centered on the radar. The code for the radar emulator is given in the Appendix A.

Table 3.1. Radar Emulator Control Parameters

Control Paramters
Radar Location
Radar Type (PAR or WSR-88D)
Volume Coverage Pattern (VCP) Modes
Beamwidth
Effective Beamwidth
Sampling Interval (Range and Azimuthal)

The observation operator H in (2.3) for reflectivity follows the relationships of Smith et al. (1975) and is as follows:

$$Z = Z_{er} + Z_{eh} + Z_{es}, \tag{3.8}$$

where Z_{er}, Z_{es}, and Z_{eh} are the equivalent reflectivity factors for rain, snow, and hail, respectively (in $mm^6\ m^{-3}$) and Z is the mean reflectivity factor. The conversion of the model variables into these reflectivity components goes as follows:

The rain component is calculated from

$$Z_{er} = c(\pi p_w)^{-1.75}(N_{0r})^{-0.75}(\rho_a q_r)^{1.75}, \tag{3.9}$$

where $c = 7.2 \times 10^{20}$, ρw is the density of water in kg m^{-3}, N_{0r} is the intercept parameter in mm^{-4} in the assumed inverse exponential drop-size distribution, ρ_a is the air density in kg m^{-3}, and q_r is the rainwater mixing ratio in g kg^{-1}.

The snow component of reflectivity (Z_{es}) is defined to be

$$Z_{es} = c(k_{iw})(\rho_s^2 / \rho_r^2)(\pi\rho_s)^{-1.75}(N_{os})^{-0.75}(\rho_a q_s)^{1.75} \tag{3.10}$$

for temperatures below freezing (dry snow) and

$$Z_{es} = c(\pi p_s)^{-1.75}(N_{0s})^{-0.75}(\rho_a q_s)^{1.75} \tag{3.11}$$

for temperatures above freezing (wet snow), where $K_{iw} = (0.21/0.93)$ is the ratio of the dielectric constants for ice and water, ρ_s is the density of snow in kg m^{-3}, N_{0s} is the intercept parameter for the distribution of snow in mm^{-4}, qB_s is the snow mixing ratio in kg kg^{-1} and the other quantities are the same as for Z_{er}.

The hail component of reflectivity is calculated from

$$Z_{eh} = c(k_{iw})(\rho_h^2 / \rho_r^2)(\pi\rho_h)^{-1.75}(N_{oh})^{-0.75}(\rho_a q_h)^{1.75} \tag{3.12}$$

where ρ_h is the density of hail in kg m^{-3}, N_{0h} is the intercept parameter for the distribution of hail in mm^{-4}, and q_h is the hail mixing ratio in kg kg^{-1}. The reflectivity $Z(r_0,\theta_0,\phi_0)$ located within the radar sampling volume centered at range r_0, elevation θ_0 and azimuth ϕ_0 is expressed as

$$Z(r_0,\theta_0,\phi_0)=\frac{\sum_{vol} Z_{ijk}\omega_{ijk}}{\sum_{vol}\omega_{ijk}} \tag{3.13}$$

and the corresponding radial velocity ($v(r_0,\theta_0,\phi_0)$) as,

$$v(r_0,\theta_0,\phi_0)=\frac{\sum_{vol} V_{ijk}Z_{ijk}\omega_{ijk}}{\sum_{vol} Z_{ijk}\omega_{ijk}} \tag{3.14}$$

where Z_{ijk} and v_{ijk} are the model reflectivity and radial velocity respectively, at model grid point (i,j,k), and ω_{ijk} is the beam weighting function. The radial velocity V_{ijk} at model grid points are calculated from

$$V_{ijk}=u_{ijk}\sin(\phi)\cos(\theta)+v_{ijk}\cos(\phi)\cos(\theta)+(w_{ijk}+V_T)\sin\theta \tag{3.15}$$

where u_{ijk}, v_{ijk} *and* w_{ijk} are the model grid point wind components and V_T is the terminal fall speed of hydrometeors. Now, the mean reflectivity and Doppler velocity values in (3.14) and (3.15) at the center range, azimuth and elevation of the effective resolution volume within the beamwidth is approximated by computing the weighted mean of

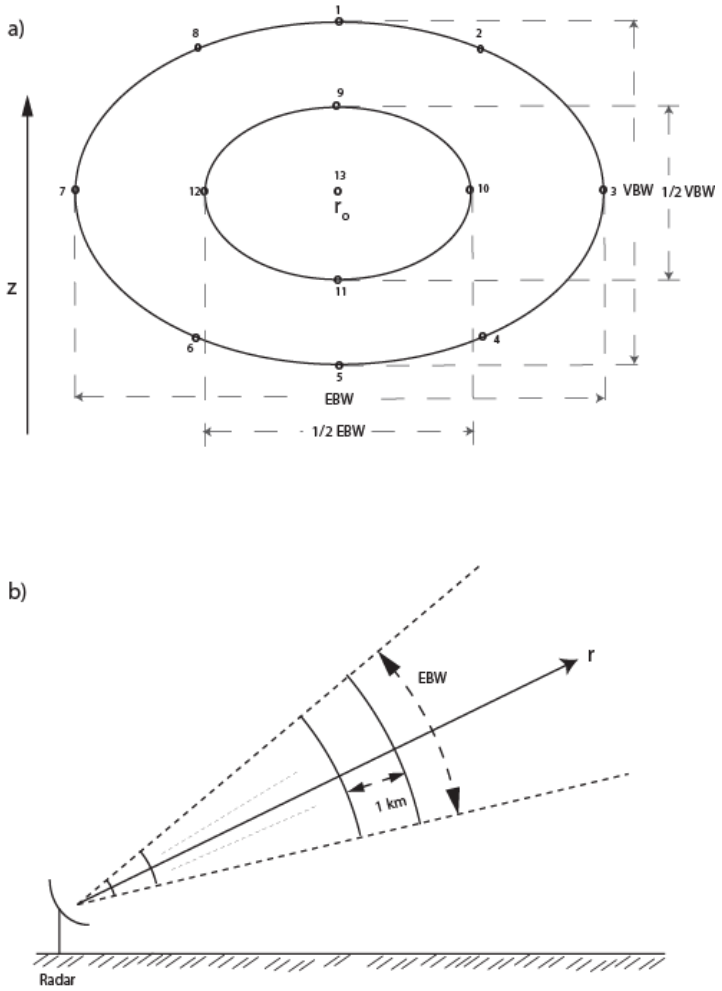

Figure 3.1 Schematic illustration of the a) vertical resolution volume and b) the horizontal resolution of the radar beam. Points 1 through 8 approximate a weight of 0.50, points 9 through 12 approximate a weight of 0.84 and center point 13 approximates a weight of 1.0 for the simplified volume averaging technique. The effective beamwidth (EBW) is 1.39 and the vertical beamwidth (VBW) is 0.89.

individual doppler velocity and reflectivity values over 13-points within the resolution volume as shown in a schematic illustration in Figure 3.1.

The 8 outer points within the volume carry a constant weight of 0.50, the 4 points in the inner ellipsoid has a weight of 0.84 while the center point has a weight of 1.0. A trilinear interpolation of the model grid points are used to obtain these 13 points in the resolution volume. Finally, the mean radar reflectivity factor Z is converted to logarithmic radar reflectivity in units of dBZ using

$$dBZ = 10 \log_{10} Z \tag{3.16}$$

To account for the measurement and sampling errors for radial velocity and reflectivity observations v_k, random numbers are drawn from a Gaussian distribution of zero mean and standard deviations of 2 ms^{-1} and 2 dBZ, respectively, and are added to the observations. The WSR-88D and PAR antenna half-power beamwidth is assumed to be 0.89^o with 1.0^o azimuth interval and a 1.39^o effective beamwidth. The Volume Coverage Pattern (VCP) 11[2] precipitation mode scanning strategy is used to scan the weather. The VCP 11 mode (Figure 3.2) consists of 14 elevation angles or sweeps, so each volume scans contains 14 sets of observations at different angles.

The radar reflectivity observations assimilated include 0 dBZ (non-precipitating) observations. A full volume scan on average contains about 3-4 times as many non-precipitation observations as the supercell storm is isolated in nature. Previous studies (Tong and Xue 2005; Aksoy et al. 2009) shows that assimilating clear air reflectivity observations (0 dBZ) helps to suppress the spurious convective cells around the main storm. The radial velocity observations are assimilated only where the observed

[2] The VCP 11 elevation angles are : 0.05^o, 1.45^o, 2.40^o, 3.30^o, 4.30^o, 5.20^o, 6.20^o, 7.50^o, 8.70^o, 10.0^o, 12.0^o, 14.0^o, 16.70^o and 19.50^o

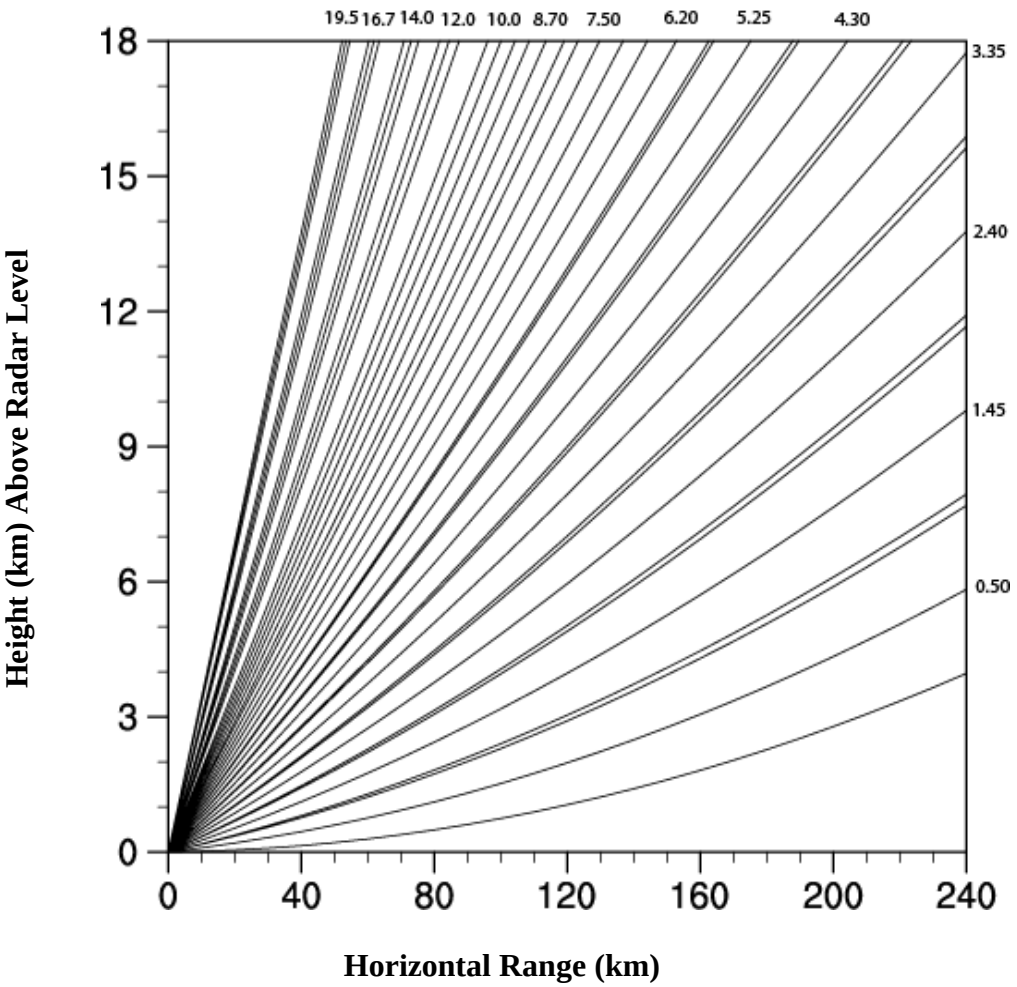

Figure 3.2 Radar scan angles for VCP 11 scanning mode. There are 14 elevation angles in this mode and the beam width is 0.89.

reflectivity values are greater than 10 dBZ. Radial velocity below 10 dBZ does not provide useful storm information and thus this threshold helps in reducing the number of radial velocity observations. A snapshot of radar reflectivity and radial velocity observations at 7.5P^{o} elevation angle (which is about 5.05 km above the ground) created using the radar emulator is shown in Figure 3.3. The radar is located at the southwest corner, outside of the computational domain of Fig 3.3a. The radar emulator captures the main feature of the storm (Figure 3.3b) even though small scale details are missing when compared to the reflectivity from the truth run (Figure 3.3a). The snapshot of the radial velocity (Fig 3.3b) at the same elevation angle shows that the wind is moving away from the radar.

While the radar data from the radar emulator is generated as realistic as possible, it is however distinct from the real radar observations. The real WSR-88D radar observations have a range gate spacing of 250 m for radial velocity and 1 km for reflectivity while the range gate spacing for PAR for both radial velocity and reflectivity are 1 km. To reduce the heavy computational burden of assimilating observation using EnSRF data assimilation technique, the reflectivity and radial velocity observations are created at a coarser 1.0-km range sampling interval instead of the 0.25 km interval available from both WSR-88D and PAR radars so that the number of observations m is less than the number of model state n. To assimilate the WSR-88D observations, synthetic radar observations are generated for 2-3 sweeps every minute rather than assuming the entire volume is collected simultaneously. Out of the 14 sweeps, the lower 12 sweeps of observations are generated 3 sweeps per minute for the first 4 min with the remaining upper 2 sweeps valid for the fifth minute of the volume scan. Observations for

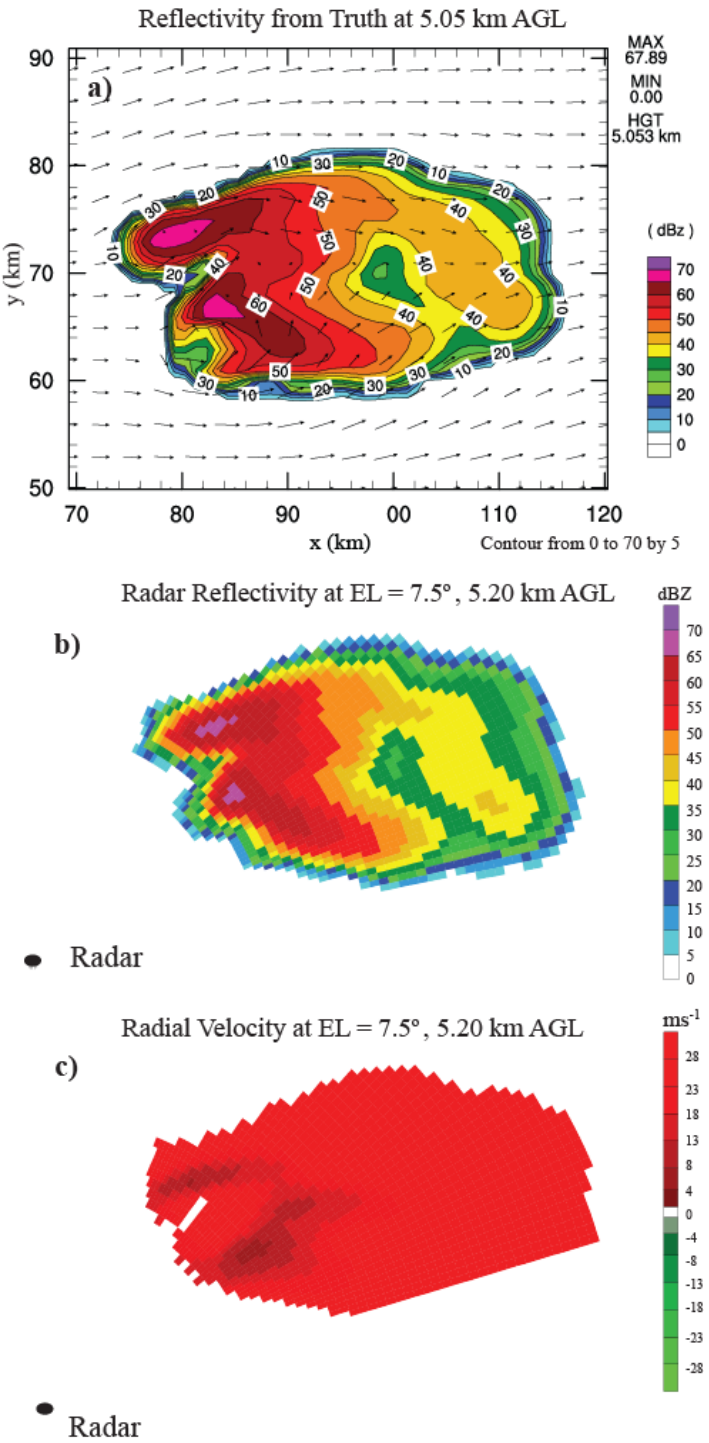

Figure 3.3 Synthetic radar observations created from (a) the truth run (model reflectivity contours in dBZ and the horizontal wind vectors in ms^{-1} at 5.053 km above ground, and the synthetic radar observations of b) reflectivity (dBZ) and c) doppler velocity (ms^{-1}) at 7.5^{o} elevation angle in spherical radar coordinates at $t = 39$ min.

PAR complete volume scan is available every 1 min, the WSR-88D observations are available every 5 minutes, with 2-3 elevations every minute (Figure 3.4).

3.4 Experimental Design

In this study, a 40 member ensemble is used to assimilate the PAR and WSR-88D observations. The domain size and grid resolution of the ensemble members are identical to the truth run. The domain of the ensemble also moves at $u = 17$ and $v = 7$ ms^{-1} following the truth run to keep the storm inside the domain. The cutoff radius for covariance estimation of the filter is 4 km in both horizontal and vertical directions. The reflectivity and radial velocity observations are assimilated in the filter serially. Each time an observation is assimilated, the ensemble mean and each of the ensemble members are updated for each model variable at each grid point within 4 km of that observation location. The 40 member ensemble forecast runs are distributed among a number of processors using the shared memory parallelization via OpenMP (Open Multi-Processing).

Initializing the Ensemble:

Each member of the 40 member ensemble is initialized from the same classic Weisman-Klemp sounding in a horizontally homogeneous environment as in the truth. To facilitate the development of storms, 7 thermal bubbles (ellipsoidal θ perturbations) at random locations within the 30 km to 70 km portion of the domain in x and y directions and within 0.25 to 2.25 km in z direction are introduced at the initialization time ($k = 0$)

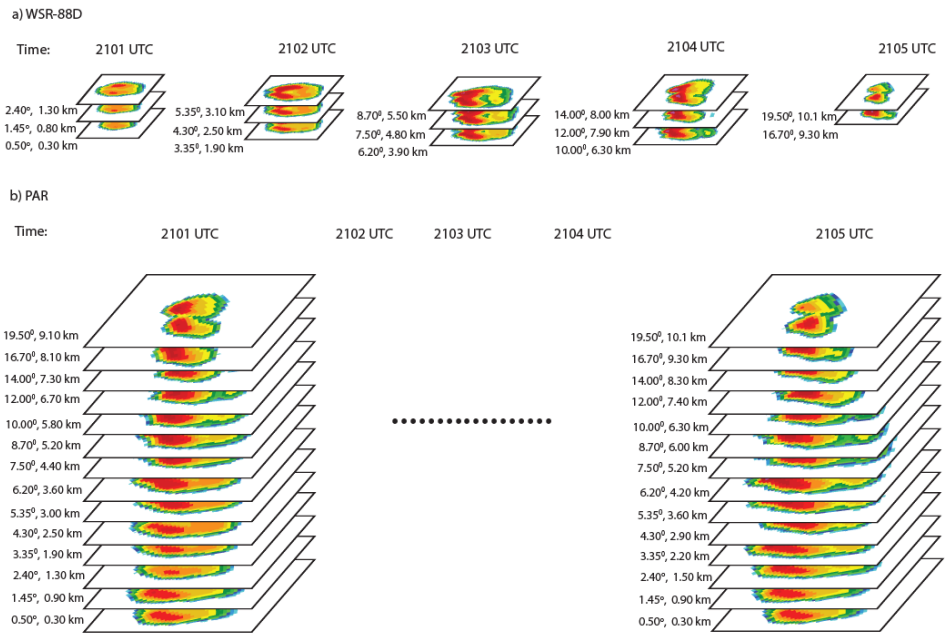

Figure 3.4 Synthetic a) WSR-88D and b) PAR radar observations using VCP 11 scanning mode during a 5-min interval staring at 2100 UTC and ending at 2105 UTC. PAR scans a complete volume of observations every minute, while WSR-88D scans 3 or 2 elevation angles every minute with a complete volume scan every 5 minutes.

to each ensemble member. The region where the bubbles are added includes the region where the synthetic radar echoes are seen later. The bubbles are 7.5 km radius in the horizontal direction and 2.0 km radius in the vertical direction. The magnitude of the θ perturbations at the center of the ellipsoid is 1.5 K and the magnitude decreases to 0 at the edge. Each perturbation is positive, and the perturbations are additive in locations where they overlapped. The ensemble members thus differ from each other in the location and magnitude of the thermal bubbles but have an identical base environment. The bubbles for 4 different ensemble members are shown in Figure 3.5. This method of initialization is very helpful as the thermal bubbles initiate convective cells and produce the covariance information needed for the ensemble to successfully assimilate the radar data. Many of the cells are spurious in that they are outside the domain of radar observations and survive throughout the assimilation. However, the assimilation of clear air observations (0 dBZ) suppresses the unwanted spurious convective cells around the main supercell.

The data assimilation procedure for the EnSRF is as follows:

1) First create an initial 40 member ensemble of model states at time k = 0 min.
2) Advance the ensemble to a make forecasts to the first observations time k = 25 min. During this time, the ellipsoidal θ perturbations (within the 40 x 40 km wide and 2 km tall portion of the domain) initiate convective cells in the ensemble members.
3) Assimilate observations serially using (2.50) and (2.51) until all the observations valid at time k = 25 min are assimilated.
4) Advance the ensemble to make a forecast for 1 min to the next observations time at k =26 min.

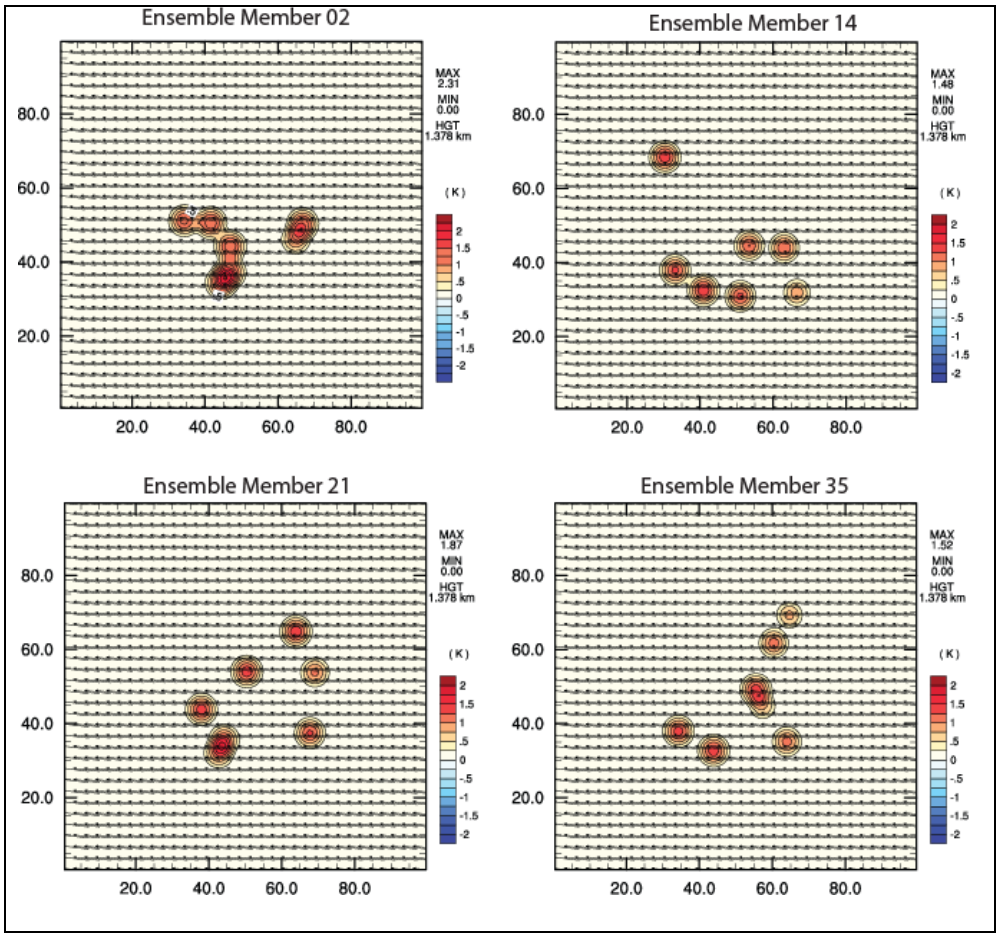

Figure 3.5 Temperature perturbations (bubbles) of ensemble members 2, 14, 21 and 35 at 1.4 km above the ground. The bubbles are added to a 40x40 km wide portion of the 100x100 km domain.

5) Repeat step 3 and 4 until all available observations are assimilated.
6) After assimilating all the observations, the ensemble members are set to make forecasts.

The model variables updated by the filter are u, v, w, θ, q_v, q_c, q_r, q_i, q_s and q_h. Comparable assimilation results are obtained whether the filter is allowed to update k_m and π or not. Therefore, to reduce computational time, these two variables are not updated. One aspect of the EnSRF data assimilation scheme is the tendency for ensemble spread to become too small. This is due to limited ensemble size and model errors. To help maintain the storm and ensemble spread in the model during the assimilation cycles, a random number of thermal bubbles (ellipsoidal θ perturbations) are added to the members near the storm locations where the difference between the observed and ensemble mean reflectivity field exceeds 30 dBZ. The thermal perturbations have a temperature excess of 1.5 K at the center of the ellipsoid that decrease to zero at a horizontal radius of 7.5 km and vertical radius of 2 km. A 5-min interval is used between the thermal perturbations for WSR-88D observations assimilation to correspond roughly to the time between complete volumetric radar scans. For PAR observations assimilation the time interval is 2-min.

3.4.1 60-min Assimilation

The first experiment assimilates 12 volume scans of storm observations from a WSR-88D radar. It takes 1 h for a WSR-88D to produce the 12 volume scans, while during this time period PAR can produce 60 volume scans of observations of the same storm (Fig 3.6). After 60 min of data assimilations starting at k = 25 mins and ending at k

= 85 mins, the ensemble members are set free to make a forecast for the next 35 minutes. During the first volume scan, the radars are located inside the model domain, southwest relative to the storm. However the storm motion is away from the radar, such that the radars are located outside the computational domain to the west-southwest of the supercell during the last volume scan. The objectives of this experiment are to evaluate if the analyses obtained from assimilating observations for a relatively longer period of time (60-min) perform as expected and to verify that the EnSRF system is stable.

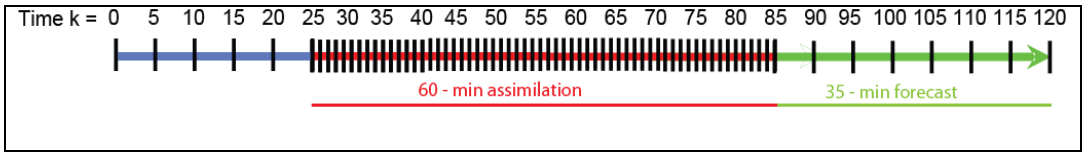

Figure 3.6 Schematic illustration of the EnSRF experiment for 60-min assimilation.

3.4.2 15-min Assimilation

Unlike the previous experiment, this experiment assimilates radar observations for a 15-min period starting at t = 25 min and ending at t = 39 min. During this 15-min assimilation period, 15 volume scans of PAR observations and 3 volume scans of WSR-88D observations are assimilated. After 15 min of data assimilation, the ensemble members are used to produce a 50 min forecast. After initializing the ensemble members at k = 0, the members are integrated forward in time for k = 25 min before the assimilation of the first observations. The 40 ensemble members from the last assimilation cycle are set to make an ensemble of forecasts for 50 min. A schematic illustration of the time frame of the experiment is shown in Figure 3.7. The radar is located at x = 3.6 km and y = 4.9 km off of the southwest corner of the domain during the

first volume scan. The initialization and other ensemble configuration details are identical to the previous experiment.

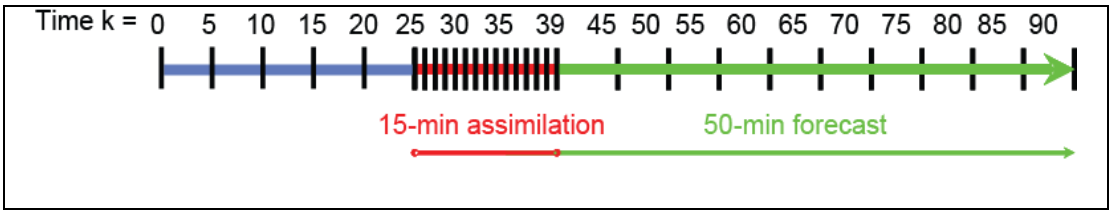

Figure 3.7 Schematic illustration of the EnSRF experiment for 15-min assimilation.

3.5 Results

The accuracy of the analyses and forecasts from PAR and WSR-88D observation assimilations are evaluated using both statistical and graphical comparison of the ensemble mean analyses and forecasts to the truth run. Since the objective is to evaluate how well the supercell is captured in the analyses and determine accurate forecasts when using the analyses as initial conditions, the analyses and forecasts errors are calculated only in areas where there is precipitation. To do this, values are averaged over only those model grid points where the total precipitation mixing ratio (sum of rain q_r, snow q_s, ice q_i and hail q_h mixing ratios) is greater than 0.10 g kg^{-1}. Statistical measures include the root-mean-square error (rms) of the unobserved variables of *u, v, w, t* and total precipitation mixing ratios calculated as the difference between the truth and the ensemble mean analyses.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(F_i - O_i)^2} \tag{3.16}$$

here F_i is the ensemble mean analyses, O_i is the observation, *i is* the index for the number of model grid points.

The ultimate goal of storm-scale data assimilation is to increase warning lead times by obtaining more accurate short term forecasts of severe storms events. Thus, to evaluate the accuracy of the forecasts from both PAR and WSR-88D observation assimilation, the 40 analyses from the last assimilation cycles are used as the initial conditions for each of the ensemble members and short-term forecasts are produced. The ensemble mean forecasts are then compared with the truth run.

3.5.1 Analyses

A. 60-min Assimilation

The rms errors of the ensemble mean analyses from assimilating PAR observations have a larger decrease in the rms errors for *u*, *v*, *w* wind components, temperature and total precipitation mixing ratios compared to the WSR-88D observation assimilation during the first 30-min of the assimilation period (Figure 3.8). This result is not surprising as the PAR assimilation is using 5 times more observations over the same time interval. By the end of the 60-min assimilation cycle, the magnitude of the rms errors from both assimilations become close to each other. Horizontal plots of reflectivity and vertical vorticity at the last assimilation cycle (t = 84 min) from PAR and WSR-88D observation assimilation shows that both observations capture the split supercell structure and the developing hook echo rather accurately (Figure 3.9).

The maximum reflectivity and vertical vorticity and its extent also are comparable between the two and closely match the truth. The 60-min long assimilation cycle

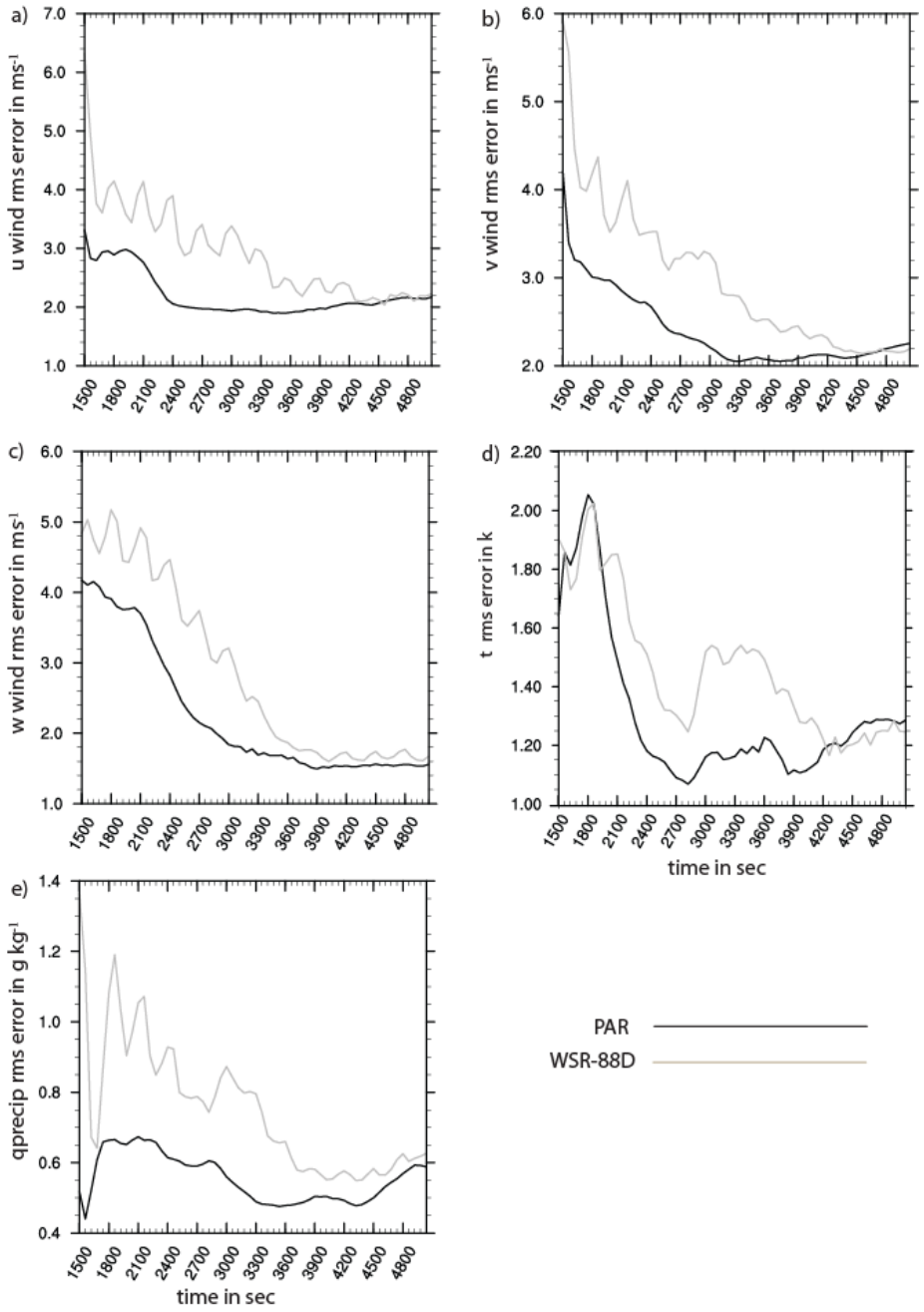

Figure 3.8 The rms errors of ensemble mean analyses vs. time(s) for the 60-min assimilation experiment starting at t = 25 min and ending at t = 84 min for (a) u (msP^{-1}), (b) v (ms^{-1}), (c) w (ms^{-1}), (d) t (k) and (e) total precipitation mixing ratios (g kg^{-1}) for PAR (black lines) and WSR-88D (gray lines) observations assimilation. Values are averaged over the domain at grid points where the total precipitation mixing ratios (sum of *qr, qh* and *qs*) is greater than 0.10g kg^{-1}. Note that 300s = 5.

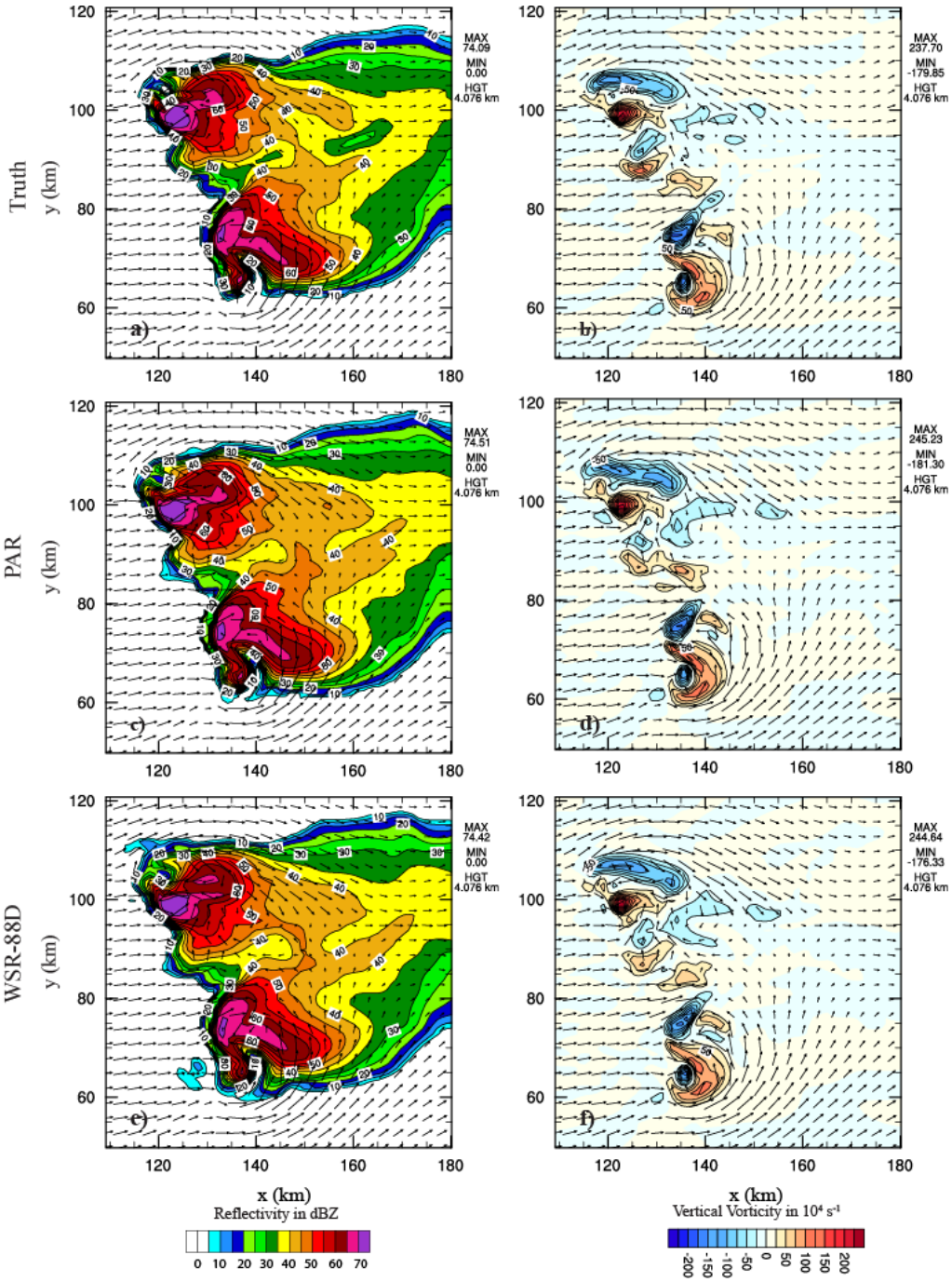

Figure 3.9 Reflectivity and vertical vorticity at 4.076 km above ground at the 60-min assimilation time (t=84 min) from (a and b) truth run and ensemble mean analyses from (c and d) PAR observations and (e and f) WSR-88D observation assimilation.

suppresses almost all spurious convection in the ensemble members through the assimilation of non-precipitating observations for both PAR and WSR-88D observation assimilation. Overall, the results from this experiment supports the conclusion drawn in earlier studies that 10 or more volume scans of radar observation assimilation generates very accurate analyses of severe storm events (Tong and Xue 2005; Xue et al. 2006).

B. 15-min Assimilation

The rms errors from both PAR and WSR-88D observation assimilations are seen to decrease rapidly for all variables (Figure 3.10). However, the faster volume scan of PAR observation generates significantly smaller rms error compared to the WSR-88D assimilation for all variables. The increase and decrease (zig-zag pattern) in the error curve from assimilating WSR-88D observations are more distinct than the PAR error curve and corresponds to the error from assimilating observations during the 5-min long volume scans. The reflectivity and vertical velocity structure of the supercell storm in mid-levels from PAR observation assimilation more closely resembles the truth than that of the WSR-88D observation assimilation (Figure 3.11). The PAR ensemble-mean analyses captures the location, structure and the strength of the two main precipitation cores as in the truth, while the WSR-88D analyses fail to capture the high-reflectivity core of the northern cell and barely captures the high-reflectivity core of the southern cell. In addition, while a number of spurious cells still surround the main supercell in the WSR-88D analyses, the more frequent observations assimilation from PAR suppresses most of the spurious convection. This result reinforces the conclusion that the frequent assimilation of 0 dBZ reflectivity observations is indeed very helpful in suppressing

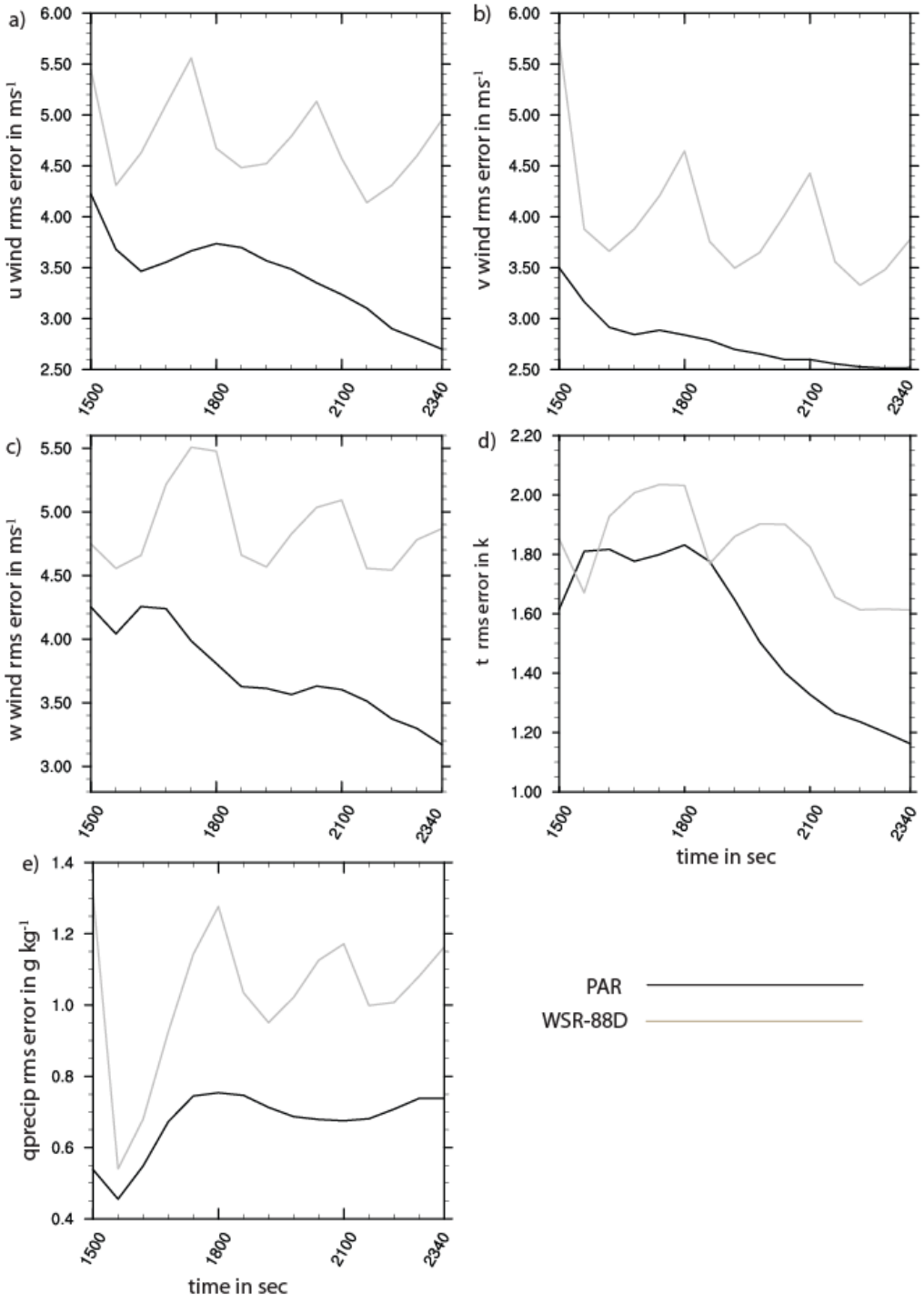

Figure 3.10 Same as in Figure 3.8 but for the experiment with 15-min assimilation period starting at t = 25 min and ending at t = 39 min.

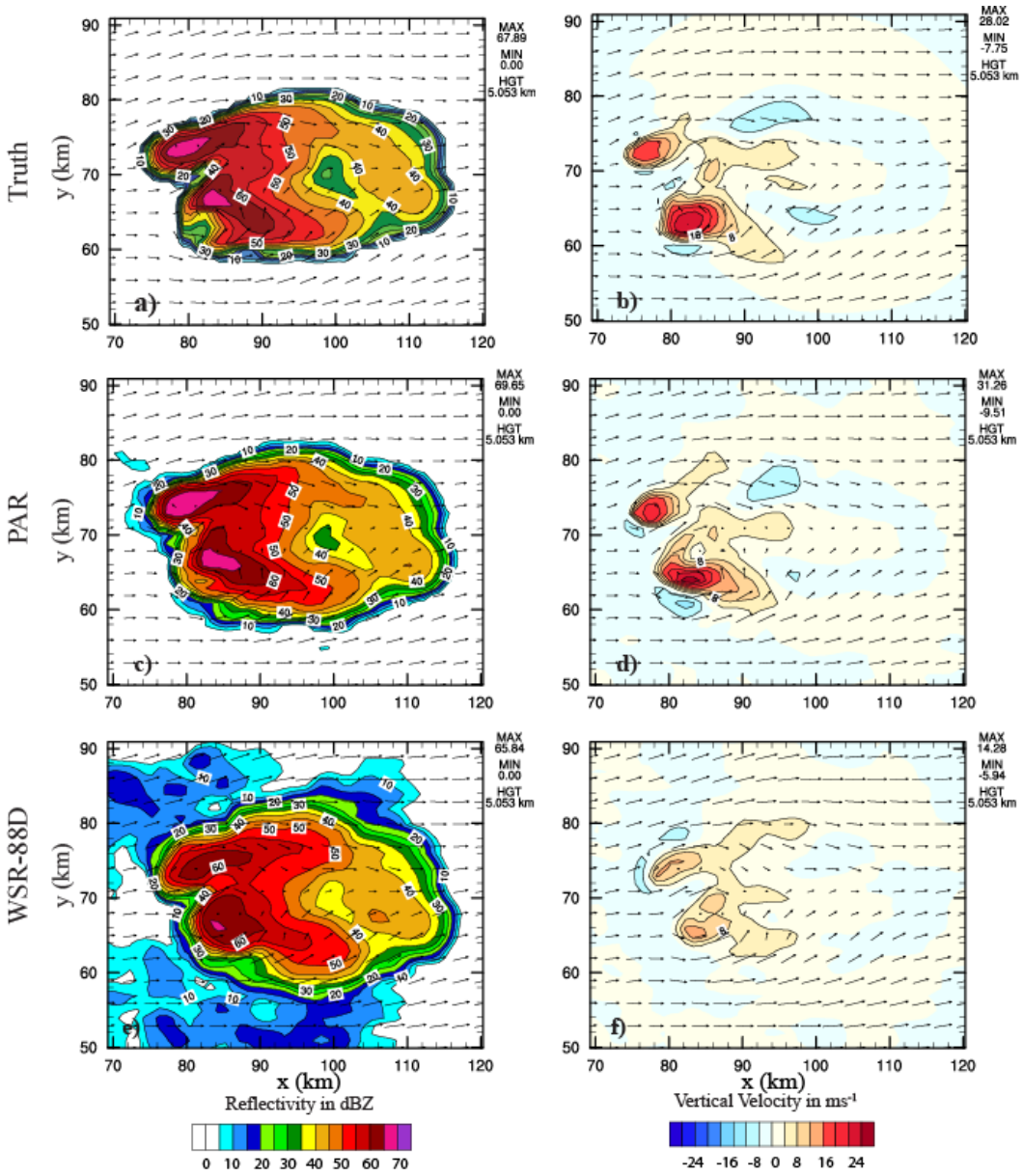

Figure 3.11 Same as in Figure 3.9 but for a 15-min assimilation period for reflectivity and vertical velocity contours at the last assimilation cycle (t=39 min) 5.053 km above ground.

spurious convection. Furthermore, the two strong updrafts in excess of 16 ms^{-1} from the northern and southern cells (Figure 3.11b) in the truth are well represented in the PAR analyses (Figure 3.11d), while the WSR-88D analyses (Figure 3.11f) fail to capture the location, structure and the strength of the updrafts. While the maximum updraft from the WSR-88D assimilation is 14.28 ms^{-1}, the maximum updraft from PAR observation assimilation and the truth is 31.26 and 28.02 ms^{-1}, respectively. Similar results also are found for other variables at other vertical levels of the model domain. These results clearly show the benefit of assimilating faster volume scan observations for capturing the split supercell structure of the storm in the analyses resulting in a more accurate depiction of severe weather events.

3.5.2 Forecasts

A. 60-min Assimilation

Figure 3.12 shows the rms errors from the ensemble mean forecasts averaged over the domain where the total precipitation mixing ratio exceeds 0.10g kg^{-1} during the 35-min forecast period. The rms error grows rapidly during the forecast period from both PAR and WSR-88D observation assimilation as expected. While the rapidly growing forecasts rms errors from PAR observations assimilation are smaller than that of the forecasts WSR-88D observations for the first few minutes, the errors from the PAR observations exceeds the WSR-88D errors and remains larger for the rest of the forecast period. This is true for all the variables shown in Figure 3.12. The basic structure and the evolution of the storm, including the split storm cells and the hook echoes from the ensemble mean 15-min forecasts (Figure 3.13) are rather accurate. However, the 15-min.

forecasts reflectivity contours at 3.1 km AGL from both PAR (Fig 3.13 b) and WSR-88D (Fig 3.13 c) observation assimilation more closely matches each other than to the truth (Fig 3.13 a). Similar results also are seen from other variables (not shown).

B. 15-min Assimilation

The rms errors of the ensemble mean forecasts show that the rms errors grow rapidly during the forecast period from both PAR and WSR-88D observation assimilation as expected (Figure 3.14). However the forecast errors from PAR observation assimilation are significantly smaller than the forecast errors from WSR-88D observation assimilation for the entire 50-min forecast period. The reflectivity contours from the truth simulation and 5-min forecast at 6.1 km AGL and 20-min forecasts at 2.1 km from PAR and WSR-88D observation assimilation indicate that the forecasts from PAR observation assimilation maintains the strength, split storm cell structure and location of the two main precipitation core more closely to the truth than that of the WSR-88D forecasts (Figure 3.15). Thus, the more accurate analyses from the PAR observation assimilation yields better forecasts compared to the WSR-88D forecasts.

3.6 Summary

The EnSRF data assimilation method is implemented to assimilate radar observations for a shorter assimilation period using the perfect model framework. The synthetic reflectivity and radial velocity WSR-88D and PAR observations are created from a truth simulation of a supercell storm at a coarser 1 km range resolution. The

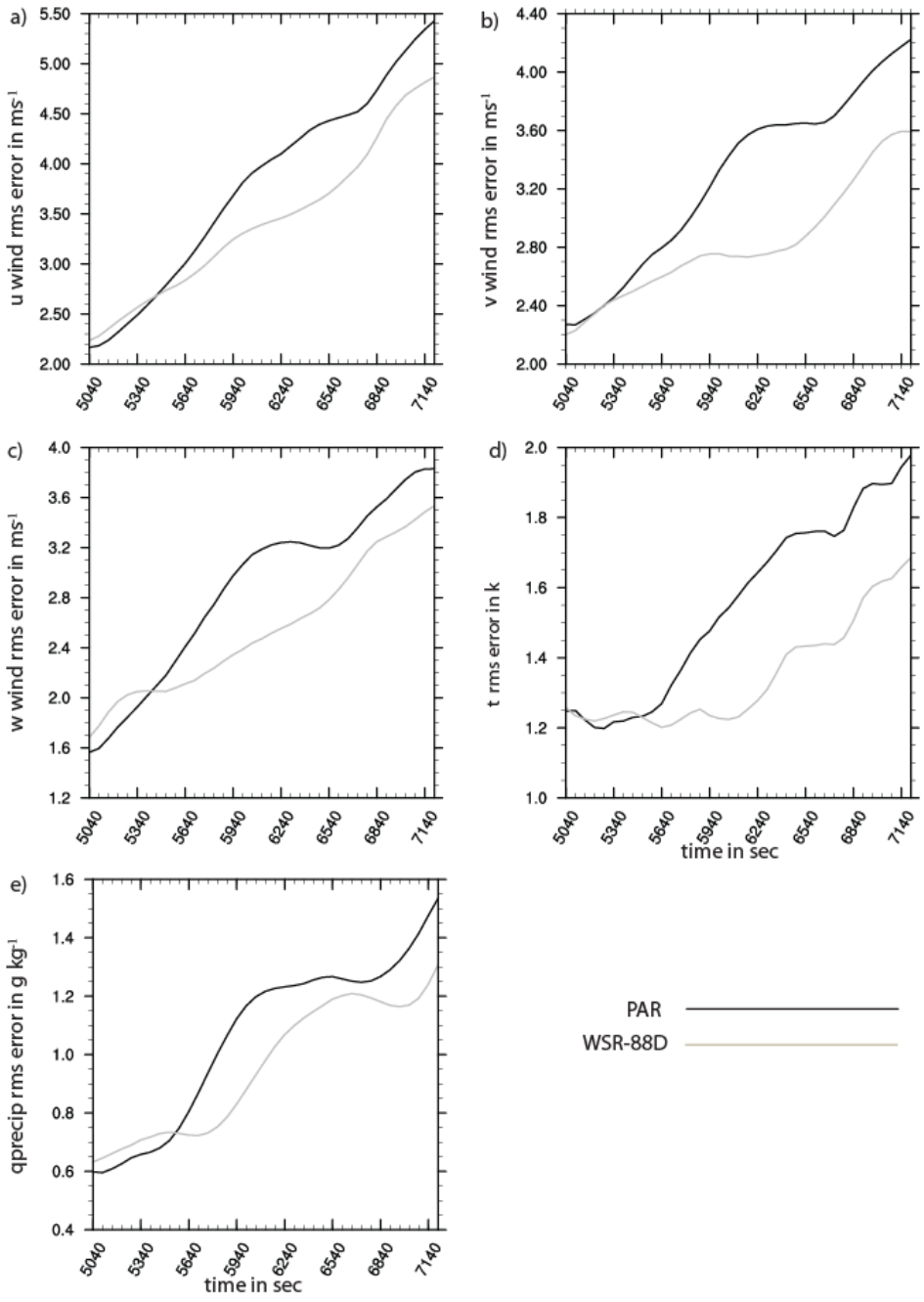

Figure 3.12 The rms errors of ensemble mean forecast from the 60-min assimilation experiment during the 35-min forecast period starting for (a) u (ms^{-1}), (b) v (ms^{-1}), (c) w (ms^{-1}), (d) t (k) and (e) q ($g\ kg^{-1}$). Values are averaged over the domain where the total precipitation (sum of q_r, q_h , q_i and q_s mixing ratios) is greater than $0.10g\ kg^{-1}$. Details are shown in the legend.

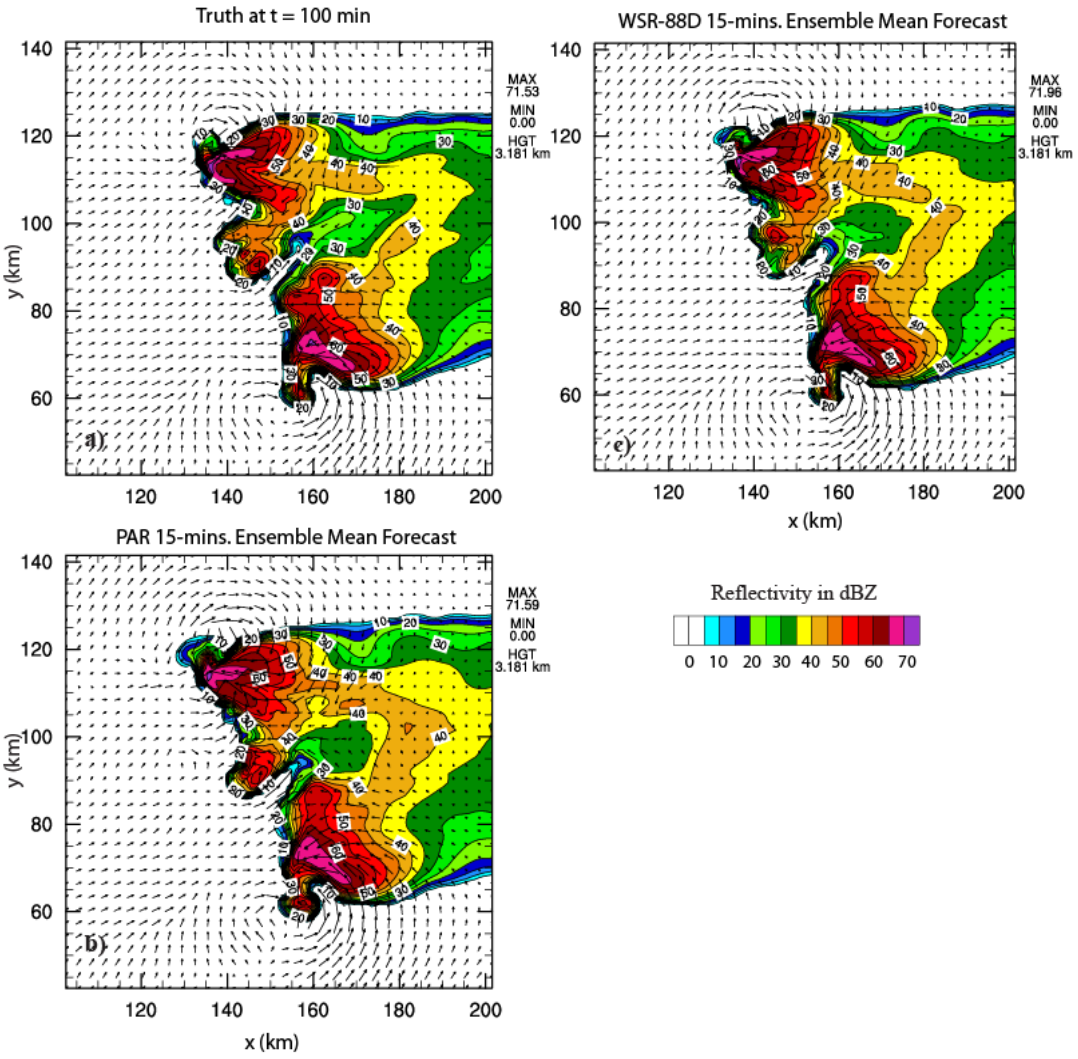

Figure 3.13 Reflectivity contours at 3.18 km AGL for (a) truth and 15-min ensemble mean forecasts from (b) PAR observations assimilation and (c) WSR-88D observations assimilation from the 60-min assimilation experiment.

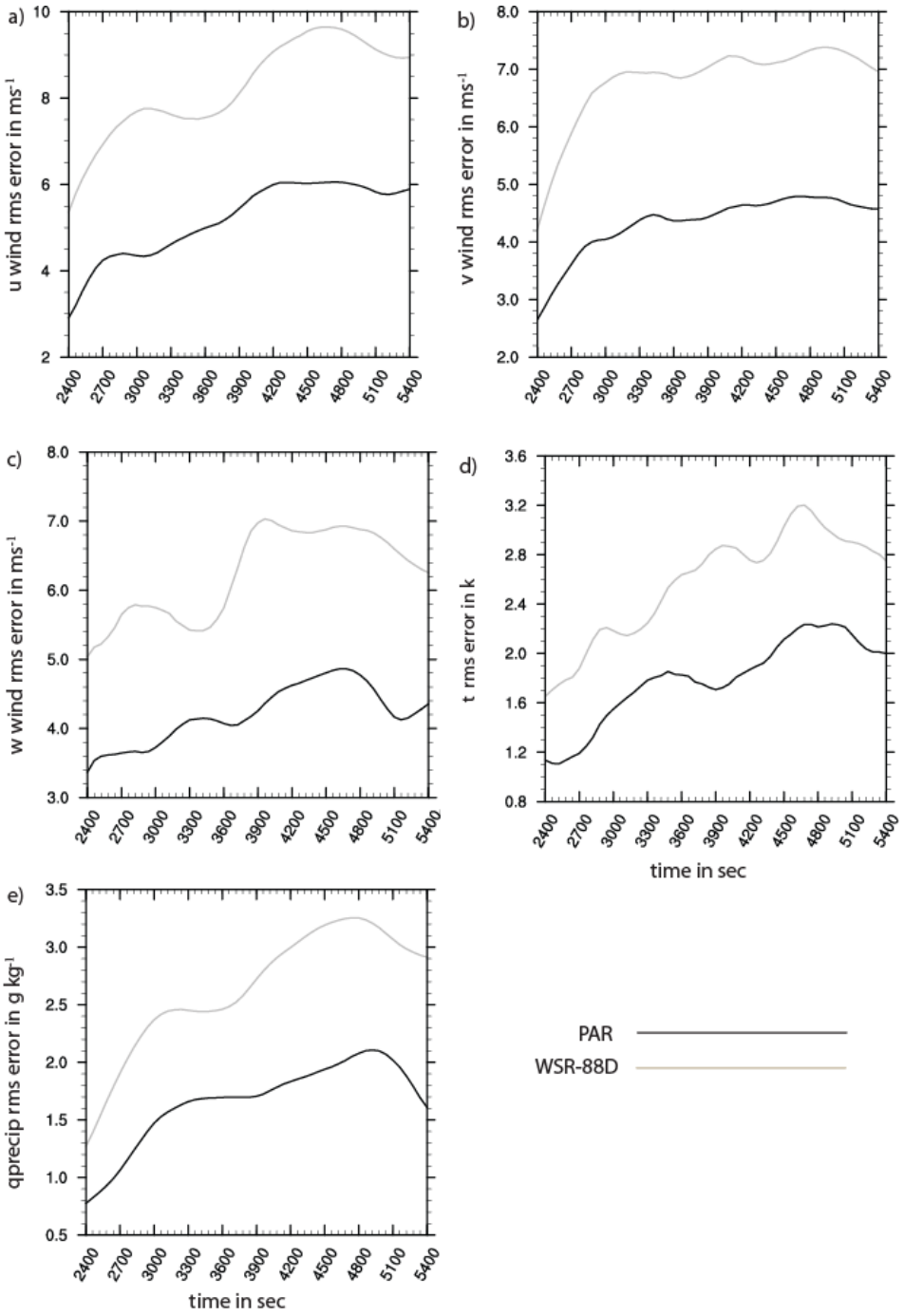

Figure 3.14 The rms errors of ensemble mean forecast from the 15-min assimilation experiment during the 50-min forecast period for (a) u (ms^{-1}), (b) v(ms^{-1}), (c) w(ms^{-1}), (d) t(k) and (e) q (g kg^{-1}). Values are averaged over the domain where the total precipitation (sum of *qr, qh* and qs mixing ratios) is greater than 0.10g kg^{-1}. Details are shown in the legend.

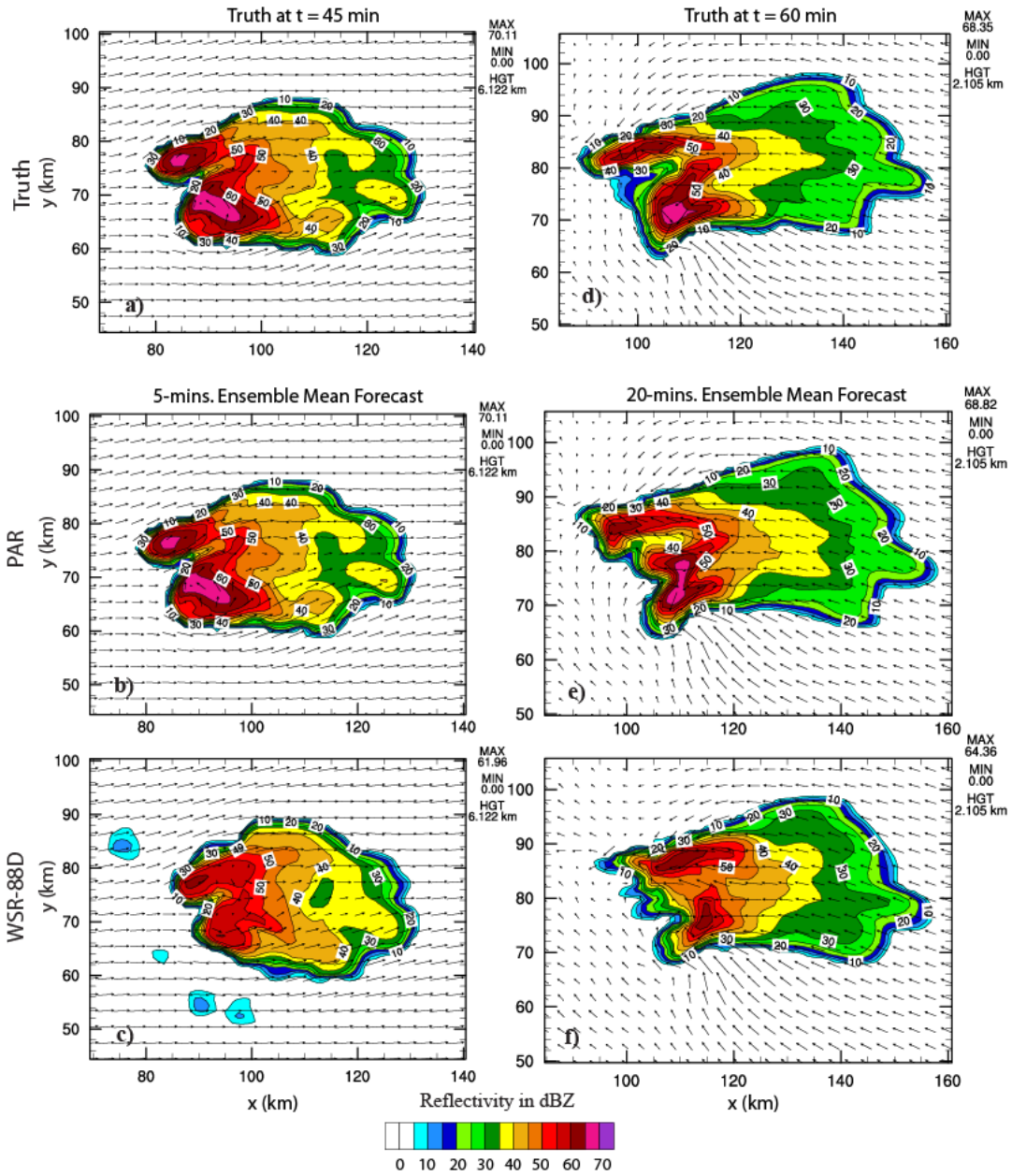

Figure 3.15 Reflectivity contours for (a and d) truth and forecasts from the 15-min assimilation experiment from (b and e) PAR observations assimilation and (c and f) WSR-88D observations assimilation, (b) and (c) are 5 min ensemble mean forecast while (e) and (f) are 20 min ensemble mean forecasts.

analyses and forecasts from both radar observations assimilation experiments are compared and presented.

The results indicate that both PAR and WSR-88D observation assimilation over a 60-min assimilation period produce qualitatively similar analyses of the supercell and match closely to the truth. However, results from assimilating radar observations for a shorter assimilation period of 15-min show that PAR observations provide more accurate analyses and forecasts of the depiction of supercell compared to the WSR-88D data. Results also indicate that the more frequent PAR observations assimilation is able to suppress most of the spurious cells in regions around the storm during the shorter 15-min assimilation period with more accurate depiction of the two precipitation cores and generates smaller rms errors for unobserved variables of winds, temperature and precipitation mixing ratios compared to those from the WSR-88D observation assimilation. There is a rapid increase in rms errors in both PAR and WSR-88D ensemble mean forecasts during the 50-min forecast period, but the errors for PAR observation assimilation are consistently smaller than for WSR-88D observation assimilation. These results signify the benefits of more frequent data coverage in a shorter period of time on the quality of the storm analyses and forecast, i.e. the more complete the storm observations, the better analyses and forecasts. However caution is warranted as the results obtained from this study may be too optimistic since the experiments are based on a perfect model assumption where model error does not play a role. To present the impact of model error in radar data assimilation, imperfect model experiments are conducted in the next chapter and compared with the perfect model data assimilation results.

Moreover, the value of using a variety of intercept and density parameters within the same microphysics scheme also is examined.

Chapter 4

Data Assimilation Using Ensemble Square-Root Filter: Perfect and Imperfect Model Experiment

4.1 Introduction

The experiments conducted in Chapter 3 are based on perfect model assumption where both the truth simulation and the ensemble members for the data assimilation experiments use the same LFO microphysics scheme with predefined constant parameters for precipitation particles. Since both the truth and the ensemble experiment use the same microphysics scheme, model errors do not play a role. However, such a good performance obtained in Chapter 3 is not expected in real-world experiments where the forecast model unavoidably has errors. Therefore to evaluate the impact of model error in EnSRF radar data assimilation, experiments need to be conducted under imperfect-model scenarios to account for model error.

One of the major sources of error in storm-scale data assimilation and forecasts is the microphysical parameterization scheme used in the model to represent the microphysical characteristics of the storms (Dowell et al. 2004; Gilmore et al. 2004; van den Heever and Cotton 2004; Dowell and Dowell 2009; Snook and Xue 2008; Tong and Xue 2008a). Microphysics schemes represent a number of different phase changes of water species and a number of different interactions between cloud and precipitation particles, requiring many assumptions to make these schemes both realistic and computationally affordable (Stensrud 2007). The most commonly used type of microphysical scheme in storm-scale modeling is the single-moment bulk microphysics

scheme (Lin et al. 1983; Tao and Simpson 1993; Schultz 1995; Straka and Mansell 2005; Hong and Lim 2006) that predicts only the particle mixing ratios of the hydrometeors. A single-moment scheme uses constant values for the intercept parameters and the densities of hydrometeors in the calculation of hydrometeor size distributions, and these intercept and density parameters are defined in the experiments somewhat arbitrarily. However, several observational studies indicate that the particle densities and the intercept parameters of hydrometeor distributions can vary widely among storms and even within a single storm (Gunn and Marshall 1958; Houze et al. 1979, 1980; Mitchell 1988; Pruppacher and Klett 2000; Cifelli et al. 2000; Brandes et al. 2007). Several sensitivity studies also demonstrate the impact of the variations of particle parameters on storm structure, intensity and precipitation characteristics (Gilmore et al. 2004; van den Heever and Cotton 2004; Snook and Xue 2008). Thus, applying predefined constant parameters for precipitation particles in storm-scale model cannot adequately represent the highly uncertain thunderstorm precipitation characteristics and can lead to significant errors in the analyses and forecasts of severe storms.

However, determination of suitable values for the microphysical parameters in storm scale data assimilation is very difficult due to the unavailability of in situ microphysics observations. Since the selection of microphysical parameters in storm-scale modeling has profound impact on the analyses and forecasts of severe weather events, and an arbitrary selection of those parameters may lead to significant error, one approach to account for the uncertainty in a storm-scale ensemble modeling system is to vary the microphysical parameters within the same microphysics scheme among the

ensemble members. The hope is that by using a variety of realistic precipitation particle parameters, an ensemble is more likely to span the truth.

Therefore in an effort to explore the impact of model errors and also the variations in parameters within the same microphysics scheme in storm-scale forecasting, OSSEs are conducted applying both initial condition variations and a range of different realizations of the intercept and density parameters using an EnSRF data assimilation technique. The first set of experiments is based on the assumption of a perfect model in which both the truth simulation and the ensemble system use the same microphysics scheme. The second set of experiments is based on imperfect model assumptions in which the microphysics scheme for the truth simulation and the microphysics scheme for the assimilation system are different. Thus, the imperfect model assumption includes error in the forecast models from the microphysical parameterization. This chapter is the basis for the paper Yussouf and Stensrud (2010b) that is currently in review.

The experimental design of this study is described in Section 4.2. Section 4.3 presents the results obtained from the EnSRF analyses and forecasts, followed by a final discussion in Section 4.4.

4.2 Experimental Design

Two simulations of a splitting supercell storm similar to the truth run in Chapter 3 are generated using two different microphysics schemes. The domain and the initialization of the 2-h long truth runs are inherited from Chapter 3. Synthetic radial-velocity and reflectivity WSR-88D observations are then constructed from these truth

solutions using the same radar emulator as in Chapter 3. The ensemble design for this study also is inherited from Chapter 3 with some modifications presented in this section.

A. The two truth simulations and synthetic radar observations

The first truth simulation applies the Gilmore et al. (2004) version of the Lin-Farley-Orville (Lin et al. 1983) single-moment bulk microphysics scheme (Truth_LFO). The LFO scheme contains three ice categories (ice crystals, snow and hail/graupel) and calculates the mixing ratios of six water species: water vapor, cloud water, cloud ice, rain, snow and hail/graupel. In the LFO scheme, the term hail is used to represent high density graupel, ice pellets, frozen rain and hailstone. The second truth simulation applies the 10-ICE (Straka and Mansell, 2005) single-moment bulk microphysics scheme (Truth_10ICE). It has the same two water particle categories (cloud water and rain) as the LFO scheme, but includes ten ice categories (i.e., 6 graupel and hail categories, 3 ice categories and snow) that are characterized by habit, size and density. The extra ice hydrometeor categories that are included in the 10ICE scheme better represent the range of precipitation ice characteristics in a deep convective storm. Both LFO and 10-ICE microphysics schemes assumes a monodisperse particle size distribution for cloud water and cloud ice and approximate an inverse exponential form (Marshall and Palmer 1948) for the particle size distributions of rain and ice categories as follows:

$$n_x(D) = n_{0x} e^{-\lambda_x D_x} \tag{4.1}$$

where x is rain or ice categories, D is the particle diameter (m), n is the number of particles per unit volume (m^{-4}), λ is the slope parameter that defines the decrease in particle counts as diameter increases (m^{-1}) and n_{0x} is the intercept parameter that defines

the maximum number of particles per unit volume at $D = 0$ size. The slope parameter varies with mixing ratio and is given by

$$\lambda_x = \left(\frac{\pi \rho_x n_{0x}}{\rho q_x} \right) \tag{4.2}$$

where ρ_x is the density of the particle, ρ is the air density, and q_x is the mixing ratio. From (4.1) and (4.2), it is obvious that the particle size distribution is strongly influenced by the selected values of n_{0x} and ρ_x. The values of the density and the intercept parameters used for the truth simulation from the two microphysics scheme are given in Table 4.1.

Table 4.1 The intercept and the density parameters of the precipitation particles for the Truth_LFO and Truth_10ICE simulations.

LFO Scheme			10 ICE Scheme		
Catagory	Intercept m^{-4}	Density kg m^{-3}	Catagory	Intercept m^{-4}	Density kg m^{-3}
Hail/Graupel	$4x10^4$	900	Graupel (low)	$4.0x10^5$	300
Snow	$3x10^6$	100	Graupel (medium)	$2.0x10^5$	500
Rain	$8x10^6$	1000	Graupel (high)	$1.0x10^5$	700
Ice	-	-	Frozen drops	$4.0x10^5$	800
			Small hail	$4.0x10^4$	800
			Large hail	$1.0x10^3$	900
			Snow	$8x10^6$	100
			Rain	$8x10^6$	1000
			Rimed ice	$1.0x10^8$	300
			Plate ice	-	900
			Column ice	-	900
			Cloud droplets	-	1000

The truth runs from the two microphysics schemes produce a similar supercell storm, but with differences in the location, strength and structure of the storm (Figure 4.1). The cold pool at the lowest model level from Truth_10ICE (Figure 4.1b) is colder than the cold pool from Truth_LFO (Figure 4.1a). The high-reflectivity core of the

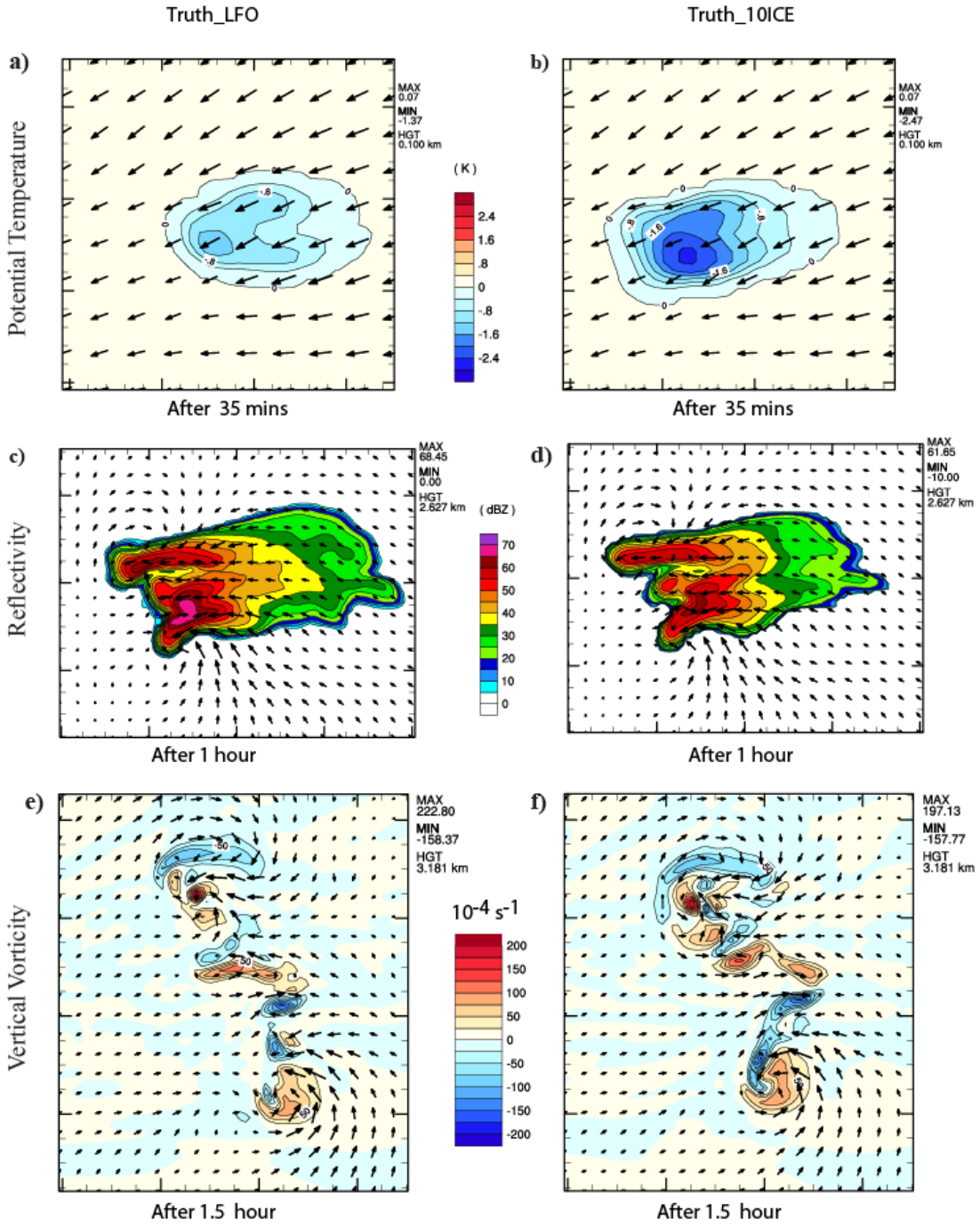

Fig 4.1 Potential temperature (K) at t = 35 min of the simulation at the lowest model level (100 m AGL) (a and b), reflectivity (dBZ; c and d) 2.6 km AGL at t = 1 hr and vertical vorticity (s^{-1}; e and f) at 3.1 km AGL at t = 1.5 hr from the truth simulation using the LFO and 10 ICE microphysics scheme.

southern cell from the Truth_LFO (Figure 4.1c) is more intense than the reflectivity core of the southern cell from Truth_10ICE (Figure 4.1d) and the mid-level vertical vorticity fields also differ from each other (Figure 4.1 e and f). Similar differences also are found for other variables at other vertical levels of the model domain and at other simulation times.

B. The ensemble configuration and OSSE design

As mentioned earlier, the 40-member ensemble for the experiment is similar to the ensemble in Chapter 3. To facilitate the development of storms, 3 thermal bubbles (1.5 K maximum ellipsoidal θ perturbations) are introduced at the initialization time (t = 0) to each ensemble member following Synder and Zhang (2003), and Dowell et al. (2004a, b). These bubbles have 7.5 km (2.0 km) radius in the horizontal (vertical) direction are placed at random horizontal locations within 10 km of the domain center and between 0.25 to 2.25 km in z direction.

After initializing the ensemble members at t = 0, the members are integrated forward in time for 25 min before assimilation of the first observations. A 30 min long assimilation period starts at t = 25 min and ends at t = 54 min. During this assimilation period, 6 volume scans of WSR-88D observations are assimilated. The radar is located at x = -3.6 km and y = -4.9 km from the southwest corner of the domain during the first volume scan. The observations valid within 1 min of the current time are assimilated followed by advancing the ensemble members 1 min to the next observation time. No additional localized perturbations (Dowell and Wicker 2009) or covariance inflations (Snyder and Zhang 2003; Dowell et al. 2004a; Tong and Xue 2005) are added to the

members to maintain the ensemble spread during the assimilation cycles. After the 30 min of data assimilation, all of the 40 ensemble members from the last assimilation cycle are used to produce a 1-h long ensemble of forecasts (Figure 4.2). Two sets of experiments are implemented in this study to assess the benefits of a multi-parameter ensemble system.

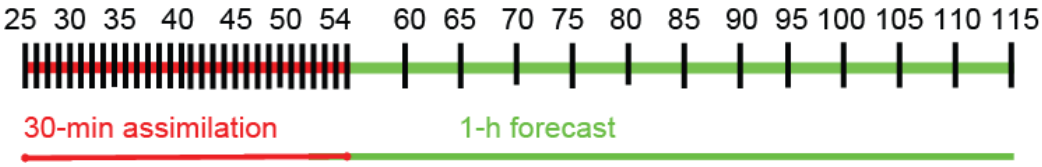

Figure 4.2 Schematic illustration of the EnSRF experiment for 35-min assimilation.

4.2.1 Perfect Model Experiment

The ensemble members use the LFO microphysics scheme and the synthetic WSR-88D reflectivity and radial velocity observations assimilated are generated from the Truth_LFO. Two experiments are conducted using these identical background environments. The first ensemble (Perfect_Control) uses the same constant intercept and density parameters for the hydrometeor categories for all ensemble members as in the Truth_LFO. The values of the parameters for the Truth_LFO and Perfect_Control experiments are the same typical values used in Lin et al. (1983) as listed in Table 1 and these values generate an intense storm with high density hail. The ensemble members in the Perfect_Control experiment thus have the identical base environment and microphysics scheme as in the truth but differ from each other in the location and magnitude of the thermal bubbles. Here we are assuming that the model is perfect and the

environmental condition is perfectly represented thereby giving the ensemble data assimilation and forecast system the best chance to produce excellent results.

The second ensemble (Perfect_MP) also uses the LFO microphysics scheme but instead of using constant precipitation particle intercept and density parameters, each ensemble member uses different values for these parameters. Thus, the ensemble members in the Perfect_MP experiment differ from each other not only in the location and magnitude of the thermal bubbles but also differ in intercept and density parameters within the same LFO microphysics scheme. The parameters varied include the intercept parameters for rain (n_{0r}), snow (n_{0s}) and hail/graupel (n_{0h}) and the bulk densities of snow (ρ_s) and hail/graupel (ρ_h). These values are varied within their typical uncertainty range based on past observational studies reported in the literature. The intercept parameter n_{0r} is varied between the range 3.98×10^6 and 3.16×10^7, n_{0h} between 4.50×10^3 and 4.00×10^5 and n_{0s} between 1.0×10^6 and 1.58×10^7 m^{-4}. The density ρ_s is varied between 20 and 400 and ρ_h between 400 and 900 kg m^{-3}. The lists of parameter values assigned to the 40 ensemble members in the multi parameter (MP) experiment are shown in Table 4.2. The mean of the intercept and density values from the 40 ensemble members differ from the values assigned in the truth run as listed at the bottom of Table 4.2. The use of a variety of density and intercept parameters across the ensemble members result in supercell storms that are different from each other in terms of structure, strength and intensity.

Table 4.2 List of ensemble members with the values of intercept parameters and densities of rain, hail/graupel and snow particles from the LFO microphysics scheme.

Ensemble Members	Hail/graupel intercept n_{0h} (m^{-4})	Density of hail/graupel ρ_h (kg m^{-3})	Snow intercept n_{0s} (m^{-4})	Density of snow ρ_s (kg m^{-3})	Rain intercept n_{0r} (m^{-4})
1	4.00×10^4	900	3.00×10^6	100	8.00×10^6
2	4.50×10^3	900	1.04×10^7	50	7.14×10^6
3	5.07×10^3	800	6.77×10^6	100	5.19×10^6
4	5.70×10^3	500	2.18×10^6	350	1.22×10^7
5	6.41×10^3	700	1.89×10^6	400	6.09×10^6
6	7.22×10^3	600	8.38×10^6	250	9.32×10^6
7	8.12×10^3	800	7.27×10^6	150	2.70×10^7
8	9.14×10^3	900	3.84×10^6	50	2.30×10^7
9	1.03×10^4	400	1.76×10^6	200	4.43×10^6
10	1.16×10^4	500	1.43×10^6	300	1.09×10^7
11	1.30×10^4	600	1.07×10^6	400	8.38×10^6
12	1.47×10^4	700	2.89×10^6	250	7.53×10^6
13	1.65×10^4	800	5.10×10^6	150	5.77×10^6
14	1.86×10^4	900	8.99×10^6	300	3.16×10^7
15	2.09×10^4	400	1.38×10^7	100	8.83×10^6
16	2.35×10^4	500	2.51×10^6	300	4.20×10^6
17	2.65×10^4	600	2.34×10^6	100	3.00×10^7
18	2.98×10^4	700	1.53×10^6	150	1.96×10^7
19	3.35×10^4	800	7.80×10^6	200	1.76×10^7
20	3.77×10^4	900	1.33×10^6	100	1.58×10^7
21	4.24×10^4	400	4.12×10^6	350	2.18×10^7
22	4.78×10^4	500	4.43×10^6	100	1.50×10^7
23	5.37×10^4	600	5.48×10^6	250	2.56×10^7
24	6.05×10^4	700	3.58×10^6	400	1.35×10^7
25	6.80×10^4	800	1.00×10^6	20	6.42×10^6
26	7.66×10^4	400	1.28×10^7	300	3.98×10^6
27	8.62×10^4	500	5.88×10^6	200	2.42×10^7
28	9.70×10^4	900	1.15×10^6	50	1.28×10^7
29	1.09×10^5	400	1.24×10^6	350	4.67×10^6
30	1.23×10^5	700	2.03×10^6	50	2.07×10^7
31	1.38×10^5	800	9.65×10^6	350	1.04×10^7
32	1.56×10^5	900	1.19×10^7	200	5.48×10^6
33	1.75×10^5	500	1.64×10^6	250	9.82×10^6
34	1.97×10^5	600	6.31×10^6	400	1.67×10^7
35	2.22×10^5	700	1.11×10^7	100	1.15×10^7
36	2.49×10^5	800	4.75×10^6	300	2.84×10^7
37	2.81×10^5	900	2.70×10^6	150	1.86×10^7
38	3.16×10^5	400	1.48×10^7	50	7.94×10^6
39	3.55×10^5	700	1.58×10^7	400	4.92×10^6
40	4.00×10^5	900	3.33×10^6	300	6.77×10^6
Average	9.00×10^4	675	5.45×10^6	214	1.33×10^7
LFO_Truth	4.00×10^4	900	3.00×10^6	100	8.00×10^6
10ICE_Truth	1.00×10^5	700	8.00×10^6	100	8.00×10^6

4.2.2 Imperfect Model Experiment

Unlike the previous experiment, the synthetic reflectivity and radial velocity observations assimilated by the ensemble members are generated from the Truth_10ICE run. The ensemble members in the first ensemble (Imperfect_Control) use the LFO microphysics scheme with the same constant precipitation particle parameters as in the Perfect_Control experiment. The ensemble members in the second MP ensemble (Imperfect_MP) also use the LFO microphysics scheme but with the same variety in the intercept and density parameters as in Perfect_MP (Table 4.2). The initialization and other ensemble configuration details are identical to the previous experiment. The imperfect model experiment explores the performance of the EnKF system for the same storm event in the presence of model errors in the different microphysics scheme.

The ultimate goal of storm-scale data assimilation is to obtain accurate short-term forecasts of severe storms events. To evaluate the accuracy of the ensemble forecasts from assimilating WSR-88D observations over a 30-min period, the 40 analyses from the last assimilation cycles are used as the initial conditions for each of the ensemble members and 1-h short-term forecasts are produced.

4.3 Results

The accuracy of the analyses and forecasts for both perfect and imperfect model assimilation experiments, when using fixed or varied microphysics scheme parameters in the ensemble system, are compared with the truth runs. The evaluation criteria include statistical comparisons between the truth and the ensemble system. Statistical measures

include root-mean-square error of the unobserved variables and equitable threat scores (ETSs: Wilks 2006). The ETS is calculated from the contingency table that gives discrete joint sample distribution of ensemble mean forecasts and the reference simulation in terms of cell count. An ETS of 1 denotes a perfect forecast while the forecast accuracy decreases as the ETS decreases towards zero. Various plots of ensemble maximum values are used to determine whether or not the ensembles capture the range of values found in the truth runs.

4.3.1 Analyses

To evaluate how well the supercell is captured by the ensemble system during the 30- min assimilation period, the rms errors of *u, v* and *w* wind components, temperature, and total precipitation (rain, snow and hail/graupel) mixing ratios from the ensemble mean analyses for both perfect and imperfect model assimilation experiments are examined (Figure 4.3). The rms errors from both experiments are seen to decrease rapidly for all variables as more observations are assimilated. At the end of the assimilation period, the rms errors for winds and temperature variables for the control and multi-parameter ensembles from both Perfect (Figures 4.3a and c) and Imperfect (Figures 4.3b and d) model experiments are very similar. However, while the rms errors of total precipitation mixing ratio from the Perfect_MP are larger than that of the Perfect_Control (Figure 4.3e), the rms errors of the Imperfect_MP are significantly smaller than that of the Imperfect_Control (Figure 4.3f) throughout the 30 minute assimilation period. Thus, in the presence of model error, the Imperfect_MP is able to capture the true precipitation mixing ratios better and hence produce smaller rms errors.

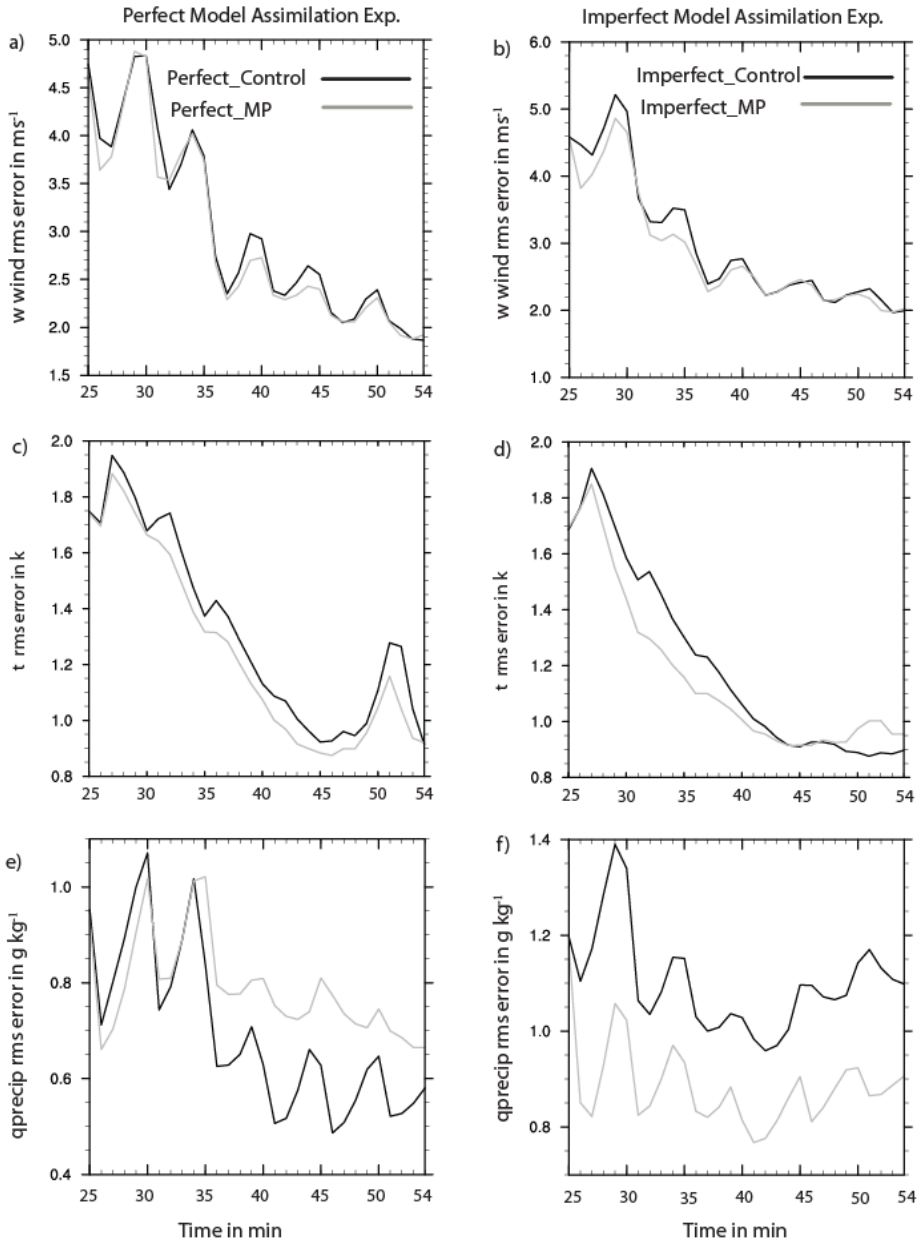

Figure 4.3 The rms errors of ensemble mean analyses vs. time(sec) during the 30-min assimilation period from the perfect and imperfect model experiment starting at t = 25 min and ending at t = 54 min for w (m s^{-1}) (a and b), t (k) (c and d) and total precipitation (rain, snow, hail/graupel) mixing ratios (g kg^{-1}) (e and f) for the control (black lines) and muliparameter (gray lines) ensemble system. Values are averaged over the domain at grid points where the total precipitation mixing ratios (sum of *qr*, *qh* and *qs*) in the truth run is greater than 0.10g kg^{-1}.

4.3.2 Forecasts

The rms errors of the ensemble mean forecasts during the 1 hour forecast period for perfect and imperfect model assimilation experiments are shown in Figure 4.4. The quality of the forecast in both plots deteriorates rapidly with time as expected. However, in the perfect model experiment, the Perfect_Control yields smaller rms errors compared to the Perfect_MP for the winds, temperature and total precipitation (Figures 4.4a, c and e). The smaller rms errors from the Perfect_Control are expected since the ensemble uses identical intercept and density parameters of the hydrometeor categories for all ensemble members as in Truth_LFO. In the absence of model error from the perfect model assumption, the EnKF only has to correct the initial condition errors. In the imperfect model experiment, the rms errors for winds and temperature variables (Figures 4.4 b and d) from the Imperfect_MP are very similar to the rms errors from the Imperfect_Control during the first 40-mins of the forecast period but yield smaller rms errors during the remaining 20 min of the forecasts. Moreover the Imperfect_MP generates smaller rms error than that of the Imperfect_Control for total precipitation mixing ratio (Figure 4.4f) throughout the 1-h forecast period. Therefore the variations in the microphysical parameters have a larger impact on the microphysical fields than on wind and temperature fields.

To quantify the forecast accuracy from the ensemble mean forecasts, the ETS is calculated by comparing the ensemble mean forecast with the truth for reflectivity values exceeding a 35 dBZ threshold and for precipitation (rain, snow and hail/graupel) mixing ratios exceeding a 1.0 g kg^{-1} threshold. Results indicate that for the perfect model assimilation experiment, the ETS for the Perfect_Control is larger than that of the

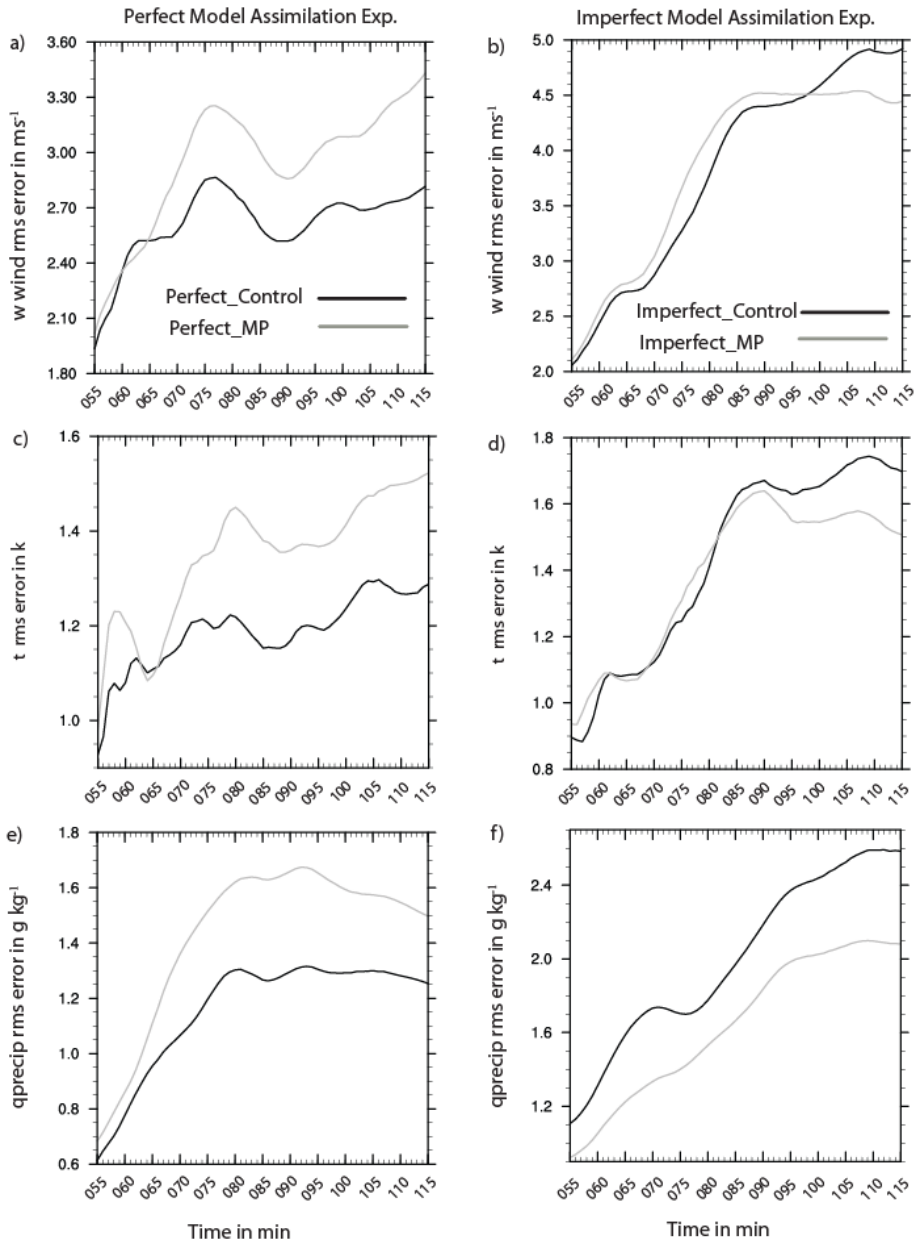

Figure 4.4 The rms errors of ensemble mean forecast vs. time(sec) during the 1-h forecast period from the perfect and imperfect model experiment starting at t = 55 min and ending at t = 115 min for w (m s^{-1}) (a and b), t (k) (c and d) and total precipitation (rain, snow, hail/graupel) mixing ratios (g kg^{-1}) (e and f) for the control (black lines) and muliparameter (gray lines) ensemble system. Values are averaged over the domain at grid points where the total precipitation mixing ratios (sum of *qr*, *qh* and *qs*) in the truth run is greater than 0.10g kg^{-1}.

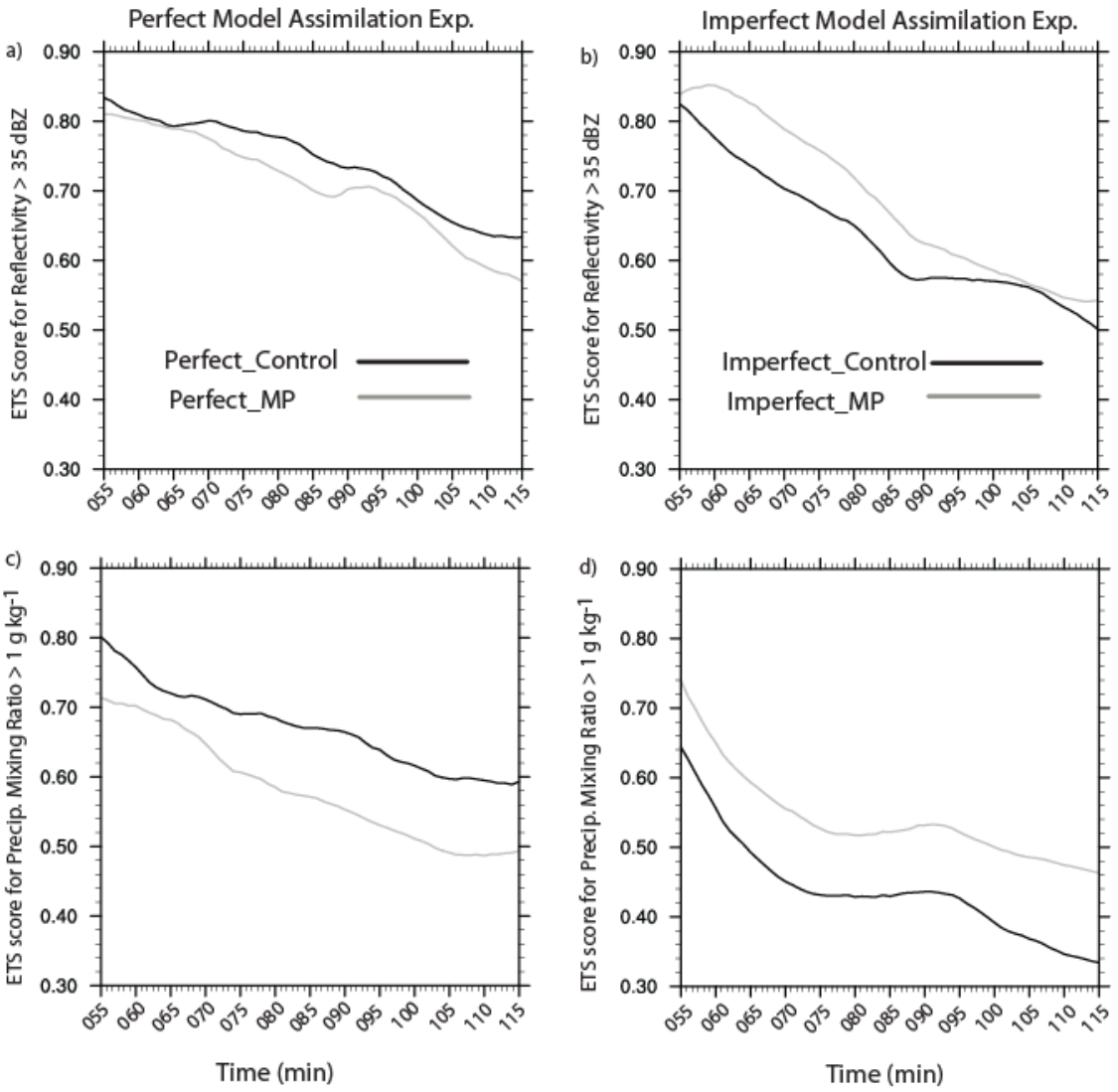

Figure 4.5 Values of equitable threat score (ETS) for reflectivity values exceeding 35 dBZ threshold for a) Perfect and c) Imperfect Model experiments and the precipitation (rain, snow and hail/graupel) mixing ratios exceeding 1.0 g kg^{-1} threshold for c) Perfect and d) Imperfect Model experiments as function of forecast time (sec). Details are shown in legends

Perfect_MP (Figures 4.5a and b) for the entire forecast period for both reflectivity and total precipitation mixing ratios. In contrast, for the imperfect model assimilation experiment, the Imperfect_MP yields a higher ETS throughout the 1-h forecast period compared to that of the Imperfect_Control (Figures 4.5 c and d) for both threshold values.

The maximum mean hail diameter (mm) at the lowest model level from anywhere in the model domain during the 1-h forecast period for the perfect and imperfect model assimilation experiments indicate that the truth value is often on the edge of the Perfect_Control ensemble (Figure 4.6a). Results from the Imperfect_Control show that the truth lies outside the ensemble envelope after 65 min and the ensemble members tend to overpredict the hail diameter (Figure 4.6b). In contrast, the MP ensembles (Figures 4.6c, d) capture the truth well within the ensemble members and also yield larger spread even though no additional methods are used to maintain the spread during observation assimilation period for both perfect and imperfect model assimilation experiments. The MP results also show how variations in hydrometeor parameters can dramatically change the prediction of hail size.

The large differences between the control and MP ensembles for maximum hail size would seem to indicate differences in storm structure. Yet all the MP storms are splitting supercells and have reflectivity values within 10 dBZ of the truth runs. Instead, these results highlight the variety of hydrometeor combination that can produce a given value of reflectivity, and, therefore, the sensitivity of the forecasts to the assumed microphysical parameters. It may be that these parameters can be estimated during the

data assimilation process, as done by Tong and Xue (2008a, b), but there is no guarantee that these estimated parameters will produce an accurate storm forecast.

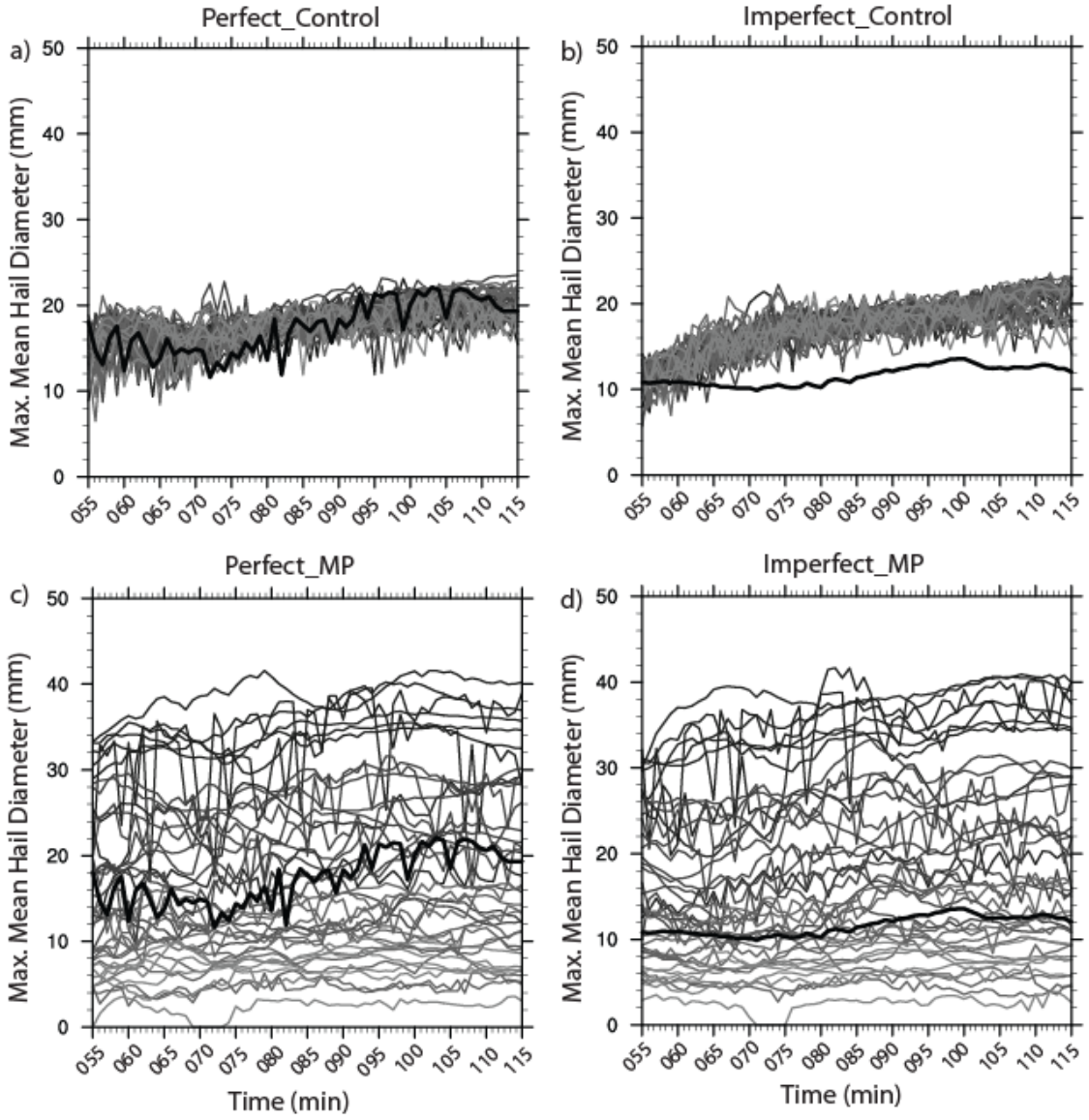

Figure 4.6 The maximum mean hail diameter (mm) at the lowest model level (100m AGL) during the 1-h forecast period from the truth (thick black line) and the 40 ensemble members (different shades of gray lines) for a) Perfect_Control, b) Imperfect_Control c) Perfect_MP, and d) Imperfect_MP assimilation experiment.

The ability of the EnKF to forecast the important variables in the convective storm environment is illustrated by comparing the forecast time series of the minimum cold pool temperature (Figure 4.6) from each ensemble member for both perfect and imperfect model assimilation experiments. Not surprisingly, the ensemble members from the control runs for both perfect and the imperfect model experiment provide insufficient ensemble spread, with the truth falling outside the ensemble envelope for different forecast periods, indicating that methods to artificially increase the spread are needed. In contrast, the MP experiments not only improve the ensemble spread, but also capture the truth well within the envelope of the ensemble members. The spread obtained from the MP ensemble not only represents the uncertainty from the initial conditions, but also the uncertainty from the various microphysical processes.

The ground relative total rainfall (mm) accumulated from the moving supercell storm valid at the end of 1-h forecast period is shown in Figure 4.7. The accumulated rainfall amounts from the Imperfect_MP ensemble mean forecast (Fig 4.7c) more closely resemble the truth (Figure 4.7a) than the rainfall amounts from the Imperfect_Control (Figure 4.7b) experiment. The Imperfect_Control produces higher rainfall amounts from the northern and the southern storms cells when compared to the truth 10ICE run. These results highlight the importance of using an MP ensemble in the presence of model error. Using a combination of different density and intercept parameters of the hydrometeor category can significantly improve the analyses and forecasts over experiments using constant intercept and density parameters for the hydrometeor categories. This is especially true when examining the extreme values of the model fields that would be most helpful in determining and identifying potential hazards.

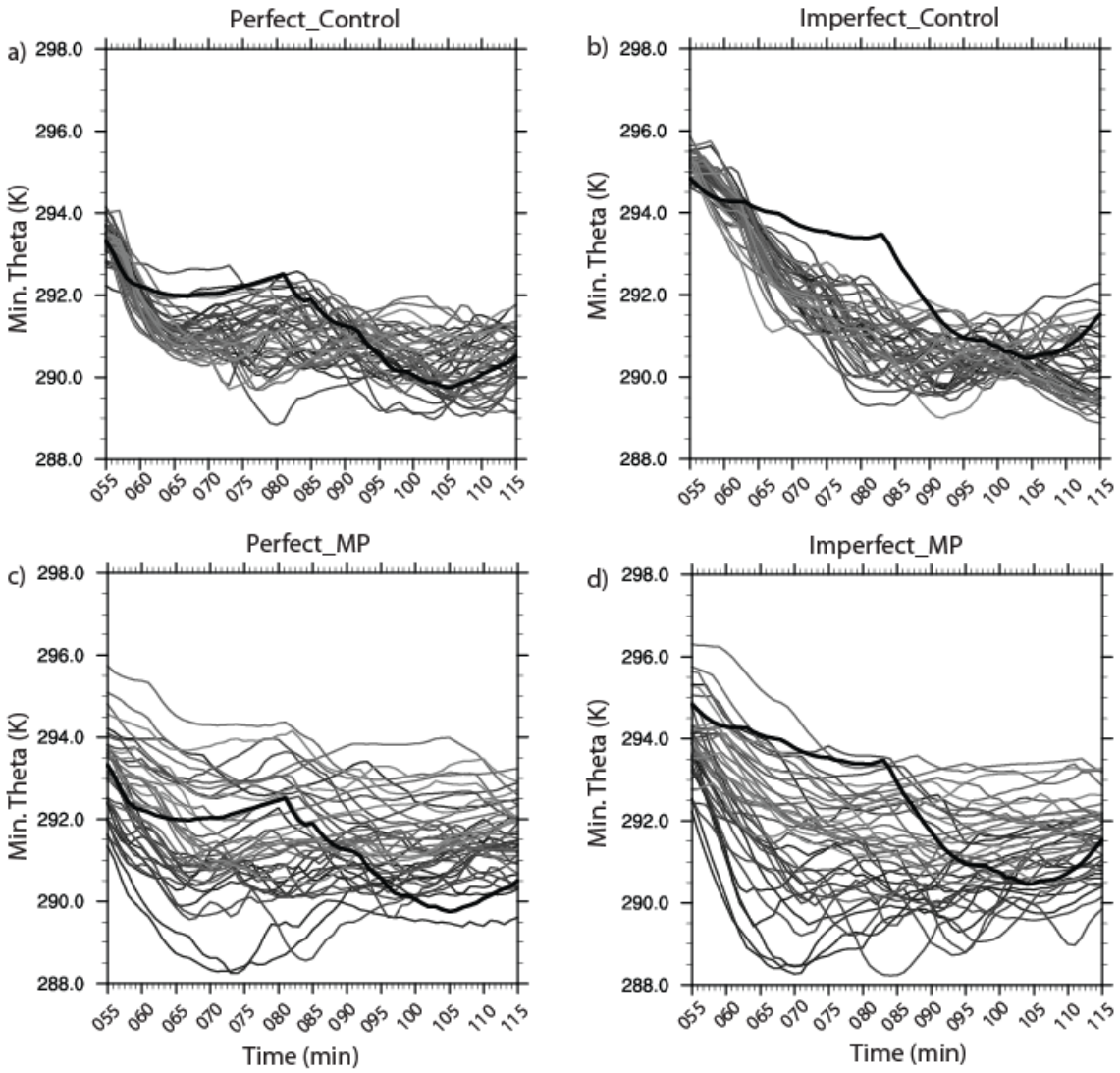

Figure 4.7 Same as in Figure 4.6 but for minimum potential temperature (K) at the lowest model level (100m AGL).

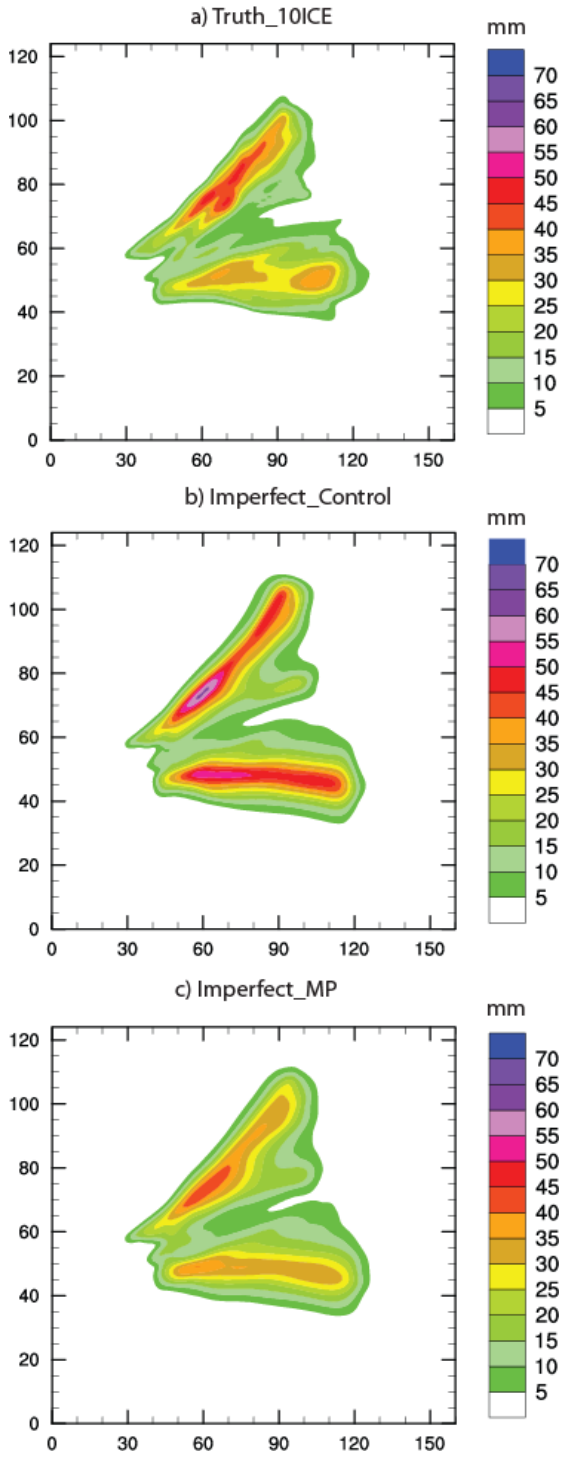

Figure 4.8 The ground-relative 1-h accumulated rainfall (mm) amounts of the supercell storm from a) Truth_10ICE and the ensemble mean forecasts of 1-h accumulated rainfall (mm) from b) Imperfect_Control and c) Imperfect_MP assimilation experiment.

4.4 Summary

The goal of this study is to evaluate the potential value of using a range of intercept and density parameters within the same microphysics scheme in the presence of model error. Two truth simulations of a splitting supercell storm are generated using LFO and 10ICE microphysics schemes in an identical storm environment. Two sets of OSSEs are conducted from both a perfect and an imperfect model framework using an EnKF data assimilation technique with 1) constant intercept and density parameters for the hydrometeors in all ensemble members and 2) a range of different values of the intercept and density parameters for the hydrometeors in the different ensemble members. Synthetic WSR-88D reflectivity and radial velocity observations are created from the truth runs using a realistic volume averaging technique and these observations are assimilated into the ensemble system over a 30-min period. The 40 ensemble analyses at last assimilation cycle are then used to make 1 h forecasts.

Results show that the EnKF system performs reasonably well with the imperfect model assumption. It is found that a multi-parameter ensemble within the imperfect model framework (Imperfect_MP) generates more accurate forecasts of ensemble mean precipitation mixing ratios and accumulated rainfall compared to that of the control imperfect model ensemble (Imperfect_Control). This conclusion does not always apply for the perfect model assumption where model error does not play a role. Moreover the 1-h forecast time series of the 40 ensemble members for lowest cold pool temperature at 100 m AGL indicates that the truth almost always lies within the envelope of ensemble members for the perfect and imperfect MP ensembles, whereas the truth more often lies on the edge or outside the ensemble envelope for the perfect and imperfect control

ensemble. The MP ensembles also yield larger ensemble spread than the control experiments. The results from this study support the idea that the microphysical parameter diversity across the ensemble members may be beneficial to a storm-scale ensemble forecasting system.

Caution is warranted as the results obtained in these studies are based on synthetic radar observations. In real observation assimilation, the model error can potentially be larger than that considered in this study. Moreover the selection of density and intercept parameters as shown in Table 4.2 is far from optimal. Thus the possibility of using multi-parameter ensemble in storm-scale data assimilation system should be tested on a broader range of experiments using real radar observations of severe weather events with careful selection of these highly uncertain microphysical parameters so that these values are representative of the various storm systems. Due to our limited understanding, it is likely that even the use of more sophisticated microphysics parameterization schemes will face challenges in some storm environments. This is not necessarily a deficiency but instead represents the reality of microphysics parameterization. Using a variety of realistic intercept and density parameters, the ensemble is more likely to span the observations and provide improved short range forecasts for a wide range of storm systems.

Chapter 5

Data Assimilation Using Extended Information Filter

5.1 Introduction

The previous two chapters apply the EnSRF data assimilation technique to assimilate high spatial resolution radar observations to NWP model and the results obtained are promising. However, the computational time for EnSRF methods scales linearly with the number of observation. Thus from the computational point of view, the algorithm for EnSRF method is efficient when the number of observations to assimilate is smaller. When the number of observations is very large as in the case with radar data, the EnSRF method becomes exceedingly time consuming. With the advent of new radar and other remote sensing technology, it is highly likely that the observation dimensionality exceeds the dimensionality of the model state vector. Therefore, efficient filter designs for efficient assimilation of these observations needs to be explored. One possible candidate for this purpose is the information form of the filter which is algebraically equivalent to Kalman filter. While the traditional Kalman filter calls for inverting the matrix in observation space, the information filter calls for the inversions of the model space. Therefore, the information form of the filter may be computationally more efficient than the traditional Kalman filter when the number of observations is very large in dimension compared to the model state ($m > n$). Even though the information filter has been around for years, to our knowledge the applicability of the information filter as data assimilation technique for high frequency measurements has not yet been tested for atmospheric models. The information filter is not widely used and is not widely covered

in the literature. Thus, this chapter explores the possibility of information filter as an efficient data assimilation technique for large observation system. As a first step, the information filter is implemented using a low dimensional simple atmospheric model, and its performance is compared with the Kalman filter as a benchmark. The model used for this purpose is the Lorenz 96 model (L96) which is a simple non-linear model (Lorenz 1996; Lorenz and Emanuel 1998; Lorenz 2005, 2006), computationally cheap and shares many characteristics with the realistic atmospheric models. This model is widely used as a test bed for examining the data assimilation schemes in meteorological community (Anderson 2001; Whitaker and Hamill 2002; Ott et al. 2004; Fertig et al. 2007; Nokano et al. 2007; Leutbecher et al. 2007; Ambadan and Tong 2009). Therefore, we apply the information filter to a simple Lorenz model as this is useful for initial testing of new ideas, before complex high-dimensional models and real observations are used. Since the Lorenz model is nonlinear, the extended form of the information filter (EIF) is implemented and is compared with the extended Kalman filter (EKF). The EIF experiments conducted in this chapter uses the state space formulation (Simon 2006).

5.2 Description of Lorenz Model

The L96 model is a one-dimensional atmospheric model introduced by E. Lorenz in 1995 to explain the dynamics of weather at fixed latitude. The model consists of 40 ordinary differential equations, with the dependent variables representing values of some atmospheric quantity at 40 sites spaced equally about a latitude circle. The equations contain quadratic, linear, and constant terms representing advection, dissipation, and

external forcing. The model contains N variables $x_1, \ldots, x_N$, which may be thought of as atmospheric variables in N sectors of a latitude circle and is governed by

$$\frac{dx_n}{dt} = -x_{n-2}x_{n-1} + x_{n-1}x_{n+1} - x_n + F \tag{5.1}$$

for $n = 1, \ldots, N$. To make (5.1) meaningful for all values of n, $x_{-1} = x_{N-1}$, $x_0 = x_N$, and $x_{N+1} = x_1$ are defined so that the variables form a cyclic chain, and the values can be assumed as some unspecified scalar meteorological quantity, like the temperature, at N equally spaced sites extending around a latitude circle (Figure 5.1). The model does not simulate the atmosphere's latitudinal or vertical extent. The constant F is positive and is known as the forcing term, t is the time.

The model is formulated as one of the simplest possible systems that treat all variables alike. However there are certain properties in the model that are similar to many atmospheric models and are as follows (Lorenz and Emanuel 1998):

1. The two nonlinear terms are intended to simulate advection. These two terms are quadratic and together conserve the total energy, defined as $(x^2_1 + \cdots + x^2_N)/2$.
2. The linear terms represents mechanical or thermal dissipation and decreases the total energy.
3. The constant term represents external forcing and prevents the total energy from decaying to zero.

The variables are scaled so that the coefficients of the quadratic and linear terms are unity. The time unit is thus the dissipative decay time, which is assumed to equal 5 days. Numerical integration of this model indicates that small errors (differences between

solutions) tend to double in about 2 days. Further details of the model and its behaviors can be found in Lorenz (2005, 2006) and Lorenz and Emmanuel (1998).

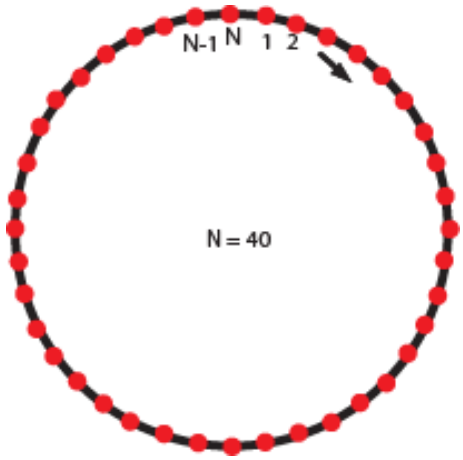

Figure 5.1 Latitude circle of the Lorenz 96 model with 40 grid points (N = 40).

5.3 Experimental Details

The Lorenz model is computationally stable with a time step of 0.05 units which equals 6 h (Lorenz and Emanuel 1998). A unit time $\Delta t = 1$ is associated with 5 days. Thus for the Lorenz 96 model, a "year" consists of twelve 30-day months, or 72 time units, or 2880 time steps. Similar to Lorenz and Emanuel (1998), $N = 40$, F = 8 and a fourth-order Runge–Kutta time integration scheme with a time step of 0.05 non-dimensional units or 6 h is used to run the model to create initial conditions. The model and the experiments conducted in this chapter is implemented using MATLAB software and the codes are listed in appendix B. Extensive testing of the simulation of the model is conducted to validate the code and to make sure that the performance of the model is similar to that found in Lorenz (1996), Lorenz and Emanuel (1998), and Lorenz (2005, 2006).

Creating the initial condition for the truth and the model state:

The initial state of the model ($\hat{x}_0, \hat{P}_0$) and the 'truth' are obtained by integrating the L96 model for a long period of time starting from an arbitrary start-up value. For the truth run, random numbers from a uniform distribution between 0 and 1 are assigned to each of the 40 variables, and an initial perturbation of 0.008 is added to the 20th variable (x_{20}). The model is then integrated forward in time for 10 years or 14400 steps similar to Lorenz (1996). The final values which are more or less free of transient effects are taken as the true initial values for the assimilation experiments. Observations (z_k) are created from the truth run by adding Gaussian noise with zero mean and specified standard deviation. The climatological mean ($\mu_{c\lim}$) and the standard deviation ($\sigma_{c\lim}$) for the 10 years truth run are 2.3432 and 3.6385 respectively.

For creating the model initial condition ($\hat{x}_0, \hat{P}_0$), an ensemble of 100 model members is used. The first ensemble member starts with the identical setting as in the truth run and the remaining 99 ensemble members are generated from the first ensemble member by adding additional random perturbation (with 0 mean and 0.0001 standard deviation) in one additional randomly selected variable (or grid points). A plot of the 100 ensemble members for grid point 30 at the start time is shown in Figure 5.2 that indicates

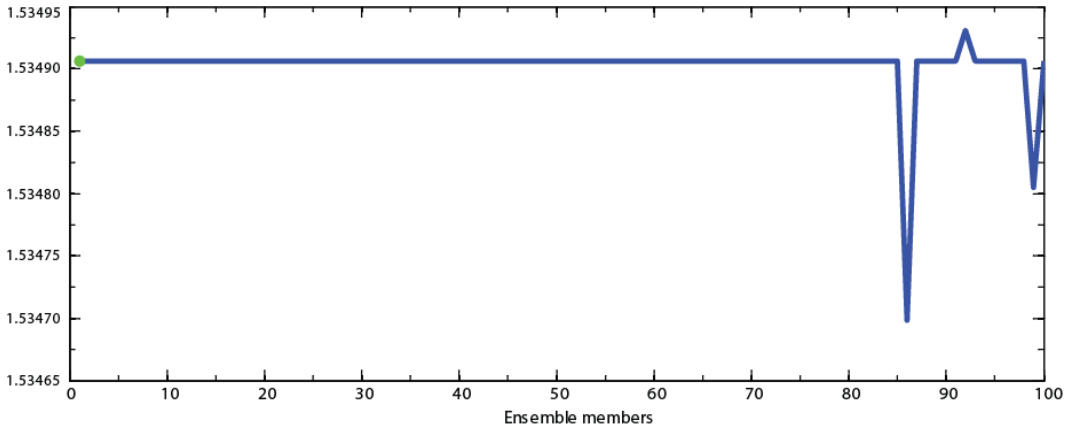

Figure 5.2 Values of ensemble members (blue) and the truth (green) at grid point 30 (or variable 30) at start time.

that additional perturbations are added to ensemble members 86, 93 and 99. The remaining ensemble members and the truth have identical values for grid point 30.

The 100 ensemble members are then integrated forward in time for 10 years similar to the truth run. The 100 ensemble members and the truth at the end of 10 year time integration is shown in Figure 5.3. The small initial perturbations to the different states of the ensemble at the start time evolve with time and the ensemble members at the end of the integration period are chaotic. Finally the ensemble members at the end of the 10 year simulated period are averaged to obtain the model initial condition $\hat{x}_0$. The initial ensemble covariance matrix $\hat{P}_0$ at the end of the l0 year simulation period is obtained from the ensemble members. The contour plot of P_0 is shown in Figure 5.4a. The covariance plots clearly show the strong variance of the grid points along the diagonal and small covariances among the grid points.

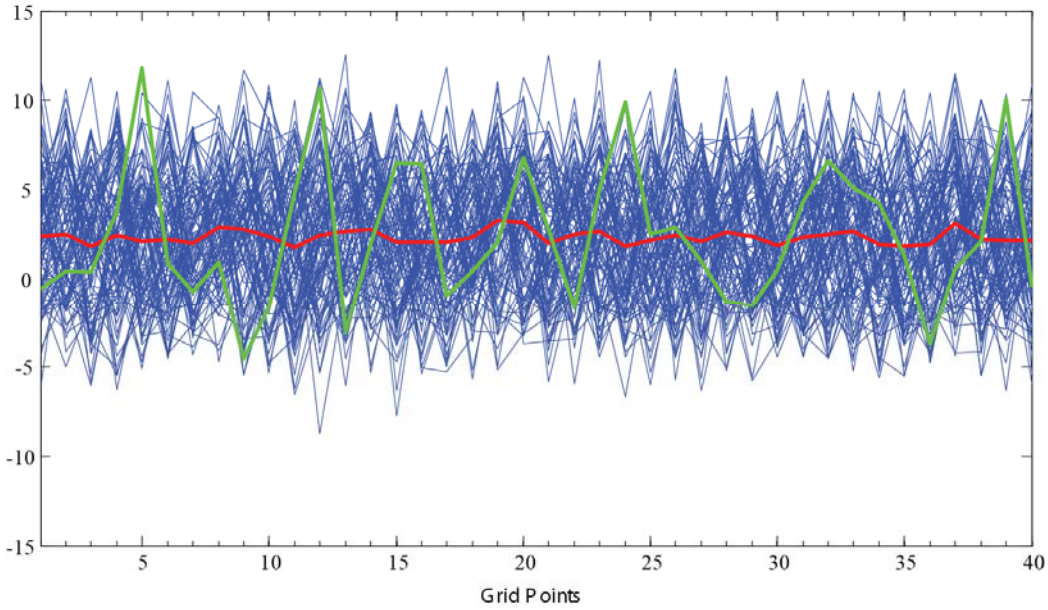

Figure 5.3 The 100 ensemble members (blue lines), truth run (green line) and the ensemble mean (red line) after integrating the model for 14400 time steps.

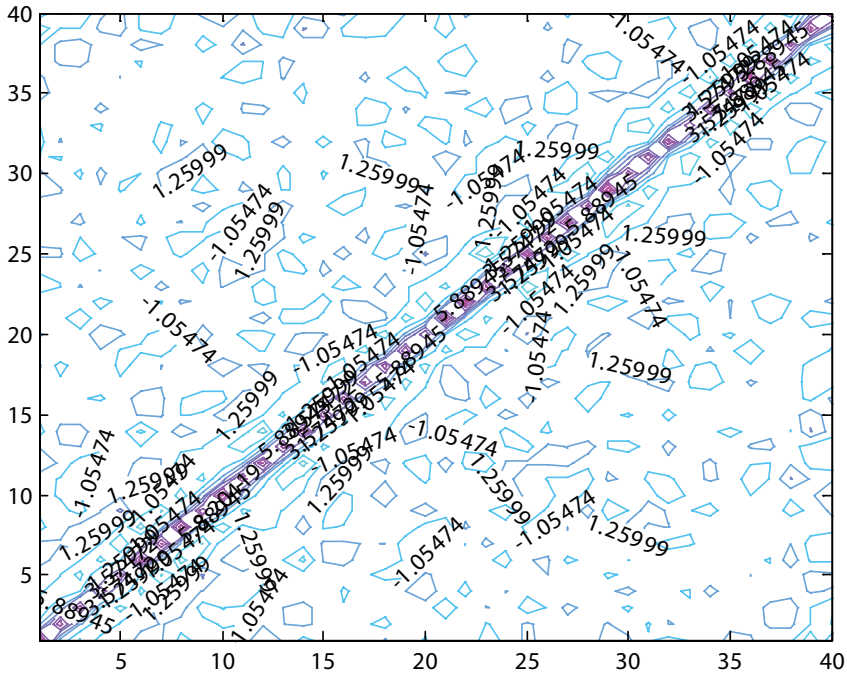

Figure 5.4 The covariance P_0 of the model (contours) after integrating the model for 14400 time steps.

Data Assimilation Experiments:

The assimilation experiment is initialized at time k = 0 and the initial value of the mean and covariance of the model is assigned to $\hat{x}_0$ and $\hat{P}_0$ respectively. It is assumed that the observations are available at every time step and are assimilated by the model every time step for the next 500 time steps. For the assimilation experiment, the time step is assumed to be 0.01 to secure stability. The model error standard deviation σ_Q is assumed as σ_Q = 0.3 and the model error covariance Q is represented by a diagonal matrix $Q = \sigma_Q^2 I$, where I is the identity matrix.

The observation error standard deviation is taken as $\sigma_R = 0.25\sigma_{c\lim}$ (i.e. the observation error standard deviation is 25% of the climatological standard deviation). The observational error covariance matrix is assumed as $R = \sigma_R^2 I$. We also assume that the observation and model error covariance matrices Q and R and the H operator are constant over time. The observations y^o are computed at each assimilation cycle from the truth run by adding uncorrelated Gaussian random noise 0 mean and σ_R standard deviation.

At each time step, observations are created from the truth run and are assimilated into the model to create analysis. After the assimilation, the analysis is integrated forward in time to the next time step and the cycle goes on. Three sets of experiments are conducted based on observation density on the 40-dimensional model. Each set of experiments is conducted using two combination of σ_Q and σ_R, namely $\sigma_Q = 0.3, \sigma_R = 0.25\sigma_{c\lim}$ and $\sigma_Q = 0.25\sigma_{c\lim}, \sigma_R = 0.25\sigma_{c\lim}$. Also to ensure fairness, the

assimilation experiments for EKF and EIF are carried out using identical settings. The experiments are as follows:

5.3.1 Sparse Observation (*m* = *n)* Network

It is assumed that the observations are available at every grid point (Figure 5.1a). Since each of the model grid points (state variable) are observed directly, the observational operator *H* is assumed to be identity.

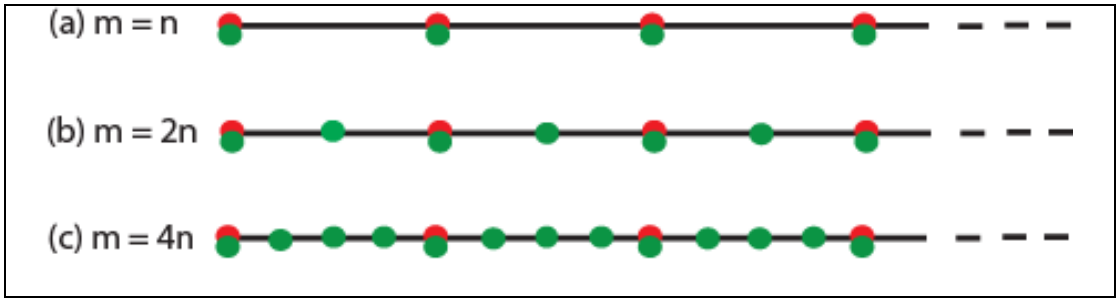

Figure 5.5 Location of observations (green circle) and model grid points (red circle) for (a) $m = n$, (b) $m = 2n$ and (c) $m = 4n$ experiments. Here m is the number of observations and n is the number of model grid points.

5.3.2 Moderately Densed Observation (*m* = 2*n)* Network

In addition to observations at every site, observations also are available in between every two sites (Figure 5.5b). Thus, there are 80 observations in total. The *H* operator includes a linear interpolation from the model grid to the location of the observation site in between two model grid point.

5.3.3 Highly Densed Observation (*m* = 4*n)* Network

In addition to observations at every site, three observations also are available in between every two sites (Figure 5.5c) and are equally spaced. Thus, there are 160 observations in total. The *H* operator includes a linear interpolation from the model grid to the location of each observation in between two model grid point.

5.4 Results

To guarantee that there is no inadequacy of the model to explain the observations and the filter is working as it should, the term $r_k = (z_k - H_k x_k^f)$ known as the innovation or the residual is calculated for both the EKF and EIF experiments. For all experiments 100 samples with each sample run consisting of a 500 assimilation cycle is conducted to calculate $E(r_k)$. The resultant $E(r_k)$ for both EKF and EIF are ~0.0. Thus both filters are working as they should.

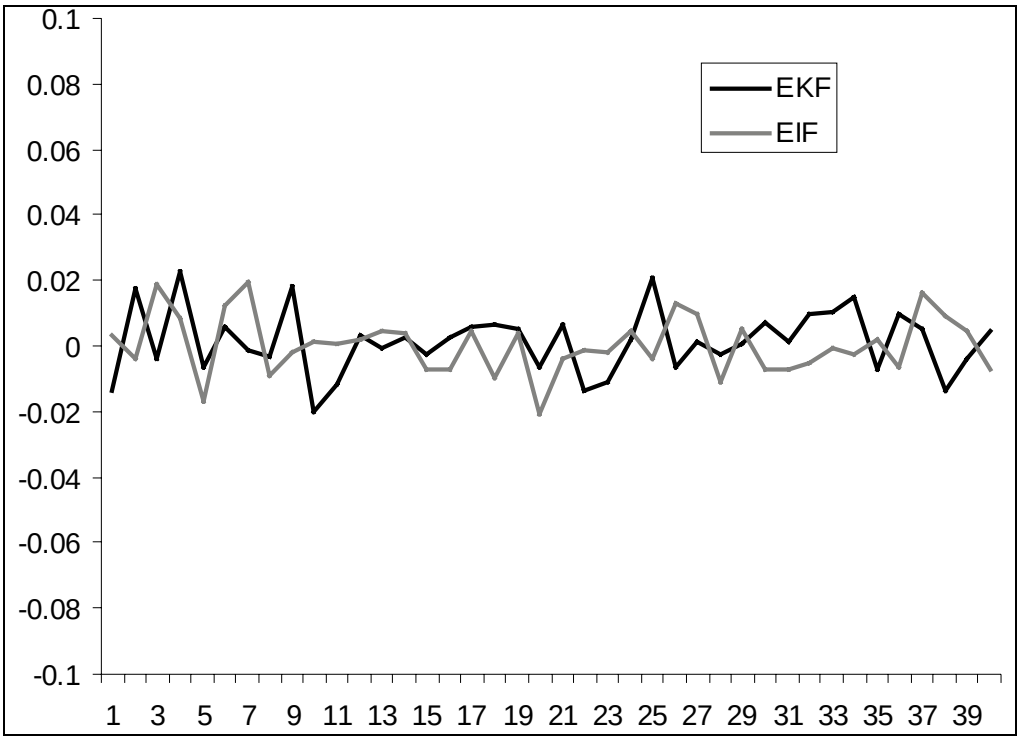

Figure 5.6 The expected value of the innovations $E(r_k)$ for both EKF (in black line) and EIF (in gray line) at 40 observation location from the *m*=*n* experiment.

Moreover from Figure 5.7, it is also seen that with observation assimilation cycle the model forecast more closely converges to the truth. The accuracy of the EKF and EIF are evaluated using the root-mean-square error (rms) of the analyses and forecasts and are calculated as the difference between the truth and the analyses and forecasts. The rms error is defined as

$$E = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - X_i^{true}\right)^2}$$

where $n = 40$ is the number of grid points, X_i is the ith variable for the forecast and analyses, and X^{true} is the "true" state from which the observations were sampled.

The EKF and EIF rms error for $\sigma_Q = 0.30, \sigma_R = 0.25\sigma_{c\lim}$ and $\sigma_Q = 0.25\sigma_{c\lim}, \sigma_R = 0.25\sigma_{c\lim}$ using the sparse observation (*m = n)* network, moderately densed observation (*m* = 2*n)* network and highly densed observation (*m* = 4*n)* network as shown in Figure 5.8, Figure 5.9 and Figure 5.10 respectively. The rms errors from both EKF and EIF are very similar to each other. The initial rms error from the model forecast reduces from ~3.75 to ~0.50 after the first assimilation cycle and both the analyses and the forecast rms errors varies with the range of ~0.50-1.0 for the rest of the assimilation period. These results support the theory that the information filter is algebraically equivalent to the Kalman filter.

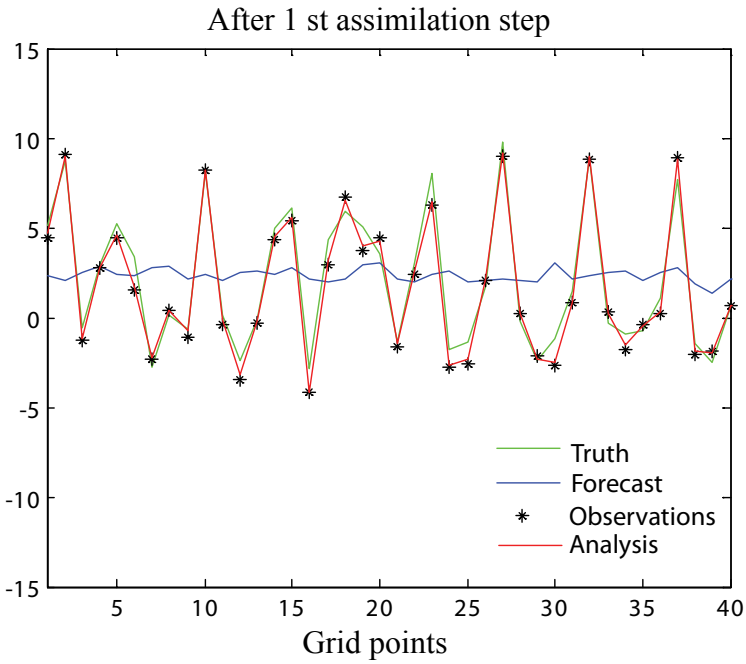

Figure 5.7 The truth run (in green), observation locations (black starts), model forecast (in blue) and the analysis (in red) after the first assimilation cycle.

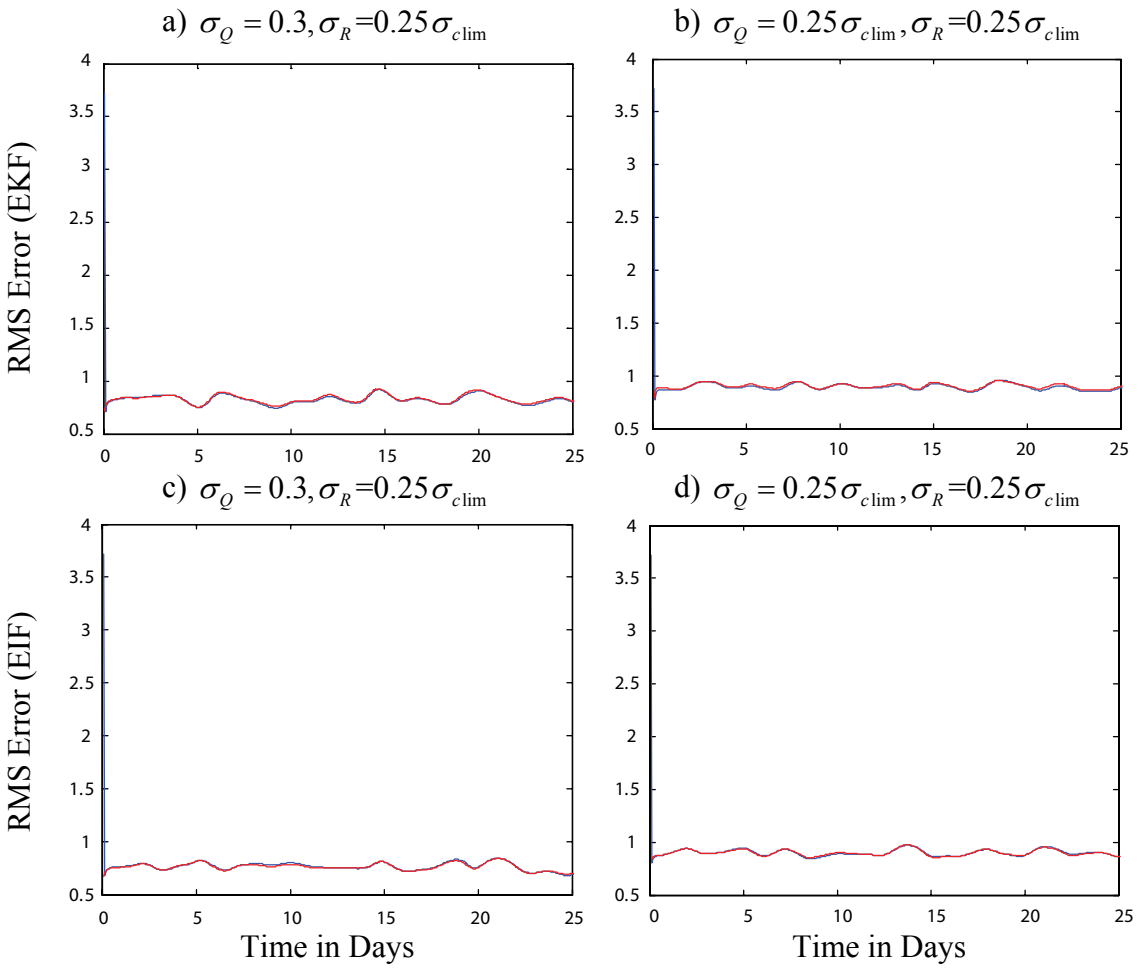

Figure 5.8 The rms error for the (a, b) EKF and (c,d) EIF forecast and analyses during data assimilation period for the sparse observation (*m = n)* network. The blue line indicates the model forecast error and the red line indicates the analyses error.

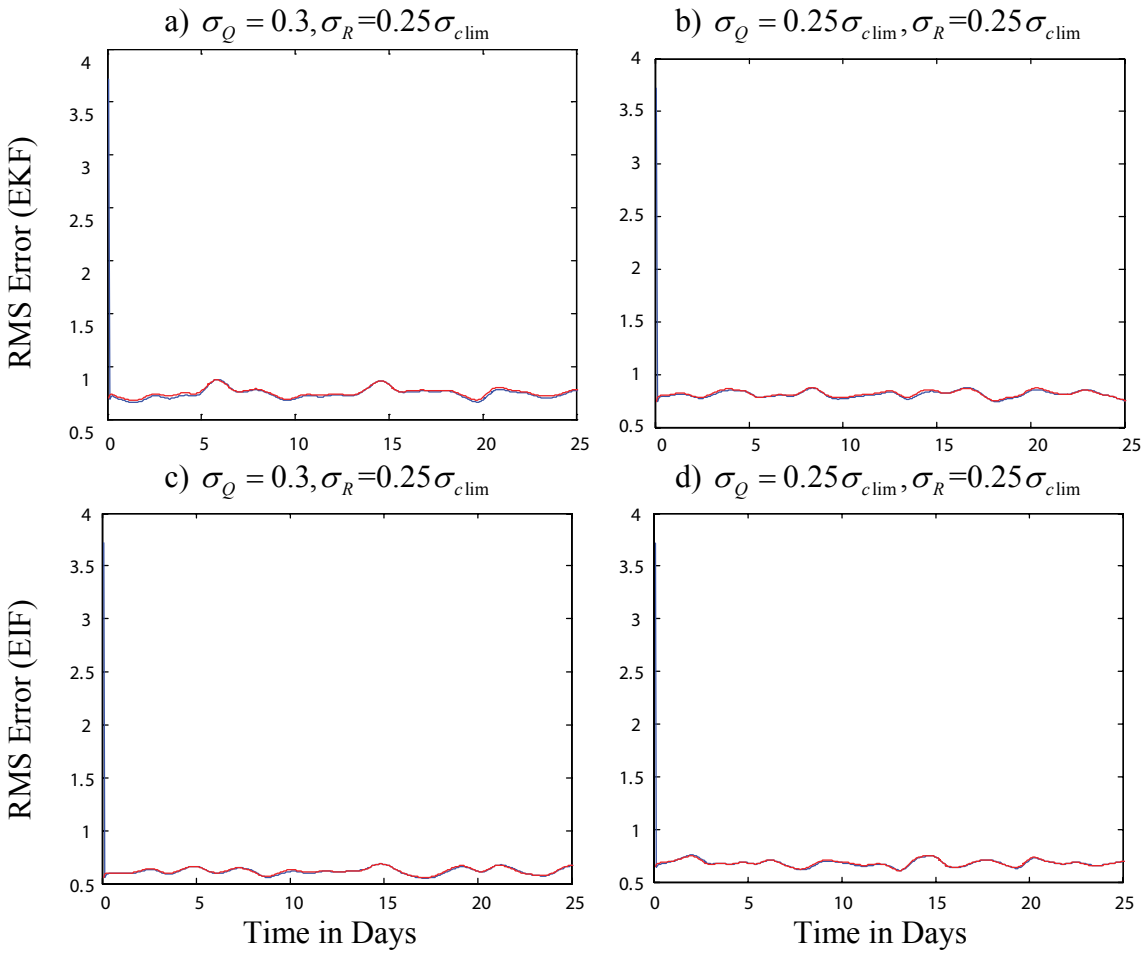

Figure 5.9 Same as in Figure 5.8 but for the moderately densed observation (*m* = 2*n)* network

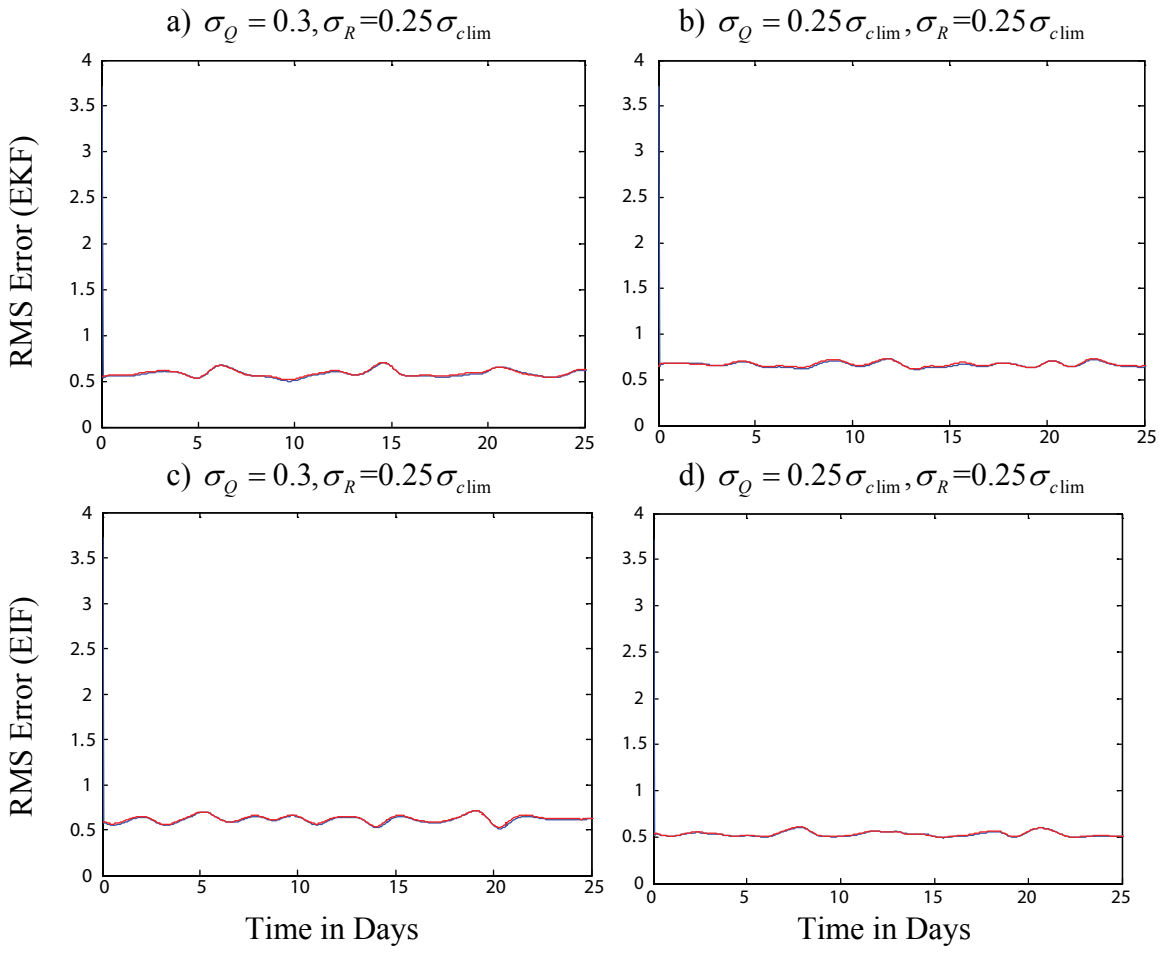

Figure 5.10 Same as in Figure 5.8 but for the highly densed observation ($m = 4n$) network

5.5 Computational Speed

Following Lewis et al. (2006), we quantify the amount of work needed to perform one complete iteration of forecast and analyses step for EKF and EIF. The complexity is calculated in terms of the number of floating point operations (flops). To multiply two matrices $A \in R^{nxm}$ and $B \in R^{mxr}$ it takes $2mnr$ flops (mnr multiplications and mnr additions). While it is true that in general multiplication takes more time than addition, to simplify the process of estimating the cost, it is useful to assume a unit cost model where

the unit of cost (measured in time) is equal to the maximum of the cost of performing a single operation of addition, subtraction multiplication and division. Using this convention, the total cost in terms of the number of flops as a function of the size of the problem is listed in Table 5.1 for EKF and in Table 5.2 for EIF.

Table 5.1: Estimation of the computational cost for EKF.

Item	Operation	Type of Computation	Cost
x_{k+1}^f	$M_k(\hat{x}_k)$	Matrix-vector multiply	$2n^2$
P_{k+1}^f	$D_M(\hat{x}_k)\hat{P}_k D_M^T(\hat{x}_k)+Q_{k+1}$	Two matrix-matrix multiply + a matrix add	$4n^3+n^2$
K_{k+1}	$H_{k+1}P_{k+1}^f H_{k+1}^T + R_{k+1}$	Two matrix-matrix multiply + a matrix add	$4n^2m+m^2$
	$[H_{k+1}P_{k+1}^f H_{k+1}^T + R_{k+1}]^{-1}$	Inverse of a matrix	$(1/3)m^3$
	$P_{k+1}^f H_{k+1}^T[H_{k+1}P_{k+1}^f H_{k+1}^T + R_{k+1}]^{-1}$	One matrix-matrix multiply	$2m^2n$
	Total Cost of K_{k+1}		$4n^2m+2m^2n+(1/3)m^3+m^2$
$\hat{P}_{k+1}$	$I - K_{k+1}H_{k+1}$	One matrix-matrix multiply and add identity matrix	$2n^2m+n$
	$[I - K_{k+1}H_{k+1}]P_{k+1}^f$	matrix-matrix multiply	$2n^3$
	Total Cost of $\hat{P}_{k+1}$		$2n^3+2n^2m+n^2$
$\hat{x}_{k+1}$	$z_{k+1} - H_{k+1}x_{k+1}^f$	Matrix-vector multiply and a vector add	$2nm+m$
	$K_{k+1}[z_{k+1} - H_{k+1}x_{k+1}^f]$	Matrix-vector multiply	$2nm$
	$x_{k+1}^f + K_{k+1}[z_{k+1} - H_{k+1}x_{k+1}^f]$	Vector add	n
	Total Cost of $\hat{x}_{k+1}$		$4nm+n+m$
Total Cost: $6n^3+4n^2+6n^2m+2m^2n+(1/3)m^3+m^2+4mn+n+m$			

Table 5.2: Estimation of the computational cost for EIF

Item	Operation	Type of Computation	Cost
x_{k+1}^f	$M_k(\hat{x}_k)$	Matrix-vector multiply	$2n^2$
Y_{k+1}^f	Q_{k+1}^{-1}	Inverse of a matrix	$(1/3)n^3$
	$\hat{Y}_k + D_M^T(\hat{x}_k) Q_{k+1}^{-1} D_M(\hat{x}_k)$	Two matrix-matrix multiply + a matrix add	$4n^3+n^2$
	$[\hat{Y}_k + D_M^T(\hat{x}_k) Q_{k+1}^{-1} D_M(\hat{x}_k)]^{-1}$	Inverse of a matrix	$(1/3)n^3$
	$Q_{k+1}^{-1} - Q_{k+1}^{-1} D_M(\hat{x}_k)[\hat{Y}_k + D_M^T(\hat{x}_k) Q_{k+1}^{-1} D_M(\hat{x}_k)]^{-1} D_M^T(\hat{x}_k) Q_{k+1}^{-1}$	Three matrix-matrix multiply + a matrix add	$6n^3+n^2$
	Total Cost of Y_{k+1}^f		$(32/3)n^3+2n^2$
$\hat{Y}_{k+1}$	R_{k+1}^{-1}	Inverse of a matrix	$(1/3)m^3$
	$Y_{k+1}^f + H_{k+1}^T R_{k+1}^{-1} H_{k+1}$	Two matrix-matrix multiply + a matrix add	$2m^2n+2n^2m+n^2$
	Total Cost of $\hat{Y}_{k+1}$		$(1/3)m^3+2m^2n+2n^2m+n^2$
K_{k+1}	$(\hat{Y}_{k+1})^{-1}$	Inverse of a matrix	$(1/3)n^3$
	$(\hat{Y}_{k+1})^{-1} H_{k+1}^T R_{k+1}^{-1}$	One matrix-matrix multiply	$2n^2m$
	Total Cost of K_{k+1}		$(1/3)n^3+2n^2m$
$\hat{x}_{k+1}$	$z_{k+1} - H_{k+1} x_{k+1}^f$	Matrix-vector multiply anda vector add	2nm+m
	$K_{k+1}[z_{k+1} - H_{k+1} x_{k+1}^f]$	Matrix-vector multiply	2nm
	$x_{k+1}^f + K_{k+1}[z_{k+1} - H_{k+1} x_{k+1}^f]$	Vector add	n
	Total Cost of $\hat{x}_{k+1}$		4nm+n+m
Total Cost: $11n^3+5n^2+4n^2m+2m^2n+(1/3)m^3+4mn+n+m$			

From the above two tables, it is obvious that the cost of EKF is smaller than EIF when *n* and *m* are equal.

The approximate computational time of EKF and EIF for the three sets of experiments is listed in Table 5.3. The computational times are based on a PC of 3.4 GHz Intel Pentium 4 with 2GB of RAM. The total computational time is larger for the EIF than for EKF for all experiments. This is expected since the state space formulation of EIF involves several matrix inversions and thus the computationally demanding matrix inverse contributes a significant component of the computation time for EIF compared to that of EKF. Moreover, as expected the computation time is longer as the number of observations are increased for both EKF and EIF. This is in agreement with the computational complexities of EKF and EIF as shown in Table 5.1 and Table 5.2 respectively. However, while the computational time for EKF grows by a factor of ~3.27 as the number of observations *m* increases from 80 to 160, the computation time for EIF grows by a factor of ~1.07. Therefore, the computation time for EIF is less effected with the addition of observations compared to the EKF.

Table 5.3: Approximate computational run time (on 3.2GHz Intel Xeon processor) for the three sets of experiments using both EKF and EIF.

σ_Q, σ_R	EKF			EIF		
	m=n	m=2n	m=4n	m=n	m=2n	m=4n
$\sigma_Q = 0.30, \sigma_R = 0.25\sigma_{c\lim}$	0.6030	1.1541	3.7827	7.362	7.5459	8.0517
$\sigma_Q = 0.25\sigma_{c\lim}, \sigma_R = 0.25\sigma_{c\lim}$	0.7301	1.2296	3.5680	7.3967	7.5470	7.9880

5.6 Summary

Both EKF and EIF are implemented in this chapter using a 40 dimensional L96 model. To generate the truth run, the model with random start-up values is integrated forward in time. In addition, to calculate the model initial condition $\hat{x}_0$ and the initial ensemble covariance matrix $\hat{P}_0$, a 100 member ensemble is created by taking the start-up values of the truth run and adding small perturbations to random grid points and is integrated forward in time. Both the truth run and the 100 member ensemble is stopped after 14400 time steps to model initial condition ($\hat{x}_0, \hat{P}_0$) are created.

Both the filters are initialized using the same initial conditions ($\hat{x}_0, \hat{P}_0$) and the observation assimilation experiments are conducted. The truth run and the model are run in parallel and at every time step observations are created from the truth run and assimilated into the model. The filter experiments are conducted for 500 time steps. The performance of both filters is then compared. Results clearly indicate that both EKF and EIF produce similar rms errors for the three different experiments conducted using three different observation resolutions. This essentially supports the theory that the information filter is algebraically equivalent to the Kalman filter. The computational time for the EIF is larger than that of the EKF filter as expected due to the large computational cost of matrix inversion of the EIF techniques. However, the increment in computational cost for EIF is much smaller than that of EKF for increased number of observation assimilation.

Chapter 6

Summary and Future Work

The Ensemble Kalman filtering technique introduced about a decade ago has become very popular within the meteorological community as an effective data assimilation technique. The EnSRF technique, a variant of Ensemble Kalman filtering technique shows promise in initializing storm-scale NWP models using radar observations for thunderstorm prediction. Studies suggests that the assimilation of WSR-88D radar observations in storm-scale NWP models using EnSRF data assimilation techniques can produce reasonable analyses and forecast of storms from assimilating observations for about an hour. However, severe weather events can evolve very rapidly and the weather forecasters may not have the flexibility to assimilate radar observations for an hour to make a forecast. With the advent of the new PAR technology, it is now possible to obtain a snapshot of the storms in less than a minute as compared to the operational WSR-88D radar that takes about 5 minutes to scan the same weather phenomena.

Thus to quantify the value of assimilating PAR observations for a shorter period of time, the EnSRF data assimilation technique is applied to assimilate radar observations into the storm-scale model in Chapter 3. A realistic radar emulator is developed and artificial WSR-88D and PAR reflectivity and radial velocity observations are generated from a simulated supercell storm. Both WSR-88D and PAR samples the weather at high spatial resolution. However, the computational time of the EnSRF algorithm scales linearly with the number of observations. Therefore, to reduce the heavy computational

burden of assimilating high spatial resolution radar observation, synthetic observations are generated at a coarser 1-km range resolution instead of the 0.25 km interval available from the radar. The experiments are conducted based on perfect model assumption where both the truth run and the ensemble use the same microphysics scheme, that is the model error do not play any role. One experiment assimilates 3 volumes of WSR-88D radar observations and another experiment assimilates 15 volumes of PAR observations during the short 15-min assimilation period. Finally, the analyses and short-term (less than 1 hour) forecasts from WSR-88D and PAR observations assimilation are compared. In general, the high-temporal frequency PAR observation assimilations using EnSRF technique is very promising. Results indicates that PAR observations assimilation provide more accurate analyses and forecasts of the storm compared to the WSR-88D assimilation. Thus assimilating high temporal frequency radar data for a shorter period of time may improve short-term forecasting and warnings of severe weather events with the possibility of increasing warning lead time.

The experiments conducted in Chapter 3 are based on perfect model assumption. However, in real world scenario, model errors play an important role in data assimilation and forecasts and needs to be incorporated in the experiments. In Chapter 4, the impact of model error in radar data assimilation is conducted based of imperfect model assumption. In addition the potential value of using a range of intercept and density parameters within the same microphysics scheme in the presence of model error also is explored in Chapter 4. Two reference simulations of a splitting supercell storm are generated using LFO and 10ICE microphysics schemes in an identical storm environment. Two sets of OSSEs are conducted from both a perfect and an imperfect model framework using the EnSRF data

assimilation technique using both constant and a range of different intercept and density parameters for the hydrometeors in the ensemble members. Synthetic WSR-88D reflectivity and radial velocity observations at coarser resolution are created from the truth runs using the same radar emulator as in Chapter 3 and these observations are assimilated into the ensemble system over a 30-min period. The 40 ensemble analyses at last assimilation cycle are then used to make 1 h forecasts. Results show that the EnSRF system performs reasonably well with the imperfect model assumption. It is found that a multiparameter ensemble within the imperfect model framework generates more accurate forecasts of ensemble mean precipitation mixing ratios and accumulated rainfall compared to that of the control imperfect model ensemble. This conclusion does not always apply for the perfect model assumption where model error does not play a role. Moreover the 1-h forecast time series of the 40 ensemble members for maximum hail diameter and the lowest cold pool temperature at 100 m AGL indicates that the truth almost always lies within the envelope of ensemble members for the perfect and imperfect MP ensembles, whereas the truth more often lies on the edge or outside the ensemble envelope for the perfect and imperfect control ensemble. The MP ensembles also yield larger ensemble spread than the control experiments. The results from this study support the idea that the microphysical parameter diversity across the ensemble members may be beneficial to a storm-scale ensemble forecasting system.

Both Chapter 3 and Chapter 4 apply the EnSRF data assimilation technique to assimilate radar observations to NWP model. However, due to computational time limitation of the EnSRF algorithm, the synthetic radar observations are sampled at a coarser resolution. The large number of radar observations sampled by the radar and the

benefits of assimilating radar observations of the same storm from multiple radars clearly indicate that the number of observations likely will be very high and the EnSRF data assimilation method may not be an efficient data assimilation method for storm-scale modeling. Thus in an effort to test more efficient data assimilation technique, the novel information filter data assimilation method is implemented using the simple 40 dimensional Lorenz 1996 model in Chapter 5. There are several variants of the information filter in the literature. Chapter 5 of this dissertation explores the extended form of the information filter (EIF) using the state space formulation (Simon 2006) and is compared against the benchmark extended Kalman filter data assimilation technique (EKF). The EIF is used to assimilate three different densities of radar observations. Results indicate that while the rms errors from both EKF and EIF are comparable as shown in Chapter 5, the computational cost of EIF is much higher than that of the EKF. This is due the heavy computation demand of matrix inversion. However, as the number of observations *m* increases from 80 to 160, the computational time for EKF grows by a factor of ~3.27, while for EIF the increment is by a factor of ~1.09. The results obtained clearly indicate that the information filter may be computationally cheaper than that of the Kalman filter when numbers of observations are very high.

Assimilation of high density radar and other remote sensing observations in storm scale modeling is an active area of research. The plan for future works includes the assimilation of real radar observations into the model within realistic storm environment. A broader range of experiments using real radar observations of severe weather events will be conducted using multi-parameter ensemble in storm-scale data assimilation system. Careful selection of these highly uncertain microphysical parameters so that these

values are representative of the various storm systems will be tested. There are several versions of the information filter and we have implemented the state space formulation of the information filter in this study. This study represents only a first step in this direction. There are other formulations of the information filter as discussed in Chapter 2 which leads to the question, which formulation is better? One limitation of transformed state space formulation (Mutbambara 1998) of EIF is that it requires parallel implementation of EKF to obtain the transformed state vector and the information matrix. This poses a very good theoretical question: can we develop an extended nonlinear information filter while totally remaining within the transformed space? This is a difficult problem and will be pursued in the near future.

References

Aksoy, A., D. Dowell, and C. Snyder, 2009a: Ensemble Kalman filter assimilation of real radar observations: A multi-case comparative study of storm-scale forecasts and analyses. *Mon. Wea. Rev.*, **137**, 1805-1824.

Ambadan, J.T., and Y. Tang, 2009: Sigma-Point Kalman Filter Data Assimilation Methods for Strongly Nonlinear Systems. *J. Atmos. Sci.*, **66**, 261–285.

Anderson, B.D.O, and J. B. Moore, 1979: *Optimal Filtering.* Prentice Hall, Inc. Englewook Cliffs, N. J.

Anderson J. L., 2001: An ensemble adjustment filters for data assimilation, *Monthly Weather Review*, **129**, 2884-2903.

Bierman, G. J., 1977: Factorization methods for discrete sequential estimation, Academic Press, 241 pp.

Bishop C. H., B. J. Etherton, and S. J. Majumdar, 2001: Adaptive sampling with the ensemble transform Kalman filter. Part I: Theoretical aspects, *Monthly Weather Review*, **129**, 420–436.

Brandes, E. A., K. Ikeda, G. Zhang, M. Schönhuber, and R. M. Rasmussen, 2007: A statistical and physical description of hydrometeor distributions in Colorado snowstorms using a video disdrometer. *J. Appl. Meteor. Climatol.,* **46,** 634–650.

Burgers G., J. van Leeuwen, and G. Evensen, 1998: Analysis scheme in ensemble Kalman filter, *Monthly Weather Review*, **126**, 1719-1724.

Businger, P. and G. H. Golub, 1965: Linear least squares solution by householder transformations. *Numer. Math*, **7**, 269-276.

Caya, A., J. Sun, and C. Snyder, 2005: A Comparison between the 4DVAR and the Ensemble Kalman Filter Techniques for Radar Data Assimilation. *Mon. Wea. Rev.*, **133**, 3081–3094.

Cifelli, R., C. R. Williams, D. K. Rajopadhyaya, S. K. Avery, K. S. Gage, and P. T. May, 2000: Drop-size distribution characteristics in tropical mesoscale convective systems. *J. Appl. Meteor.,* **39,** 760–777.

Coniglio, M.C., D.J. Stensrud, and L.J. Wicker, 2006: Effects of upper-level shear on the structure and maintenance of strong quasi-linear mesoscale convective systems. *J. Atmos. Sci.*, **63**, 1231-1252.

Cox, H. C. 1964: On the estimation of state variables and parameters for noisy dynamic systems. *IEEE Trans. On A. C.*, **AC-9**, 5-12.

Dowell, D. C., and L. J. Wicker, 2009: Additive noise for storm-scale ensemble forecasting and data assimilation. *J. Atmos. Oceanic Technol*.

Dowell, D. C., F. Zhang, L. J. Wicker, C. Snyder, and N.A. Crook, 2004a: Wind and temperature retrievals in the 17 May 1981 Arcadia, Oklahoma, supercell: Ensemble Kalman filter experiments. *Mon. Wea. Rev.*, **132**, 1982-2005.

Dowell, D. C., L. J. Wicker, and D. J. Stensrud, 2004b: High resolution analyses of the 8 May 2003 Oklahoma City storm. Part II: EnKF data assimilation and forecast experiments. Preprints, *22nd Conf. Severe Local Storms*, Hyannis, MA, Amer. Meteor. Soc.

Dyer, P. and McReynolds, S., 1969: Extension of square-root filtering to include process noise. *J. of Optimization Theory and Applications,* **3**, 444-459.

Evensen, G., 1992: Using the extended Kalman filter with a multi-layer quasigeostrophic ocean model. *J. Geophys. Res.*, **97(C11),** 17905-17924.

Evensen, G., 1994: Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *J. Geophys. Res.*, **99(C5),** 10143-10162.

Evensen, G., 2007: *Data Assimilation: The Ensemble Kalman Filter*. New York: Springer-Verlag.

Fertig E., J. Harlim, and B. R. Hunt, 2007: A comparative study of 4D-VAR and a 4D ensemble Kalman filter: Perfect model simulations with Lorenz-96. *Tellus*, **59A**, 96–101.

Forsyth, D. E., and Coauthors, 2004: The National Weather Radar Testbed (phased-array) becomes operational. Preprints, *20th Int.Conf. on Interactive Information and Processing Systems* (*IIPS*) *for Meteorology, Oceanography, and Hydrology,* Seattle, WA, Amer. Meteor. Soc., CD-ROM, P1.1.

Gaspari, G., and S. E. Cohn, 1999: Construction of correlation functions in two and three dimensions. *Quart. J. Roy. Meteor. Soc.*, **125**, 723-757.

Gilmore, M.S., J.M. Straka, and E.N. Rasmussen, 2004: Precipitation uncertainty due to variations in precipitation particle parameters within a simple microphysics scheme. *Mon. Wea. Rev*., 2004, **132**, 2610-2627.

Golub, G. H., 1965: Numerical methods for solving linear least square problems. *Numer. Math*, **7**, 206-216.

Gunn, K. L. S., and J. S. Marshall, 1958: The distribution with size of aggregate snowflakes. *J. Meteor.,* **15,** 452–461.

Heinselman, P., D. Priegnitz, K. Manross, T. Smith, and R. Adams, 2008: Rapid sampling of severe storms by the National Weather Radar Testbed Phased Array Radar. *Wea. Forecasting*, **23**, 808-824.

Hong, S.-y. and J.-O. J. Lim, 2006: The WRF single-moment 6-class microphysics scheme (WSM6). *J. Korean Meteor. Soc.*, **42,** 129-151.

Householder, A. S., 1964: *Theory of matrices in numerical analysis.* Waltham, Mass.: Blaisdell, ch.5.

Houtekamer P. L., and H. L. Mitchell, 1998: Data assimilation using an ensemble Kalman filter technique, *Monthly Weather Review*, **126**, 796-811.

Houze, R. A., Jr., P. V. Hobbs, P. H. Herzegh, and D. B. Parsons, 1979: Size distributions of precipitation particles in frontal clouds. *J. Atmos. Sci.,* **36,** 156–162.

Houze, R. A., Jr., C.-P. Cheng, C. A. Leary, and J. F. Gamache, 1980: Diagnosis of cloud mass and heat fluxes from radar and synoptic data. *J. Atmos. Sci.,* **37,** 754–773.

Jazwinski, A. H., 1970: *Stochastic processes and filtering theory*. Academic Press, NY.

Jung, Y., G. Zhang, and M. Xue, 2008a: Assimilation of simulated polarimetric radar data for a convective storm using ensemble Kalman filter. Part I: Observation operators for reflectivity and polarimetric variables. *Mon. Wea. Rev*., **136**, 2228– 2245.

Jung, Y., M. Xue, G. Zhang, and J. Straka, 2008b: Assimilation of simulated polarimetric radar data for a convective storm using ensemble Kalman filter. Part II: Impact of polarimetric data on storm analysis. *Mon. Wea. Rev*., **136**, 2246– 2260.

Kalman, R. E., 1960: A new approach to linear filtering and prediction problems. *Transactions of the American Society of Mechanical Engineering, Journal of Basic Engineering*, **82**, 34-45.

Kalman, R. E., and R. S. Bucy, 1961: New results in linear filtering and prediction theory. *Transactions of the American Society of Mechanical Engineering, Journal of Basic Engineering*, **83**, 95-108.

Kaminski P. G., A.E. Bryson Jr., and S. F. Schmidt 1971: Discrete Square Root Filtering: A Survey of Current Techniques. *IEEE Tran. on Automatic Control*, **6**, 727-735.

Lakshmivarahan S., and S. Dhall, 1990: *Analysis and Design of Parallel Algorithms: Arithmetic and Matrix Problems.* Mcgraw Hill Series in Supercomputing and Parallel Processing, 657 pp.

Lakshmivarahan S., and D. J. Stensrud, 2009: Ensemble kalman filter: an innovative approach for meteorological data assimilation. *IEEE Control System Society*, Special Issue, **29**, 34-46.

Lei, T., M. Xue, T. Yu, and M. Teshiba, 2007: Study on the optimal scanning strategies of phase-array radar through ensemble Kalman filter assimilation of simulated data. *33rd Int. Conf. Radar Meteor.*, Cairns, Australia, Amer. Meteor. Soc., CDROM P7.1.

Leutbecher, M., R. Buizza, and L. Isaksen, 2007: Ensemble forecasting and flow-dependent estimates of initial uncertainty. In flow-dependent aspects of data assimilation, *Workshop Proceedings,* ECMWF, Shineld Park, Reading, UK, 185-201.

Lewis J. M., S. Lakshmivarahan, and S. Dhall, 2006: *Dynamic Data Assimilation: A Least Squares Approach*. Cambridge, UK: Cambridge University Press, 654 pp.

Lin, Y.-L., R.D. Farley, and H.D. Orville, 1983: Bulk parameterization of the snow field in a cloud model. *J. Appl. Meteor.*, **22**, 1065-1092.

Lord, S. J., E. Kalnay, R. Daley, G. D. Emmitt, and R. Atlas, 1997: Using OSSEs in the design of the future generation of integrated observing systems. Preprints, *First Symp. on Integrated Observation Systems*, Long Beach, CA, Amer. Meteor. Soc., 45–47.

Lorenz, E. N. 1996. Predictability—a problem partly solved. In Proceedings on predictability, held at ECMWF on 4–8 September 1995.

Lorenz E., 2005: Designing chaotic models. *J. Atmos. Sci.*, **62**, 1574–1587.

Lorenz E., 2006: Predictability—a problem partly solved. *Predictability of Weather and Climate,* T. Palmer and R. Hagedorn, Eds., Cambridge University Press, 40–58.

Lorenz E., and K. A. Emmanuel, 1998: Optimal sites for supplementary weather observations: Simulation with a small model. *J. Atmos. Sci.*, **55**, 399–414.

Marshall, J.S., and McK. Palmer, 1948: The distribution of raindrops with size. *J. Atmos. Sci*., **5**, 165-166.

Maybeck P. S., 1979: *Stochastic Models: Estimation and Control*. New York, NY: Academic Press.

Mitchell, D. L., 1988: Evolution of snow-size spectra in cyclonic storms. Part I: Snow growth by vapor deposition and aggregation. *J. Atmos. Sci.,* **45,** 3431–3451.

Morf, M. and T. Kailath, 1974: Square root algorithm for least square estimation, *IEEE Transaction on Automatic Control*, **20**, 483-497.

Mutambara, G. O., 1998: *Decentralized Estimation and Control for Multisensor Systems.* CRC press.

Nakano, S., G. Ueno, and T. Higuchi, 2007: Merging particle filter for sequential data assimilation, *Nonlin. Processes Geophysics.*, **14,** 395-408.

Ott E., B.R. Hunt, I. Szunyogh, A.V. Zimin, E.J. Kostelich, M. Corazza, E. Kalnay, D.J. Patil, J.A. Yorke, 2004: A local ensemble Kalman filter for atmospheric data assimilation, *Tellus*, **56A,** 415–428.

Pruppacher, H. R., and J. D. Klett, 1978: *Microphysics of Clouds and Precipitation*. Reidel, 714 pp.

Schultz, P., 1995: An explicit cloud physics parameterization for operational numerical weather prediction. *Mon. Wea. Rev.*, **123,** 3331-3343.

Simon, D, 2006: Optimal State Estmation Kalman, H and Nonlinear Approaches, John Wiley and sons, 526 pp

Snook, N. and M. Xue, 2008: Effects of microphysical drop size distribution on tornadogenesis in supercell thunderstorms. *Geophy. Res. Letters,* **35**, L24803, doi:10.1029/2008GL035866

Snyder, C. and F. Zhang, 2003: Assimilation of simulated Doppler radar observations with an ensemble Kalman filter. *Mon. Wea. Rev.*, **131**, 1663-1677.

Sorenson, H. W., 1976: An overview of filtering and stochastic control in dynamic systems, *Control and Dynamic Systems: Advances in Theory and Applications*, **12**, Academic Press, New York.

Stensrud, D.J., 2007: *Parameterization Schemes: Keys to Understanding Numerical Weather Prediction Models.* Cambridge University Press, 459 pp.

Straka, J.M., and E.R. Mansell, 2005: A Bulk Microphysics Parameterizaton with Multiple Ice Precipitation Categories. *J. Appl. Meteor.*, **44**, 445–466.

Tao, W.-K. and J. Simpson, 1993: Goddard cumulus ensemble model. Part I: Model description. *Terres. Atmos. Ocean Sci.*, **4,** 35-72.

Tong, M., and M. Xue, 2005: Ensemble Kalman filter assimilation of Doppler radar data with a compressible nonhydrostatic model: OSS experiments. *Mon. Wea. Rev.*, **133**, 1789-1807.

Tong, M., and M. Xue, 2008a: Simultaneous estimation of microphysical parameters and atmospheric state with radar data and ensemble Kalman filter. Part I: Sensitivity analysis and parameter identifiability. *Mon. Wea. Rev*., **136**, 1630-1648.

Tong, M., and M. Xue, 2008b: Simultaneous estimation of microphysical parameters and atmospheric state with radar data and ensemble Kalman filter. Part II: Parameter estimation experiments. *Mon. Wea. Rev.*, **136**, 1649-1668.

van den Heever, S. C., and W. R. Cotton, 2004: The impact of hail size on simulated supercell storms. *J. Atmos. Sci.,* **61,** 1596–1609.

Weber, M. E., J. Y. N. Cho, J. S. Herd, J. M. Flavin, W. E. Benner, and G. S. Torok, 2007: The next-generation multimission U.S. surveillance radar network. *Bull.* Amer. Meteor. Soc., **88**, 1739–1751.

Weisman, M. L. and J. B. Klemp, 1982: The dependence of numerically simulated convective storms on vertical wind shear and buoyancy. *Mon. Wea. Rev.*, **110**, 504-520.

Whitaker, J. S., and T. M. Hamill, 2002: Ensemble data assimilation without perturbed observations. *Mon. Wea. Rev.*, **130**, 1913-1924.

Wicker L. J., and R. B. Wilhelmson, 1995: Simulation and analysis of tornado development and decay within a three-dimensional supercell thunderstorm. *J. Atmos. Sci*, 52, 2675-2703.

Wicker L. J., and Skamarock, 2002: Time-splitting methods for elastic models using forward time schemes. *Mon. Wea. Rev*., **130**, 2088-2097.

Wilks, D. S., 2006: *Statistical Methods in the Atmospheric Sciences: Second Edition.* Academic Press, Boston, MA, 627 pp.

Wood, V. T., R. A. Brown, and D. Dowell, 2009: Simulated WSR-88D Velocity and Reflectivity Signatures of Numerically-Modeled Tornadoes. *J. Atmos. Oceanic Technol.* 26(5): 876.

Xue, M., M. Tong, and K.K. Droegemeier, 2006: An OSSE framework based on the ensemble square root Kalman filter for evaluating the impact of data from radar networks on thunderstorm analysis and forecasting. *J. Atmos. Oceanic Technol*., 23, 46-66.

Yu, T. Y., M. B. Orescanin, C. D. Curtis, D. S. Zrnic, and D. E. Forsyth, 2007: Beam multiplexing using the phased-array weather radar. *J. Atmos. Oceanic Technol.,* **24,** 616–626.

Yussouf, N., and D. J. Stensrud, 2008: Impact of high temporal frequency radar data assimilation on storm-scale NWP model simulations. Preprints, *24th Conference on Severe Local Storms,* Savannah, GA, USA, Amer. Meteor. Soc., 9B.1.

Yussouf, N., and D. J. Stensrud, 2009: Impact of the variations of precipitation particle parameters within the same microphysics scheme in radar data assimilation using EnKF data assimilation technique. Preprints, *13th Conference on Mesoscale Processes,* Saltlake City, UT, USA, Amer. Meteor. Soc., 18.2.

Yussouf, N., and D. J. Stensrud, 2010a: Impact of Phased Array Radar Observations over a Short Assimilation Period: Observing System Simulation Experiments Using Ensemble Kalman Filter. *Mon. Wea. Rev.*, **138**, 517-538.

Yussouf, N., and D. J. Stensrud, 2010b: Impact of the Variations of Microphysical Parameters in Radar Data Assimilation using Perfect and Imperfect Model Experiments. *Mon. Wea. Rev.*, in review.

Zhang, F., C. Snyder, and J. Sun, 2004: Impacts of initial estimate and observation availability on convective-scale data assimilation with an ensemble Kalman filter. *Mon. Wea. Rev.*, **132**, 1238-1253.

Zrnić, D. S., J. F. Kimpel, D. E. Forsyth, A. Shapiro, G. Crain, R. Ferek, J. Heimmer, W. Benner, T.J. McNellis, R. J. Vogt, 2007: Agile beam phased array radar for weather observations. *Bull. Amer. Meteor. Soc.*, **88**, 1753-1766

Appendix A

Program listing for the radar emulator

```
!---------------------------------------------------------------
!
! File Name: rad_volavg_lib.f90
! Author: Nusrat Yussouf
! This program contains a list of subroutines that are needed to create
! synthetic radar observation using volume averaging technique.
!
!---------------------------------------------------------------

!*****************************************************************
!  SUBROUTINE  volavg
!  This routine calculates a mean Doppler velocity value at
!  the center range, azimuth and elevation of an angular
!  beamwidth volume.
!*****************************************************************

  SUBROUTINE volavg(dopval,dbzval,rg,az,elv,samp_az,samp_rg,samp_el,&
                u,v,w,zmm,rho,xlvl,ylvl,zlvl,xe,ye,ze,nx,ny,nz,&
                iprt,spval,xorg,yorg,zorg,bw,ebw,dbz_thres,&
                pts_az,pts_el,pts_rg)

!*****************************************************************
! Input variables:
!
!  rg           range from a Doppler radar to target
!  az           azimuth angle, measuring clockwise from the  north toward which the radar
!               beam is pointing.
!  elv          elevation angle
!  samp_az    = 1 if smearing is done only in az direction,
!               = 0 no smearing
!  samp_rg    = 1 if smearing is done only in rg direction,
!               = 0 no smearing
!  samp_el     = 1 if smearing is done only in elv direction,
!               = 0 no smearing
!  u,v,w          U, V and W wind component from the 3-D model
!  zdbz,rho       Reflectivity and density from 3-D model
!  xlvl,ylvl,zlvl Scalar grid positions in x,y,z directions from model
!
!  xe,ye,ze       Staggered grid positions in x,y,z directions from model
```

```
!
!  nx,ny,nz      Number of points in x,y and z direction in the model
!  spval         missing data parameter
!  iprt          if .true., then print out parameters for debugging
!  xorg,yorg,zorg distance from radar to lower, left corner of 3-D model
!
!  bw           1-way half-power beamwidth
!  ebw          6dB 1-way effective beamwidth
!  dbz_thres    user specified reflectivity threshold
!
! Ouput variable:
!
!  dopval       mean (volume-weighted) Doppler velocity value
!  dbzval       mean (volume-weighted) reflectivity value
!*****************************************************************

    use param_module
    implicit none

    real     :: u(nx,ny,nz),v(nx,ny,nz),w(nx,ny,nz),zmm(nx,ny,nz)
    real     :: rho(nz),zlvl(nz),xlvl(nx),ylvl(ny),ze(nz)
    real     :: xe(nx),ye(ny)
    real     :: hgt_r,vt,xorg,yorg,zorg,xg_m,yg_m,zg_m
    real     :: bw,ebw,dbz_thres,ures,vres,wres,zres,rhores
    real     :: dopval,dbzval,rg,az,elv,spval
    integer  :: ituth,count,samp_az,samp_rg,samp_el,nx,ny,nz
    integer  :: num_az, num_el,num_rg,pts_az,pts_el,pts_rg

!*****************************************************************
!    local variables
!    vbw          beamwidth in the vertical direction
!    sigma**2     second central moment in deg**2 of the 2-way  antenna pattern
!                 in the azimuth direction
!    sigmb**2     second central moment in deg**2 of the 2-way  antenna pattern in the
!                 elevation direction
!    thetx        max off-axis angle in the azimuth direction
!    dthet        off-axis angle interval in the azimuth    direction
!    phi          off-axis angle in the elevation direction
!    phix         max off-axis angle in the elevation direction
!    dphi         off-axis angle interval in the elevation  direction
!    r1            trailing edge of range weighting function at
!                  which window is zero (w=0.0)
!    r2            trailing edge of range weighting function at
!                  which window is maximum (w=1.0)
!    r3            leading edge of range weighting function at  which window
!                  is maximum
```

```
!    r4        leading edge of range weighting function which window is zero (w=0.0)
!    drg       range increment
!    ae        6/5 earth radius
!    gsum      numerator of the smoothed Doppler velocity
!    fsum      denominator of the smoothed Doppler velocity
!    rg_sub    range subpoint within the beamwidth volume
!    az_sub    azimuth subpoint within the beamwidth volume
!    el_sub    elevation subpoint within the beamwidth volume
!    dopv      Doppler velocity (m/s)
!    zmm       reflectivity in  mm**6/m**3 before calculation
!    Note that the above parameters are used in metric units
!*****************************************************************

     integer :: i,j, k, ii, jj, kk
     real    :: sigma, sigmb, drg, dvbw, debw
     real    :: vbw, vbw_x, ebw_x, r1, r2, r3, r4
     real    :: sum_vr,sum_ref, sum_wtvr, sum_wtrfl
     real    :: fall_spd, zdbzval ,x_sub,y_sub,z_sub
     real    :: sin_sum, cos_sum, bb, f4,w2f4,varx,varz,wt, w2
     real    :: az_sub, el_sub, phi_sub,rg_sub,thet_sub, tlint, lint

    ! Initializing

     count = 0
     vbw = bw
     sigma = ebw/sqrt(16.0*alog(2.0))
     sigmb = vbw/sqrt(16.0*alog(2.0))

    ! elevation angles in the resolution volume and no. of sub points

     vbw_x = vbw*float(samp_el)
     dvbw = 2.*vbw_x/float(pts_el-1)
     if(samp_el.eq.0) then
        num_el = 1
     else
        num_el = pts_el
     end if

    ! azimuthal angles in the resolution volume and no. of sub points

     ebw_x = 1.5*ebw*float(samp_az)
     debw = 2.*ebw_x/(pts_az-1)

     if(samp_az.eq.0) then
        num_az = 1
 else
```

```
        num_az = pts_az
end if

    r1 = rg - 0.13*float(samp_rg)
    r2 = rg - 0.09*float(samp_rg)
    r3 = rg + 0.09*float(samp_rg)
    r4 = rg + 0.13*float(samp_rg)

    if(samp_rg.eq.0) then
        num_rg = 1
        drg = 0.0
    else
        num_rg = pts_rg
        drg=(r4-r1)/float(num_rg-1)
    end if

    sum_vr = 0.0
    sum_wtvr = 0.0
    sum_ref = 0.0
    sum_wtrfl = 0.0

  ! Calculate a mean Doppler velocity/reflectivity value at the center range (rg),
  ! azimuth (az) and elevation (el) of the effective resolution volume within the
   !beamwidth. Calculate slant range within the beamwidth volume.

    do jj = 1, num_rg

    if(num_rg.eq.1) then
       rg_sub = rg
    else
       rg_sub = r1 + float(jj-1)*drg
    endif


  ! Calculate range weighting function.

      wt = 0.0
      if(r1.lt.rg_sub.and.rg_sub.lt.r2) wt = (rg_sub-r1)/0.04
      if(r2.le.rg_sub.and.rg_sub.le.r3) wt = 1.0
      if(r3.lt.rg_sub.and.rg_sub.lt.r4) wt = (r4-rg_sub)/0.04

      w2 = wt*wt

   ! Calculate azimuth within the beamwidth volume.

     do ii = 1, num_az
```

```
   if(num_az.eq.1) then
     phi_sub = 0.0
   else
     phi_sub = -ebw_x + float(ii-1)*debw
   endif

   az_sub = az + phi_sub

   if(samp_az.ne.0) then
     varx = phi_sub*phi_sub/(2.*sigma*sigma)
   elseif(samp_az.eq.0) then
     varx = 0.0
   endif

! Calculate elevation within the beamwidth volume.

 do kk = 1, num_el

   if(num_el.eq.1) then
      thet_sub = 0.0
   else
      thet_sub = -vbw_x + float(kk-1)*dvbw
   endif

   el_sub = elv + thet_sub

   if(samp_el.ne.0) then
      varz = thet_sub*thet_sub/(2.*sigmb*sigmb)
   else
      varz = 0.0
   endif

 ! Calculate height as a function of range and elevation.

     z_sub = hgt_r(el_sub,rg_sub)

 ! Calculate a two-way antenna pattern.

     f4 = exp(-varx-varz)
     w2f4 = w2*f4

! Calculate the sum of the beam's elevation angle to the data point and the angle
! subtended by the verticals at the radar  and at the measurement point.

    bb = rg_sub*cos(el_sub*degtorad)/(ae + rg_sub*sin(el_sub*degtorad))
```

```
      cos_sum = cos(el_sub*degtorad + atan(bb))
      sin_sum = sin(el_sub*degtorad + atan(bb))

! Calculate x and y as a function of range and azimuth.

      x_sub = rg_sub*sin(az_sub*degtorad)*cos_sum
      y_sub = rg_sub*cos(az_sub*degtorad)*cos_sum

 ! Interpolate 3-D gridded data to radar target.

   xg_m = (x_sub - xorg)*1000.0
   yg_m = (y_sub - yorg)*1000.0
   zg_m = (z_sub - zorg)*1000.0

   ures = spval
   vres = spval
   wres = spval
   zres = spval
   rhores = spval

! begin trilinear interpolation.

   zres = tlint(zmm,xg_m,yg_m,zg_m,nx,ny,nz,nx-1,ny-1,nz- &
             1,0,0,xlvl,ylvl,zlvl,spval)
   ures = tlint(u,xg_m,yg_m,zg_m,nx,ny,nz,nx,ny-1,nz-1, 0,0, xe,&
             ylvl, zlvl, spval)
   vres = tlint(v,xg_m,yg_m,zg_m,nx,ny,nz,nx-1,ny,nz-1, 0,0,xlvl,&
             ye, zlvl, spval)
   wres = tlint(w,xg_m,yg_m,zg_m,nx,ny,nz,nx-1,ny-1,nz, 0,0,xlvl,&
             ylvl, ze, spval)

! linear interpolation in the vertical direction.

   rhores = lint( rho, zg_m, nz-1, zlvl, spval)

! Calculate values Doppler velocity, reflectivity factor, and
! terminal fall speed of precipitation.

   dopval = spval
   dbzval = spval
   fall_spd = spval

! Sum up computed Doppler velocity & reflectivity factor within beamwidth volume.

   if(zres.ne.spval) then
```

```
          sum_ref = sum_ref + f4*w2*zres
          sum_wtrfl = sum_wtrfl + f4*w2
       if(ures.ne.spval.and.vres.ne.spval.and.wres.ne.spval.and.&
              rhores.ne.spval) then
          fall_spd = vt(zres,rhores)
          dopval = ures*sin(az_sub*degtorad)*cos_sum + &
                   vres*cos(az_sub*degtorad)*cos_sum& + &
                   (wres+fall_spd)*sin_sum
         endif

        if(dopval.ne.spval) then
          sum_vr = sum_vr + f4*w2*zres*dopval
          sum_wtvr = sum_wtvr + f4*w2*zres
        endif

       endif

        count = count + 1
         end do
        end do
       end do


! Compute a mean Doppler velocity value (m/s) and return the result.

       if(sum_wtvr.ne.0.0) then
          dopval = sum_vr/sum_wtvr
       else
          dopval = spval
       endif

! Compute a mean reflectivity value (dBZ) and return the result.

      if(sum_wtrfl.ne.0.0) then
        zdbzval = sum_ref/sum_wtrfl
        dbzval = 10.*alog10(zdbzval)

        if(dbzval.lt.dbz_thres) then
          dbzval = spval
          dopval = spval
        endif
      else
        dbzval = spval
      endif

RETURN
```

```
  END SUBROUTINE VOLAVG


!**************************************************************
!  SUBROUTINE  volavg
!       This routine calculates a mean Doppler velocity value at the center range, azimuth
!       and elevation of an angular  beamwidth volume.
!**************************************************************

 SUBROUTINE volavg_simple(dopval,dbzval,rg,az,elv,u,v,w,zmm,rho,&
       xlvl,ylvl,zlvl,xe,ye,ze,nx,ny,nz,iprt,spval,xorg,yorg,&
       zorg,dbz_thres,phi,theta,wgt)

!**************************************************************
!    Input variables:
!
!    rg             range from a Doppler radar to target
!    az             azimuth angle, measuring clockwise from the north toward which the
!                    radar beam is pointing
!    elv            elevation angle
!    u,v,w          U, V and W wind component from the 3-D model
!    zdbz,rho       Reflectivity and density from 3-D model
!    xlvl,ylvl,zlvl  Scalar grid positions in x,y,z directions from model
!    xe,ye,ze        Staggered grid positions in x,y,z directions from model
!    nx,ny,nz,       Number of points in x,y and z direction in the model
!    spval           missing data parameter
!    iprt            if .true., then print out the parameters for debugging
!    xorg,yorg,zorg  distance from radar to the lower, left corner of 3-D model
!    dbz_thres       user specified reflectivity threshold
!
!    Ouput variable:
!    dopval    mean (volume-weighted) Doppler velocity value
!    dbzval    mean (volume-weighted) reflectivity value
!**************************************************************

     use param_module
     implicit none

     real    :: u(nx,ny,nz),v(nx,ny,nz),w(nx,ny,nz),zmm(nx,ny,nz)
     real    :: xlvl(nx),ylvl(ny),zlvl(nz),ze(nz),xe(nx),ye(ny)
     real    :: rho(nz),hgt_r,vt,xg_m,yg_m,zg_m,dbz_thres
     real    :: ures,vres,wres,zres,rhores,rg,az,elv
     real    :: spval,dopval,dbzval,xorg,yorg,zorg
     real    :: phi(13),theta(13),wgt(13)
     real    :: az_sub,el_sub,rg_sub,f4
     real    :: sum_vr,sum_ref, sum_wtvr, sum_wtrfl
```

```
      real     :: fall_spd, zmmval,x_rad,y_rad,z_rad
      real     :: sin_sum, cos_sum, bb,lint, tlint
      integer  :: nx,ny,nz, i, j, k
      integer  :: pts(13),cnt, count, iwgt
      logical  :: flag

      sum_vr = 0.0
      sum_wtvr = 0.0
      sum_ref = 0.0
      sum_wtrfl = 0.0


    ! A mean Doppler velocity/reflectivity value at the center range
    ! (rg),azimuth (az) and elevation (el) of the effective resolution
    ! volume within the beamwidth can be approximated by computing the
    ! weighted mean of individual Doppler velocity/reflectivity values
    ! over the 13 points.

      cnt = 0
      pts = 0
      flag = .false.

      do iwgt = 1, 13

        rg_sub = rg
        az_sub = az + phi(iwgt)
        el_sub = elv + theta(iwgt)
        z_rad = hgt_r(el_sub,rg_sub)

     ! Calculate a two-way antenna pattern.

        f4 = wgt(iwgt)

    ! Write down the computed variables.

        if(iprt) then
          write(6,1) iwgt,az_sub,el_sub,rg_sub,f4
        endif
1     format(' iwgt=',i2,' az_sub,el_sub,rg_sub=',3f8.3,'f4=',f8.4)

    ! Calculate the sum of the beam's elevation angle to the data  point and the angle
    !subtended by the verticals at the radar  and at the measurement point.

      bb =rg_sub*cos(el_sub*degtorad)/(ae+rg_sub*sin(el_sub*degtorad))
      cos_sum = cos(el_sub*degtorad + atan(bb))
      sin_sum = sin(el_sub*degtorad + atan(bb))
```

```
  if(iprt)write(6,*) 'bb,cos_sum, sin_sum ', bb, cos_sum,sin_sum


! Calculate x and y as a function of range and azimuth.

  x_rad = rg_sub*sin(az_sub*degtorad)*cos_sum
  y_rad = rg_sub*cos(az_sub*degtorad)*cos_sum


 ! Interpolate 3-D gridded data to radar target.

   xg_m = (x_rad - xorg)*1000.0
   yg_m = (y_rad - yorg)*1000.0
   zg_m = (z_rad - zorg)*1000.0

   if(iprt)write(6,*) 'x_rad,xorg,xg_m',x_rad,xorg,xg_m
   if(iprt)write(6,*) 'y_rad,yorg,yg_m',y_rad,yorg,yg_m
   if(iprt)write(6,*) 'z_rad,zorg,zg_m',z_rad,zorg,zg_m

  ures = spval
  vres = spval
  wres = spval
  zres = spval
  rhores = spval

!  begin trilinear interpolation.

  zres = tlint(zmm,xg_m,yg_m,zg_m,nx,ny,nz,nx-1,ny-1,nz-1, 0, 0,&
              xlvl,ylvl,zlvl,spval)

  ures = tlint(u,xg_m,yg_m,zg_m,nx,ny,nz,nx,ny-1,nz-1, 0,0, xe, &
              ylvl, zlvl, spval)

  vres = tlint(v,xg_m,yg_m,zg_m,nx,ny,nz,nx-1,ny,nz-1, 0,0, xlvl, &
                        ye, zlvl, spval)

  wres = tlint(w,xg_m,yg_m,zg_m,nx,ny,nz,nx-1,ny-1,nz, 0,0, xlvl,&
              ylvl, ze, spval)

! linear interpolation in the vertical direction.
    rhores = lint( rho, zg_m, nz-1, zlvl, spval)

! Calculate values Doppler velocity, reflectivity factor, and terminal fall speed of
! precipitation.

    if(iprt) then
```

```
        write(6,*) 'ures, vres, wres, zres, rhores', ures, vres,wres, zres, rhores
       endif

       fall_spd = spval

     ! calculate non-missing variable within the beam volume.

      if(ures.ne.spval.and.vres.ne.spval.and.wres.ne.spval.and. &
        rhores.ne.spval.and.zres.ne.spval) then
        fall_spd = vt(zres,rhores)
        dopval = ures*sin(az_sub*degtorad)*cos_sum + &
                 vres*cos(az_sub*degtorad)*cos_sum + &
                 (wres+fall_spd)*sin_sum
        cnt = cnt + 1
        pts(iwgt) = 1
        sum_ref = sum_ref + f4*zres
        sum_wtrfl = sum_wtrfl + f4
        sum_vr = sum_vr + f4*zres*dopval
        sum_wtvr = sum_wtvr + f4*zres

       endif

     end do

!  Compute a mean Doppler velocity value (m/s) and return the result.

     if (cnt .eq. 13 ) then
         flag = .true.
     else if ( cnt .ge. 9 ) then
      if ( (pts(1).eq.1).and.(pts(2).eq.1).and.(pts(3).eq.1).and. &
           (pts(4) .eq. 1).and.(pts(5).eq.1).and.(pts(9).eq.1) .and. &
           (pts(10).eq.1).and.(pts(11).eq.1).and.(pts(13).eq. 1)) then
           flag = .true.
      else if ((pts(1).eq.1).and.(pts(8).eq.1).and.(pts(7).eq.1) .and.&
           (pts(6).eq.1).and.(pts(5).eq.1).and.(pts(9).eq.1).and. &
           (pts(12).eq.1).and.(pts(11).eq.1).and.(pts(13).eq.1)) then
           flag = .true.
      else if ((pts(7).eq.1).and.(pts(8).eq.1).and.(pts(1).eq.1).and. &
           (pts(2).eq.1).and.(pts(3).eq.1).and.(pts(12).eq. 1) .and. &
           (pts(9).eq.1).and.(pts(10).eq.1).and.(pts(13).eq. 1)) then
           flag = .true.
      else if ((pts(7).eq.1).and.(pts(6).eq.1).and.(pts(5).eq.1) .and.&
           (pts(4).eq. 1).and.(pts(3).eq.1).and.(pts(12).eq.1) .and. &
           (pts(11).eq.1).and.(pts(10).eq.1).and.(pts(13).eq.1)) then
           flag = .true.
       end if
```

```
      end if

      if (flag) then

        if(sum_wtvr.ne.0.0) then
          dopval = sum_vr/sum_wtvr
        else
          dopval = spval
        endif

     ! Compute a mean reflectivity value (dBZ) and return the result.

        if(sum_wtrfl.ne.0.0) then
          zmmval = sum_ref/sum_wtrfl
          dbzval = 10.*alog10(zmmval)
          if(dbzval.lt.dbz_thres) then
            dbzval = spval
            dopval = spval
          endif
        else
          dbzval = spval
          dopval = spval
        endif
      else
          dbzval = spval
          dopval = spval
       endif

      RETURN
      END SUBROUTINE volavg_simple

!**********************************************************************
!  SUBROUTINE setup
!    This routine defines a sector of interest on ppi.  the input and  output data are listed.
!
!    Variable descriptions:
!
!    azmbeg      begining azimuth
!    azmend      ending azimuth
!    azmref      reference azimuth
!    grid_x      perimeter size in the x-direction
!    grid_y      perimeter size in the y-direction
!    rngbeg      begining range
!    rngend      ending range
!    rngref      reference range
!    xo,yo       reference center relative to the reference range and azimuth
```

```
!**********************************************************************
  SUBROUTINE setup(grid_x, grid_y, azmbeg, azmend, rngbeg, rngend, &
                   rngref, azmref,xo,yo)

      use param_module
      implicit none

      real             :: grid_x, grid_y
      real             :: azmbeg, azmend
      real             :: rngbeg, rngend
      real             :: rngref, azmref
      real             :: xo, yo
      character(LEN = 1) :: iprt

! Calculate x- and y-positions of the grid origin relative to
! the reference range and azimuth.

      xo = rngref*sin(azmref*degtorad)
      yo = rngref*cos(azmref*degtorad)
      grid_x = anint(grid_x)
      grid_y = anint(grid_y)

! Define a sector of interest.
! Compute begining and ending ranges and azimuths.

      call set_up(xo,yo,grid_x,grid_y,azmbeg,azmend,rngbeg,rngend)

! Nearest the whole number.

      azmbeg = anint(azmbeg)
      if(azmbeg.lt.0.0) azmbeg = 0.0
      azmend = anint(azmend)
      if(azmend.gt.360.0) azmend = 360.0
      rngbeg = anint(rngbeg)
      if(rngbeg.lt.0.0) rngbeg = 0.0
      rngend = anint(rngend)

      if((azmend-azmbeg).eq.360.) then
        azmbeg = azmref - 180.
        if(azmbeg.lt.0.0) azmbeg = azmbeg + 360.
        azmend = azmref + 180.
        if(azmend.gt.360.0) azmend = azmend - 360.
      endif

      iprt = 'n'
      if(iprt.eq.'y') then
```

```
      write(6,25)
   25 format(/)
      write(6,19) grid_x, grid_y
   19 format('   Grid size (km) in x- and y-directions: ',2f5.0)
      write(6,27)
   27 format('   Input data are given as:')
      write(6,20) rngref
   20 format('     Reference range (km): ',f7.2)
      write(6,21) azmref
   21 format('     Reference azimuth (deg): ',f6.1)
      write(6,28)
   28 format('   Output data are given as:')
      write(6,22) xo, yo
   22 format('     Perimeter center(xo,yo)(km)from the radar: ' 2f8.2)
      write(6,23) azmbeg, azmend
   23 format('     Begining and ending azimuths (deg): ',2f7.1)
      write(6,24) rngbeg, rngend
   24 format('     Begining and ending ranges (km): ',2f7.1)
      endif

  RETURN
  END SUBROUTINE SETUP


!  ***********************************************************************
!  SUBROUTINE: set_up
!      This routine computes beginning and ending ranges and azimuths of a sector scan.
!
!      Variable descriptions:
!
!      adjazm      add a few more azimuths to the begining or ending azimuths (deg)
!      adjrng      add some km to the begining and ending ranges
!      degtorad    convert degress to radians
!      gx          array of x-distances (nm) from radar to the side of the perimeter
!      gy          array of y-distances (nm) from radar to the side  of the perimeter
!      pi          = 3.1415709
!      prt         character that determines whether or not you want  to print out data
!      px,py       x,y-locations (nm) of the perimeter's center from  the radar
!      range       range (nm) of corner from the radar
!      rmax        maximum distance (nm) of the perimeter's corner   from the radar
!      rmin        minimum distance (nm) of the perimeter's corner   from the radar
!      rtd         convert radians to degrees
!      semp        array of ranges (nm) of the perimeter's corner
!      temp        array of angles (deg) of the perimeter's corner
!      x_dist      distance of the perimeter along the x-axis
!      y_dist      distance of the perimeter along the y-axis
```

```
!
!    ***********************************************************************

  SUBROUTINE set_up(px,py,x_dist,y_dist,azmbeg,azmend,rngbeg,rngend)

     use param_module
     implicit none

     character(LEN = 1) :: prt
     real            :: gx(2),gy(2),temp(4),semp(4)
     logical          :: igo
     real            :: rngbeg,rngend,azmbeg,azmend
     real            :: px,py,x_dist,y_dist
     integer          :: m, i, j
     real            :: a, b, c
     real            :: t, t2, s, rmin, rmax

!  Compute four corners (km) of the perimeter.

     gx(1) = px - 0.5*x_dist
     gx(2) = px + 0.5*x_dist
     gy(1) = py - 0.5*y_dist
     gy(2) = py + 0.5*y_dist
     rngbeg = 999.
     rngend = 999.
     azmbeg = 999.
     azmend = 999.
     igo = .false.

!  If the radar is inside the perimeter, then get one full ppi scan.

     if((gx(1)*gx(2).lt.0.0).and.(gy(1)*gy(2).lt.0.0)) then
       azmbeg = 0.0
       azmend = 359.0
       rngbeg = 0.0
     endif

!  Compute angle of each corner of the perimeter.

     m = 0
     do i = 1, 2
       do j = 1, 2
         m = m + 1
         temp(m) = atan2(gx(i),gy(j))*rtd
         semp(m) = sqrt(gx(i)*gx(i) + gy(j)*gy(j))
       enddo
```

```
      enddo

      do m = 1, 4
        if(temp(m).lt.0.0) temp(m) = temp(m) + 360.
      enddo

      if(rngbeg.eq.0.0) go to 4

!  Determine beginning and ending azimuths and ranges.

      if(gx(1).ge.0.0) then
        if(gy(1).gt.0.0) then
          azmbeg = temp(2)
          azmend = temp(3)
        elseif(gy(1).le.0.0.and.gy(2).ge.0.0) then
          azmbeg = temp(2)
          azmend = temp(1)
          igo = .true.
          a = semp(2)
          b = semp(1)
          c = y_dist
        elseif(gy(2).lt.0.0) then
          azmbeg = temp(4)
          azmend = temp(1)
        endif
      elseif(gx(2).le.0.0) then
        if(gy(1).gt.0.0) then
          azmbeg = temp(1)
          azmend = temp(4)
        elseif(gy(1).le.0.0.and.gy(2).ge.0.0) then
          azmbeg = temp(3)
          azmend = temp(4)
          igo = .true.
          a = semp(3)
          b = semp(4)
          c = y_dist
        elseif(gy(2).lt.0.0) then
          azmbeg = temp(3)
          azmend = temp(2)
        endif
      elseif(gy(1).ge.0.0) then
        if(gx(1).gt.0.0) then
          azmbeg = temp(2)
          azmend = temp(3)
        elseif(gx(1).le.0.0.and.gx(2).ge.0.0) then
          azmbeg = temp(1)
```

```
      azmend = temp(3)
      igo = .true.
      a = semp(1)
      b = semp(3)
      c = x_dist
    elseif(gx(2).lt.0.0) then
      azmbeg = temp(1)
      azmend = temp(4)
    endif
  elseif(gy(2).le.0.0) then
    if(gx(1).gt.0.0) then
      azmbeg = temp(4)
      azmend = temp(1)
    elseif(gx(1).le.0.0.and.gx(2).ge.0.0) then
      azmbeg = temp(4)
      azmend = temp(2)
      igo = .true.
      a = semp(4)
      b = semp(2)
      c = x_dist
    elseif(gx(2).lt.0.0) then
      azmbeg = temp(3)
      azmend = temp(2)
    endif
  endif

4 continue

  rmax = -1.e10
  rmin = +1.e10

  do i = 1, 4
    if(semp(i).ge.rmax) rmax = semp(i)
    if(rngbeg.ne.0.0) then
      if(semp(i).le.rmin) rmin = semp(i)
    endif
  enddo

  if(rngbeg.ne.0.0) rngbeg = rmin
  rngend = rmax

  if(igo) then
    s = 0.5*(a + b + c)
    t2 = s*(s-a)*(s-b)*(s-c)
    t = sqrt(t2)
    rngbeg = 2.*t/c
```

```
      endif

!  Adjust beginning and ending azimuths by adjusting a few more azimuths to them.

      rngbeg = rngbeg - adjrng
      if(rngbeg.lt.0.0) rngbeg = 0.0
      rngend = rngend + adjrng
      azmbeg = azmbeg - adjazm
      azmend = azmend + adjazm

!       Do you want to print out data of corners (km)?  prt = 'y' or 'n'

      prt = 'n'
      if(prt.eq.'y') then
        write(6,6) gx(1),gy(2)
    6   format(1x,'x,y positions in the nw corner (km)= ',2f8.2)
        write(6,7) gx(2),gy(2)
    7   format(1x,'x,y positions in the ne corner (km)= ',2f8.2)
        write(6,8) gx(2),gy(1)
    8   format(1x,'x,y positions in the se corner (km)= ',2f8.2)
        write(6,9) gx(1),gy(1)
    9   format(1x,'x,y positions in the sw corner (km)= ',2f8.2)
      endif

  RETURN
  END SUBROUTINE set_up


!****************************************************************
! Function: hgt_r
!    This function computes the height (km) of the data value.
!    Input:  el   elevation angle (deg)
!              rng  slant range (km) from radar
!    Output: computed hgt_r
!****************************************************************

  REAL FUNCTION hgt_r(el,rng)

      use param_module
      implicit none
      real :: el, ng

      hgt_r = sqrt(rng*rng + ae*ae + 2.*ae*rng*sin(el*degtorad)) - ae

  RETURN
  END FUNCTION hgt_r
```

```
!***********************************************************************
!  Function: vt
!      This function computes the terminal fall speed of hydrometeors  (a negative quantity
!      toward the ground).  Pecipitation is assumed to be liquid water.
!     zdbz     reflectivity factor (mm**6/m**3)
!     rho     density (kg/m**3)
!***********************************************************************

   REAL FUNCTION vt(zdbz,rho)

      use param_module
      implicit none
      real :: zdbz, rho

      vt = -2.6*zdbz**(0.107)*(1.2/rho)**0.4

   RETURN
   END FUNCTION vt



!***********************************************************************
! Subroutine: comp_wgt
!      This subroutine approximates the mean values of Doppler velocity and reflectivity
!      within a half-power beamwidth by computing the weighted mean of individual
!       values over the 13 points.
!***********************************************************************

   SUBROUTINE comp_wgt(bw,ebw,phi,theta,wgt)

     use param_module
     implicit none

     real    :: bw,ebw
     real    :: phi(13),theta(13),wgt(13)
     integer :: i

     vbw = bw

     do i = 1, 13

       if(i.le.8) then
         wgt(i) = 0.5
         ang = (i-1)*45.*degtorad
         phi(i) = 0.5*ebw*sin(ang)
```

```
        theta(i) = 0.5*vbw*cos(ang)
      endif

      if(9.le.i.and.i.le.12) then
        wgt(i) = 0.84
        ang = (i-9)*90.*degtorad
        phi(i) = 0.25*ebw*sin(ang)
        theta(i) = 0.25*vbw*cos(ang)
      endif

      if(i.eq.13) then
        wgt(i) = 1.0
        phi(i) = 0.0
        theta(i) = 0.0
      endif

    enddo

  RETURN
  END SUBROUTINE comp_wgt



!**********************************************************************
! Subroutine: radar_calc
!    This subroutine calculates the radar parameters from the input data.
!**********************************************************************

SUBROUTINE radar_calc(radar_lat,radar_lon,ini_grid_lat_deg, &
                ini_grid_lon_deg,range_interval,azim_interval,&
                nx,ny,nz,dx,dy,dz,map_proj,xmin,ymin,spval,nazm,&
                azm_beg,nrng,rng_beg,xorg,yorg,xradar,yradar,&
                rng_ref,azm_ref)


  use param_module
  real     :: dx,dy,dz,xorg,yorg
  real     :: rng_beg,rng_end,azm_beg,azm_end
  real     :: rng_ref, azm_ref
  real     :: xo, yo, grid_x, grid_y, grid_z
  real     :: mid_x, mid_y
  real     :: ini_grid_lat,ini_grid_lon,ini_grid_lon_deg,
  real     :: grid_lat,grid_lon,grid_lon_deg,grid_lat_deg
  real     :: ini_grid_lat_deg, mid_lon_deg
  real     :: radar_lat,radar_lon,mid_lat,mid_lon,mid_lat_deg,
  real     :: range_interval,azim_interval
```

```
  real      :: xmin, ymin, xradar,yradar, gnd_rng
  integer   :: nrng, nazm, map_proj
  integer   :: nx,ny,nz


  ini_grid_lat = ini_grid_lat_deg * degtorad
  ini_grid_lon = ini_grid_lon_deg * degtorad

! model dimensions  and model mid point distance

  grid_x = (nx-1)*dx/1000.    ! in km
  grid_y = (ny-1)*dy/1000.    ! in km
  grid_z = (nz-1)*dz/1000.    ! in km
  mid_x  = grid_x*1000.0/2.0  ! in m
  mid_y  = grid_y*1000.0/2.0  ! in m

! Model SW corner lat/lon at current time (due to ugrid and vgrid motion)

  call xy_to_ll(grid_lat, grid_lon, map_proj, xmin, ymin, ini_grid_lat, ini_grid_lon)

! Model mid point lat/lon relative to SW lat/lon corner of the model

  call xy_to_ll(mid_lat, mid_lon, map_proj, mid_x, mid_y, grid_lat, grid_lon)

  mid_lat_deg = mid_lat * rtd
  mid_lon_deg = mid_lon * rtd
  grid_lat_deg = grid_lat * rtd
  grid_lon_deg = grid_lon * rtd

! Calculating the distance of model center point relative to radar location

  call ll_to_xy(xradar,yradar,map_proj,radar_lat,radar_lon,mid_lat,&
              mid_lon)

  xradar = xradar/1000.0  ! in km
  yradar = yradar/1000.0  ! in km

! Calculating the reference range  and azimuth of radar relative  to the model center point.

   rng_ref = sqrt(xradar**2 + yradar**2)  ! in km

   if ( (xradar.eq.0.0) .and. (yradar.eq.0.0) ) then
          azm_ref = 0.0
   else if(yradar .gt. 0.0) then
          azm_ref = rtd*atan(xradar/yradar)
   else if (yradar .lt. 0.) then
```

```
          azm_ref= pii*rtd + rtd*atan(xradar/yradar)
    else if (xradar .gt. 0.) then
          azm_ref = 0.50*pii*rtd
    else
          azm_ref = 1.5* pii*rtd
    end if

    if (azm_ref .lt. 0.0) azm_ref =azm_ref + 2*pii*rtd

    xo = rng_ref*sin(azm_ref*degtorad)  ! in km
    yo = rng_ref*cos(azm_ref*degtorad)  ! in km

! Compute xorg, yorg, zorg defined as the distance of the lower, left
! corner of the model from the radar location.

    xorg = xo - grid_x/2.  ! in km
    yorg = yo - grid_y/2.  ! in km

! Call to determine beginning and ending ranges and azimuths of model grid from radar


    call setup(grid_x,grid_y,azm_beg,azm_end,rng_beg,rng_end,rng_ref,&
             azm_ref,xo,yo)

    nrng = (rng_end - rng_beg)/range_interval + 1

    if(azm_end.gt.azm_beg) then
         nazm = (azm_end - azm_beg)/azim_interval + 1
    else
         nazm = (azm_end + 360. - azm_beg)/azim_interval + 1
    endif

    if(nazm.le.0) then
       print*,'program is terminated bcz nazm <= 0'
       print*,'check to fix the problem!'
       stop
    endif

! Use an open statement to create a new formatted write file on unit 6
! with a file name 'dop_rad_simul.out'

    open(6,file='dop_rad_simul.out',status='unknown')

    write(6,*)'radar_la/lon deg= ',radar_lat*rtd, radar_lon* rtd
    write(6,*)'grid lat/lon of SW corner after moving from start= &
             ',grid_lat_deg,grid_lon_deg
```

```
  write(6,*) 'middle point of model grid lat/lon= ',mid_lat_deg,&
             mid_lon_deg
  write(6,*) 'distnce of domain middle point from SW corner mid_x/y &
             = ', mid_x, mid_y
  write(6,*) 'distance of radar relative to model center point &
             xradar/yradar = ', xradar, yradar
  write(6,*) 'xmin, ymin = ', xmin, ymin
  write(6,901) rng_ref
901   format(1x,'Reference Range (km) to Center Grid of Data    ',/, &
          4x,'Volume, Relative to Radar.....',f6.1)
      write(6,902) azm_ref
902   format(1x,'Reference Azimuth (deg) to Center Grid of Data ',/, &
          4x,'Volume, Relative to Radar.....',f6.1)
      write(6,906) range_interval
906   format(1x,'Range Interval (km).....',f7.2)
      write(6,9061) azim_interval
9061  format(1x,'Azimuth Interval (deg).....',f6.1)
      write(6,9068) dbz_thres
9068  format(1x,'Reflectivity threshold (dbz).....',f6.1)
      write(6,907) xo
907   format(1x,'X-Dist (km) Relative to Radar.....',f6.1)
      write(6,908) yo
908   format(1x,'Y-Dist (km) Relative to Radar.....',f6.1)
      write(6,909) dx/1000.0
909   format(1x,'Grid Spacing (km) in X-Direction.....',f6.1)
      write(6,910) dy/1000.0
910   format(1x,'Grid Spacing (km) in Y-Direction.....',f6.1)
      write(6,911) dz/1000.0
911   format(1x,'Grid Spacing (km) in Z-Direction.....',f6.1)
      write(6,912) grid_x
912   format(1x,'Grid Size (km) in X-Direction.....',f6.1)
      write(6,913) grid_y
913   format(1x,'Grid Size (km) in Y-Direction.....',f6.1)
      write(6,914) grid_z
914   format(1x,'Grid Size (km) in Z-Direction.....',f6.1)
      write(6,915) nx, ny, nz
915 format(1x,'Data Array Size (nx).....',i5,/,&
          1x,'Data Array Size (ny).....',i5,/, &
          1x,'Data Array Size (nz).....',i5)
      write(6,916) xorg
916   format(1x,'X-Origin in Lower, Left Corner of 3-D Model....',f6.1)
      write(6,917) yorg
917   format(1x,'Y-Origin in Lower, Left Corner of 3-D Model....',f6.1)
      write(6,919) rng_beg
919   format(1x,'Beginning Range (km).....',f6.1)
```

```
      write(6,920) rng_end
920   format(1x,'Ending Range (km).....',f6.1)
      write(6,921) azm_beg
921   format(1x,'Beginning Azimuth (deg).....',f6.1)
      write(6,922) azm_end
922   format(1x,'Ending Azimuth (deg).....',f6.1)
       write(6,923) nrng
923     format(1x,'No. of Range Gates.....',i5)
      write(6,924) nazm
924     format(1x,'No. of Azimuth Gates.....',i5)

     close(6)

    RETURN
    END SUBROUTINE radar_calc

!**********************************************************************
! Subroutine: twrite
!     Write data down to a output file.
!**********************************************************************

  SUBROUTINE twrite(nobs,nobs_x,dopv,dbz,azm,rng,elv,lu,zres_st,hgt_st)
     implicit none

     real    ::  elv
     real    ::  dopv(nobs_x), dbz(nobs_x)
     real    ::  rng(nobs_x), azm(nobs_x)
     real    ::  zres_st(nobs_x),hgt_st(nobs_x)
     integer ::  lu     ! file unit number
     integer ::  nobs, nobs_x, i

     do i = 1, nobs
       write(lu,10) i, dopv(i), dbz(i), rng(i), azm(i), &
                  elv,zres_st(i), hgt_st(i)
     enddo

10 format('i=',i6,' dopv=',f7.2,' dbz=',f7.2,' rng=',f8.3,' &
          azm=',f5.0,'elv=',f7.2,' zres=',f7.2,' hgt=',f7.2)

  RETURN
  END SUBROUTINE twrite


! **********************************************************************
! SUBROUTINE: SYNTHETIC_RADAR_OBS_VOL
! Author:     Nusrat Yussouf
```

```
!
!     This subroutine produces synthetic Doppler velocity and reflectivity observations
!     from the truth model run using volumetric averaging and outputs the results to a file.
!     Observations are produced as follows:
! ********************************************************************

SUBROUTINE SYNTHETIC_RADAR_OBS_VOL(obfile,obformat,runfile,time,secs,&
        days, refl_threshold_for_vr,dbz_thres, radar_loc_flag,&
        radar_lat_deg, radar_lon_deg,map_proj,rand_error_refl, &
        rand_error_vr, bias_error_refl,bias_error_vr, beamwidth,&
        eff_bw,azim_interval,range_interval,vcp_num,samp_az,&
        samp_rg,samp_el,pts_az,pts_el,pts_rg,radar_sample_flag,sweeps)

   use ens_module
   use ob_module
   use dart_module
   use random
   use param_module

   implicit none

   character(LEN = *)   :: obfile      ! name of observation file
   character(LEN = 120) :: runfile     ! has namelists
   integer    :: obformat     ! observation format:
                              ! 0=radar polar
                              ! 1=radar PPI
                              ! 2=DART
   integer    :: time         ! model time (seconds)
   integer    :: secs,days    ! Time in DART format
                        !(Gregorian days and seconds)
   integer    :: radar_loc_flag ! 1=location specified by
                         ! radar_lat and radar_lon,
                         ! 2=U observed, 3=V observed
   integer    :: map_proj       ! map projection (for relating
                         ! lat, lon to x, y):
                         !   0 = flat earth
   integer    :: start, end, tilt,obs_kind obs_cnt
   integer    :: nrng, nazm samp_az, samp_rg, samp_el
   integer    :: pts_az, pts_el, pts_rg
   integer    :: radar_sample_flag,sweeps
   integer    :: i,j,k,lu count
   integer    :: nobs,n ,s ! number of valid observations
   integer    :: nx,ny,nz, vcp_num, vr_count, dbz_count


   real       :: dbz_thres  ! lowest reflectivity for which
```

```
                    ! Doppler obs. are produced
   real      :: refl_threshold_for_vr ! lowest reflectivity
                                ! (dbz) for which Doppler
                                ! velocities are produced
   real      :: radar_lat_deg        ! radar latitude (deg)
   real      :: radar_lon_deg        ! radar longitude(deg)
   real      :: rand_error_refl      ! standard deviation of
                                ! random reflectivity errors

   real      :: rand_error_vr        ! standard deviation (m/s) of
                                ! random radial-velocity errors
   real      :: bias_error_refl      ! bias errors for reflectivity
   real      :: bias_error_vr        ! bias error for radial &
                                ¡ velocity

!  Local variables

    integer, parameter :: nobs_x = 70000
    integer, parameter :: max_obs = 100000
    integer, parameter :: itruth = -1
    real Nyquist_vel; parameter(Nyquist_vel=0.0) ! Nyquist velocity
    logical :: iprt,flag

    character(LEN = 5) :: ich5
    character(LEN = 4) :: ich4
    character(LEN = 3) :: ich3
    character(LEN = 3) :: jch3
    character(LEN = 35) :: file_name
    character(LEN = 1) :: iopn

    real,    pointer  :: xlvl(:)
    real,    pointer  :: xe(:)
    real,    pointer  :: ylvl(:)
    real,    pointer  :: ye(:)
    real,    pointer  :: zlvl(:)
    real,    pointer  :: ze(:)
    real,    pointer  :: pz(:)
    real,    pointer  :: tz(:)
    real,    pointer  :: u(:,:,:)
    real,    pointer  :: v(:,:,:)
    real,    pointer  :: w(:,:,:)
    real,    pointer  :: t3(:,:,:)
    real,    pointer  :: p3(:,:,:)
    real,    pointer  :: zdbz(:,:,:)
    real, allocatable :: rho(:)
    real, allocatable :: rf_true(:) ! error-free reflectivity
```

```
                          ! observtions (dBZ)
real, allocatable :: vr_true(:) ! error-free radial velocity
                          ! observations (m/s), fallspeed
                          ! component included
real, allocatable :: azm(:)     ! azimuth angle (radians during
                          ! computation, deg during output)
 real, allocatable :: elev(:)    ! elevation angle (radians during
                           ! computation, deg during output)
 real, allocatable :: rng(:)

 real, allocatable :: drf_true(:,:,:) ! error-free reflectivity
                              ! observtions (dBZ)
 real, allocatable :: dvr_true(:,:,:) ! error-free radial velocity
                              ! observations (m/s), with
                              ! fallspeed component
 real, allocatable :: drf(:,:,:)      ! reflectivity observtions
 real, allocatable :: dvr(:,:,:)      ! radial velocity
                              ! observations (m/s),with
                              ! fallspeed component
 real, allocatable :: drf_mod(:,:,:)  ! reflectivity observations
 real, allocatable :: dvr_mod(:,:,:)  ! radial velocity
                              ! observations (m/s), with
                              ! fallspeed component
 real, allocatable :: azmd(:,:,:)     ! azimuth angle (radians
                              ! during computation, deg
                              ! during output)
 real, allocatable :: elevd(:,:,:)    ! elevation angle (radians
                              ! during computation, deg
                              ! during output)
 real, allocatable :: rngd(:,:,:)
 real, allocatable :: zmm(:,:,:)

 real(kind=8), allocatable :: height(:,:,:)
 real(kind=8), allocatable :: olat(:,:,:)
 real(kind=8), allocatable :: olon(:,:,:)

 real :: dx,dy,dz,xorg,yorg,zorg,bw,ebw
 real :: rng_beg,rng_end,azm_beg,azm_end
 real :: rng_ref, azm_ref, azm_rad
 real :: xo, yo, grid_x, grid_y, grid_z, az,rg, elv,
 real :: dopval, dbzval, bb, cossum,sinsum
 real :: x_rg, y_rg, z_rg, xg_m, yg_m, zg_m
 real :: ures,vres,wres,zres,rhores
 real :: grid_alt   ! grid origin altitude (m MSL)
 real :: xmin, ymin ! coordinates (m) of southwest corner of model
 real :: beamwidth, eff_bw, range_interval,azim_interval
```

```
      real :: hgt_r, mid_x, mid_y, lat, lon, radar_lat,radar_lon
      real :: rlat, rlon, rheight,error_variance
      real :: grid_lat,grid_lon, grid_lon_deg, grid_lat_deg
      real :: ini_grid_lat,ini_grid_lon, ini_grid_lon_deg
      real :: mid_lat, mid_lon, mid_lat_deg,mid_lon_deg, &
              ini_grid_lat_deg
      real :: xradar,yradar
      real :: phi(13),theta(13),wgt(13), zres_st(nobs_x),&
              hgt_st(nobs_x)
      real :: tlint, lint, rad_vel, fall_spd, vt


! Doppler Radar VCP

       real :: vcp11(14)
       real :: vcp12(14)
       real :: vcp15(20)
       real :: vcp16(14)
       real :: vcp17(25)
       real, allocatable :: nswp(:)

       data vcp11 /0.50,1.45,2.40,3.35,4.30,5.25,6.20,7.50,8.70,10.00,&
                  12.00, 14.00,16.70,19.50 /
       data vcp12 /0.50,0.90,1.30,1.80,2.40,3.10,4.00,5.10,6.40, &
                  8.00,10.00,12.50,15.60,19.50 /
       data vcp15 /0.50,1.45,2.40,3.35,4.30,5.25,6.20,7.20,8.20,9.20,&
                  10.20,11.70,13.20,14.70,16.20,17.70,19.20,20.70,&
                  22.2, 23.7 /
       data vcp16 /0.50,1.10,1.70,2.40,3.20,4.10,5.10,6.20,7.40,8.70,&
                   10.10, 11.70, 13.50,15.50 /

!  Output various information.

     write(*,*)
     write(*,*) 'RAD_VOL_PARAM'
     write(*,*) '-----------------'
     write(*,*) 'time = ', time
     write(*,*) 'obfile = ', obfile
     write(*,*) 'obformat = ', obformat
     write(*,*) 'map_proj = ', map_proj
     write(*,*) 'beamwidth = ', beamwidth
     write(*,*) 'effective beamwidth = ', eff_bw
     write(*,*) 'range interval = ', range_interval
     write(*,*) 'azim interval = ', azim_interval
     write(*,*) 'vcp number = ', vcp_num
     write(*,*) 'samp_az = ', samp_az
```

```
write(*,*) 'samp_rg = ', samp_rg
write(*,*) 'samp_el = ', samp_el
write(*,*) 'pts_az = ', pts_az
write(*,*) 'pts_el = ', pts_el
write(*,*) 'pts_rg = ', pts_rg
write(*,*) 'refl_threshold_for_vr = ',  refl_threshold_for_vr
write(*,*) 'dbz_thres = ',      dbz_thres
write(*,*) 'sweeps = ',      sweeps

bw = beamwidth
ebw = eff_bw

if ( vcp_num .eq. 11) then
    tilt = 14
    allocate( nswp(tilt) )
    nswp = vcp11
end if
if ( vcp_num .eq. 12) then
    tilt = 14
    allocate( nswp(tilt) )
    nswp = vcp12
end if
if ( vcp_num .eq. 15) then
    tilt = 20
    allocate( nswp(20) )
    nswp = vcp15
end if
if ( vcp_num .eq. 16) then
    tilt = 14
    allocate( nswp(tilt) )
    nswp = vcp16
end if
if ( vcp_num .eq. 17) then
    tilt = 25
    allocate( nswp(tilt) )
    nswp = vcp17
end if


CALL GET_VARIABLE(ens%g(itruth), 'NX',       nx)
CALL GET_VARIABLE(ens%g(itruth), 'NY',       ny)
CALL GET_VARIABLE(ens%g(itruth), 'NZ',       nz)
CALL GET_VARIABLE(ens%g(itruth), 'DX',       dx)
CALL GET_VARIABLE(ens%g(itruth), 'DY',       dy)
CALL GET_VARIABLE(ens%g(itruth), 'DZ',       dz)
CALL GET_VARIABLE(ens%g(itruth), 'LAT',      ini_grid_lat_deg )
```

```
  CALL GET_VARIABLE(ens%g(itruth), 'LON',      ini_grid_lon_deg)
  CALL GET_VARIABLE(ens%g(itruth), 'HGT',      grid_alt)
  CALL GET_VARIABLE(ens%g(itruth), 'XG_POS',   xmin)
  CALL GET_VARIABLE(ens%g(itruth), 'YG_POS',   ymin)

  CALL GET_VARIABLE(ens%g(itruth),'XC',xlvl)
  CALL GET_VARIABLE(ens%g(itruth),'YC',ylvl)
  CALL GET_VARIABLE(ens%g(itruth),'ZC',zlvl)
  CALL GET_VARIABLE(ens%g(itruth),'XE',xe)
  CALL GET_VARIABLE(ens%g(itruth),'YE',ye)
  CALL GET_VARIABLE(ens%g(itruth),'ZE',ze)
  CALL GET_VARIABLE(ens%g(itruth),'THINIT',tz)
  CALL GET_VARIABLE(ens%g(itruth),'PIINIT',pz)
  CALL GET_VARIABLE(ens%g(itruth), 'U',       u)
  CALL GET_VARIABLE(ens%g(itruth), 'V',       v)
  CALL GET_VARIABLE(ens%g(itruth), 'W',       w)
  CALL GET_VARIABLE(ens%g(itruth), 'DBZ',    zdbz)

  radar_lat = radar_lat_deg * degtorad
  radar_lon = radar_lon_deg * degtorad

  allocate( rho(nz) )
  call density(rho, nz, pz, tz)
  allocate( zmm(nx,ny,nz) )

!     Convert dbz to mm**6/m**3 before calculation.

     do k = 1, nz
      do j = 1, ny
        do i = 1, nx
        if(zdbz(i,j,k).ne.spval) then
            zmm(i,j,k) = 10.**(zdbz(i,j,k)/10.)
         endif
        enddo
      enddo
    enddo

  call radar_calc(radar_lat,radar_lon,ini_grid_lat_deg, &
       ini_grid_lon_deg,range_interval,azim_interval,nx,ny,nz,dx, &
       dy,dz,map_proj,xmin,ymin,spval, nazm,azm_beg,nrng,rng_beg, &
       xorg,yorg,xradar,yradar,rng_ref,azm_ref)

  zorg = grid_alt
  rheight = grid_alt + ze(1)  ! in m

! allocate output observations
```

```
      allocate(vr_true(nobs_x))
      allocate(rf_true(nobs_x))
      allocate(rng(nobs_x))
      allocate(azm(nobs_x))
      allocate(elev(nobs_x))

      allocate(dvr_true(tilt,nazm,nrng))
      allocate(drf_true(tilt,nazm,nrng))
      allocate(dvr(tilt,nazm,nrng))
      allocate(drf(tilt,nazm,nrng))
      allocate(dvr_mod(tilt,nazm,nrng))
      allocate(drf_mod(tilt,nazm,nrng))
      allocate(rngd(tilt,nazm,nrng))
      allocate(azmd(tilt,nazm,nrng))
      allocate(elevd(tilt,nazm,nrng))
      allocate(height(tilt,nazm,nrng))
      allocate(olat(tilt,nazm,nrng))
      allocate(olon(tilt,nazm,nrng))

! initialization of observations

      dvr_true = spval
      drf_true = spval
      dvr_mod = spval
      drf_mod = spval
      dvr = spval
      drf = spval
      rngd = spval
      azmd = spval
      elevd = spval
      height = spval
      olat = spval
      olon = spval

!     Call to produce simulated Doppler velocity and reflectivity
!     values by scanning the radar across a 3-D reflectivity
!     structure as a function of range, azimuth and elevation
!     angles.

      obs_cnt = 0
      if (radar_sample_flag.eq.3) call comp_wgt(bw,ebw,phi,theta,wgt)

!     PAR observations, entire volume scan every minute
      if (sweeps .eq. 0) then
        start = 1
```

```
      end = tilt

!     WSR-88D observations, 2-3 sweeps every minute. Out of 14 sweeps
!     the lower 12 sweeps are generated every 3 sweeps per minute for
!     the first 4 min while the remaining upper 2 sweeps are valid at
!     the last minute.

      else if (sweeps .eq. 1) then
        if ( mod(time, 300) .eq. 0 ) then
           start = 1
           end = 3
        else if ( mod(time, 300) .eq. 60 ) then
           start = 4
           end = 6
        else if ( mod(time, 300) .eq. 120 ) then
            start = 7
            end = 9
        else if ( mod(time, 300) .eq. 180 ) then
            start = 10
            end = 12
         else if ( mod(time, 300) .eq. 240 ) then
           start = 13
           end = 14
         end if
      end if
      print *, time, start, end

        do k = start,end
          write(7, *) ' Elevation angle starts .....'
        elv = nswp(k)
        nobs = 0
        count = 0

 ! **  print statement **

      lu = 10
      write(ich5,23) elv
23     format(f5.2)

      if(ich5(1:1).eq.' ') ich5(1:1) = '0'

      write(ich3,24) ifix(rng_ref)
24     format(i3.3)

      write(ich4,26) time
26     format(i4)
```

```
      write(jch3,24) ifix(azm_ref)

      iprt = .false.
      if (iprt) open(6,file='debug'//ich5//'out',status='unknown')

      do i = 1, nazm
        az = azm_beg + (i-1)*azim_interval
        if(az.ge.360.) az = az - 360.

        do j = 1, nrng
          rg = rng_beg + j*range_interval - range_interval/2.
          dopval = spval
          dbzval = spval
          rad_vel = spval
          bb = rg*cos(elv*degtorad)/(ae + rg*sin(elv*degtorad))
          cossum = cos(elv*degtorad + atan(bb))
          sinsum = sin(elv*degtorad + atan(bb))

!  Target distance from radar in x,y and z direction

          x_rg = rg*sin(az*degtorad)*cossum ! in km
          y_rg = rg*cos(az*degtorad)*cossum ! in km
          z_rg = hgt_r(elv,rg)  ! in km

          count = count + 1

!   Target distance from model southwest corner in x,y and z direction

          xg_m = (x_rg - xorg)*1000.0  ! in m
          yg_m = (y_rg - yorg)*1000.0  ! in m
          zg_m = (z_rg - zorg)*1000.0  ! in m

          zres = tlint(zdbz,xg_m,yg_m,zg_m,nx,ny,nz,nx-1,ny-1, &
                    nz-1,0,0,xlvl,ylvl,zlvl,spval)
          ures = tlint(u,xg_m,yg_m,zg_m,nx,ny,nz,nx,ny-1,nz-1,0,0,xe, &
                    ylvl, zlvl, spval)
          vres = tlint(v,xg_m,yg_m,zg_m,nx,ny,nz,nx-1,ny,nz-1,0,0, &
                    xlvl,ye, zlvl, spval)
          wres = tlint(w,xg_m,yg_m,zg_m,nx,ny,nz,nx-1,ny-1,nz,0,&
                    0,xlvl,ylvl, ze, spval)
          rhores = lint( rho, zg_m, nz-1, zlvl, spval)

          if(ures.ne.spval.and.vres.ne.spval.and.wres.ne.spval &
                          .and.rhores.ne.spval) then
            fall_spd = vt(zres,rhores)
```

```
          rad_vel = ures*sin(az*degtorad)*cossum+ vres*cos &
                  (az*degtorad)*cossum&+(wres+fall_spd)*sinsum
         endif

        if(zres.ne.spval) then
           if(zres.ge.dbz_thres) then

!    if interpolated reflectivity value (dbz) is at or above a  prescribed reflectivity
!    threshold, then call to produce a mean Doppler velocity and reflectivity values at the
!    center of the beamwidth volume.
          if (iprt) then
             write(6,*) 'Loop starts   ', count
             write(6,*) rg, az, elv
             write(6,*) 'x_rg,xorg,xg_m', x_rg,xorg,xg_m
             write(6,*) 'y_rg,yorg,yg_m', y_rg,yorg,yg_m
             write(6,*) 'z_rg,zorg,zg_m', z_rg,zorg,zg_m
             write(6,*)'After interpol_3D_dbz, zres ',zres
          end if

          write(7,*) 'Loop starts ,zres, rg ', zres, rg

          if(nobs.gt.nobs_x) then
            write(6,925)
925  format('Program is terminated bcz nobs > nobs_x.', &
           ' You need to increase your nobs_x dimension',' in the main code.')
            stop
           endif

         if (radar_sample_flag .eq. 2) then

         call volavg(dopval,dbzval,rg,az,elv,samp_az,samp_rg,samp_el, &
                   u,v,w,zmm,rho,xlvl,ylvl,zlvl,xe,ye,ze,nx,ny,nz, &
                   iprt, spval,xorg,yorg,zorg,bw,ebw,dbz_thres, &
                   pts_az,pts_el,pts_rg )

          elseif (radar_sample_flag .eq. 3) then

          call volavg_simple(dopval,dbzval,rg,az,elv,u,v,w,zmm,rho, &
                   xlvl,ylvl,zlvl,xe,ye,ze,nx,ny,nz,iprt,spval, &
                   xorg,yorg,zorg,dbz_thres,phi,theta,wgt)

          endif

       endif

!    save the obs. for verification
```

```
      write(7,*) 'dbzval,dopval ', dbzval, dopval
      if ((dopval.ne.spval).and.(dbzval.ne.spval).and. (dbzval.ge.dbz_thres))then
          nobs = nobs + 1
          rf_true(nobs) = dbzval
          if (dbzval.gt.dbz_thres) then
              vr_true(nobs) = dopval
          else
              vr_true(nobs) = 0.0
          endif
          rng(nobs) = rg
          azm(nobs) = az
          zres_st(nobs) = zres
          hgt_st(nobs) = hgt_r(elv,rg)
      end if

!     save the obs. for DART file
          call xy_to_ll(lat,lon,map_proj,x_rg*1000,y_rg*1000,  radar_lat,radar_lon)
          rngd(k,i,j) = rg
          azmd(k,i,j) = az
          elevd(k,i,j) = elv
          height(k,i,j) = grid_alt + 1000.00 * hgt_r(elv,rg)   ! in m
          olat(k,i,j) = lat
          olon(k,i,j) = lon

          if ((dbzval.ne.spval).and.(dbzval.ge.dbz_thres))then
              drf_true(k,i,j) = dbzval
              drf(k,i,j) = drf_true(k,i,j) + bias_error_refl + &
                           rand_error_refl*random_normal()
              drf_mod(k,i,j) = zres
              obs_cnt = obs_cnt + 1
          else
              drf_true(k,i,j) = spval
              drf(k,i,j) = spval
              drf_mod(k,i,j) = spval
          endif

          If ((dopval.ne.spval).and. &
              (dbzval.ge.refl_threshold_for_vr))then
              dvr_true(k,i,j) = dopval
              dvr(k,i,j) = dvr_true(k,i,j) + bias_error_vr + &
                           rand_error_vr*random_normal()
              dvr_mod(k,i,j) = rad_vel
              obs_cnt = obs_cnt + 1
          else
              dvr_true(k,i,j) = spval
              dvr(k,i,j) = spval
```

```
            dvr_mod(k,i,j) = spval
          endif

          endif  !zres ne spval
        enddo
        enddo

      print*,'nobs',nobs

      if(nobs.eq.0) then
        print*,'prog is terminated bcz nobs = 0 in subr radar_scan'
      endif

! Open a new output file and connect the file to a unit 'lu'.
! The file lists computed Doppler velocity, reflectivity, range, azimuth, and elevation.
 file_name='simul_'//ich4//'_rg'//ich3//'_az'//jch3//'_elv'//ich5//'.dat

      print*,'file_name ',file_name

      iopn = 'y'

      if(iopn.eq.'y') then

        open(lu,file=file_name,status='unknown')

        write(lu,25) rng_ref, azm_ref, elv
 25 format('Code name: dop_rad_simul.f',/,'rng_ref=',f4.0,'  $
        azm_ref=',f4.0,' elv=',f6.2)

       call twrite(nobs,nobs_x,vr_true,rf_true,azm,rng,elv,lu,zres_st,&
                hgt_st)

        endif
        close(6)

        enddo

!Write synthetic obs to file

      print *, obs_cnt
      obformat = 0
      write(*,*) 'SYNTHETIC RADAR OBS FORMAT 0:  DART time (secs,
              days)= ', secs, days

      open(unit=obfileunit,file=obfile,status='unknown')
```

```
max_num_obs = obs_cnt
num_copies = 2
call write_DART_header(obfileunit)

rlat = radar_lat
rlon = radar_lon

n = 0
vr_count = 0
dbz_count = 0

  do k = start,end
  do i = 1,nazm
   do j = 1, nrng

   if (drf_true(k,i,j).ne.spval) THEN
      n = n + 1
      dbz_count = dbz_count + 1
      obs_kind = obs_kind_reflectivity
      error_variance = rand_error_refl**2

     CALL WRITE_DART_OB(obfileunit,n,drf(k,i,j), &
     drf_true(k,i,j),olat(k,i,j), olon(k,i,j), height(k,i,j),&
     azmd(k,i,j),elevd(k,i,j),Nyquist_vel,dbz_count,rlat, &
     rlon,rheight,obs_kind,secs,days,error_variance,&
     range_interval,bw, ebw, azim_interval,drf_mod(k,i,j))
    endif

   if (dvr_true(k,i,j).ne.spval) then

      n = n + 1
      vr_count = vr_count + 1
      obs_kind = obs_kind_Doppler_velocity
      error_variance = rand_error_vr**2

      CALL WRITE_DART_OB(obfileunit,n,dvr(k,i,j), &
      dvr_true(k,i,j), olat(k,i,j),olon(k,i,j),height(k,i,j),&
      azmd(k,i,j),elevd(k,i,j),Nyquist_vel,vr_count,rlat,rlon,&
      rheight, obs_kind,secs,days,error_variance, &
      range_interval,bw,ebw,azim_interval,dvr_mod(k,i,j))
    end if

 enddo
 enddo
 enddo
```

```
    close(obfileunit)

    deallocate(vr_true)
    deallocate(rf_true)
    deallocate(rng)
    deallocate(azm)
    deallocate(height)
    deallocate(rho)
    deallocate(olat)
    deallocate(olon)
    deallocate(dvr_true)
    deallocate(drf_true)
    deallocate(dvr)
    deallocate(drf)
    deallocate(rngd)
    deallocate(azmd)
    deallocate(elevd)

  RETURN
  END SUBROUTINE synthetic_radar_obs_vol

! ********************************************************************************
```

Appendix B

Program listing for EKF and EIF using Lorenz model

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:             lorenz.m
%
% Description:    Simulate the Lorenz 96 model from the manuscript
%                 "Predictability: A problem partly solved".
%                  The model mimics the time evolution of an unspecified
%                  scalar meteorological quantity x, at J equidistant
%                  grid points along a latitude circle. A fourth-order
%                  Runge-Kutta time integration scheme with a timestep
%                  of 0.05 unit.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;
clear all;
close all;
format short;
%
% modelresolution is the number of points in the model around a circle.
modelresolution=40;
% forcing is the forcing term for the Lorenz 96 model.
forcing=8.;
% timestep is set to represent 0.25 days each time interval.
timestep=0.05;
% initial_model_perturbation is the initial perturbation given to the true solution
initial_truth_perturbation=0.008;
% initial perturbation given to the ensembles of the Lorenz 96 model.
perturbation_range=0.0001;
% Number of time steps to integrate the model to get the initial condition
init_time = 14400; % 10 years.
% Number of ensembles to create the initial X0 and P0
ens = 100;
% Plot identifier
figid = 1;
% Each state variable of the Lorenz model is initialized randomly from a uniform
% distribution between 0 and 1.
Truth=random('unif',0,1,[modelresolution,1])*forcing;
% Apply perturbation to truth
Truth(20)=Truth(20)+initial_truth_perturbation;
% Initialization of the ensemble of lorenz model
```

```
X=zeros(modelresolution,ens);
X(:,1) = Truth;
for i = 2:ens
   X(:,i)= X(:,1);
   indx = round(random('unif',1,40));
   X(indx, i) = X(indx,i) + perturbation_range*randn(1);
end
%
% Plot Ensemble members, Ensemble mean and truth at initial time
%
figure(figid);
figid = figid + 1;
plot(Truth(:),'g-','LineWidth',3);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Grid points'); ylabel('Truth/Ens.mean/Ens.members');
title('Ensemble members,mean and truth at initial time (T = 0)')
axis([1 40 -10 10]);
hold on
plot(mean(X(:,2)),'b-','LineWidth',3);
plot(X(:,1),'k-','LineWidth',1);
for i = 2:ens
   plot(X(:,i),'k-','LineWidth',1);
end
hold off
%
% Plot the ensemble members and the truth at specified grid points
%
figure(figid);
figid = figid + 1;
y = 1:ens;
pt = 10;
plot(y,X(pt,:,1),'b-','LineWidth',3);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Grid points'); ylabel('Truth/Ensemble members');
title(['Ensemble members and truth at T=0 at grid point', num2str(pt)])
hold on
plot(Truth(pt,1),'go','LineWidth',3);
hold off
%
figure(figid);
figid = figid + 1;
y = 1:ens;
pt = 20;
plot(y,X(pt,:,1),'b-','LineWidth',3);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Grid points'); ylabel('Truth/Ensemble members');
```

```
title(['Ensemble members and truth at T=0 at grid point', num2str(pt)])
hold on
plot(Truth(pt,1),'go','LineWidth',3);
hold off
%
figure(figid);
figid = figid + 1;
y = 1:ens;
pt = 30;
plot(y,X(pt,:,1),'b-','LineWidth',3);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Grid points'); ylabel('Truth/Ensemble members');
title(['Ensemble members and truth at T=0 at grid point', num2str(pt)])
hold on
plot(Truth(pt,1),'go','LineWidth',3);
hold off
%
% initialize the truth and the model state for filtering and for archiving.
X0 = zeros(modelresolution,1);
TR0 = zeros(modelresolution,1);
P0 = zeros(modelresolution,modelresolution);
Xf=zeros(modelresolution,ens);
Xf_timeseries=zeros(modelresolution,ens,init_time);
TR0_timeseries=zeros(modelresolution,init_time);
% Integrate the model and truth forward in time for init_time time-steps and store the
values
for i  = 1:init_time
   [TR0]=model_fcst(Truth,timestep,forcing);
   for j = 1: ens
       [Xf(:,j)]=model_fcst(X(:,j),timestep,forcing);
   end
   Xf_timeseries(:,:,i) = X(:,:);
   TR0_timeseries(:,i) = Truth(:);
   Truth = TR0 ;
   X = Xf;
end
% climatological mean and standard deviation
mean_clim_grid = mean(TR0_timeseries,2);
std_clim_grid = std(TR0_timeseries,0,2);
mean_clim = mean(mean_clim_grid,1);
%Calculating Covariance
X0 = mean(Xf, 2);
err_temp_1 = zeros(modelresolution,1);
err_temp_2 = zeros(modelresolution,modelresolution);
```

```
for i = 1:ens
   err_temp_1(:) = Xf(:,i) - X0(:);
   err_temp_2 = err_temp_2 + err_temp_1*err_temp_1';
end
P0 = ( 1/(ens-1) ) * err_temp_2;
%
% plot the final ensembles, truth and mean-ensemble
%
figure(figid);
figid = figid + 1;
plot(Xf(:,1),'b-','LineWidth',1);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Grid Points'); ylabel('Ensembles / Mean / Truth');
title(['After Integrating',num2str(init_time),' time steps'])
axis([1 40 -15 15]);
hold on
plot(X0(:),'r-','LineWidth',3);
plot(TR0(:),'g-','LineWidth',3);
for i = 2:ens
  plot(Xf(:,i),'b-','LineWidth',1);
end
hold off
% plot the final P0
figure(figid);
figid = figid + 1;
[C,h] = contour(P0,10);
clabel(C,h);
title('Covariance P0 at the end of time steps')
colormap cool
%
%Write out the final values in a file.
%
fid = fopen('init.txt', 'wt');
fprintf(fid, '%8.4f\n', mean_clim);
fprintf(fid, '%8.4f\n', mean(std_clim_grid));
fprintf(fid,'\n');
for i = 1:modelresolution
   fprintf(fid, '%8.4f\n', X0(i));
end
fprintf(fid,'\n');
for i = 1:modelresolution
   fprintf(fid, '%8.4f\n', TR0(i));
end
fprintf(fid,'\n');
for i = 1:modelresolution
   fprintf(fid, '%8.4f  ', P0(:,i));
```

```
    fprintf(fid,'\n');
end
fclose(fid)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:          ekf.m
%
% Description:    First Order Extended Kalman Filter
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;
clear all;
close all;
format short;
%
% modelresolution is the number of points of the model around a sphere.
modelresolution=40;
% forcing is the forcing term for the Lorenz 96 model.
forcing=8.;
% timestep is set to represent 0.05 days each time interval.
timestep=0.01;
% obs_percent variable sets the number of observations relative to the
% model resolution.
obs_percent=100.;
%obs_percent=200.;
%obs_percent=400;
% approximate model error stdev
sig_q=0.3;
sig_obs_fac = 0.25;
% number of assimilation time steps.
assim_time = 500;
figid = 1;
%
% initialize the truth and the state for the Lorenz model
fid = fopen('C:\Experiments\init.txt', 'r');
mean_clim = fscanf(fid, '%g\n', [1,1]);
sig_clim = fscanf(fid, '%g\n', [1,1]);
fscanf(fid,'\n');
[X0, number_of_values_read] = fscanf(fid,'%g\n',[40,1]);
fscanf(fid,'\n');
[TR0, number_of_values_read] = fscanf(fid,'%g\n',[40,1]);
fscanf(fid,'\n');
```

```
[P0, number_of_values_read] = fscanf(fid,'%g\n',[40,40]);
fclose(fid);
% 100% represents observation at every model grid point. The number of
% observations is set based upon the modelresolution and obs_percent.
%
iobs = obs_percent/100.;
% Observations based on observation densities
switch iobs
% obs at every grid and at 3 additional points in between two grids
case 4.0
   obsloc=1:modelresolution*4;         % locations of  obs.
   sig_obs =sig_obs_fac*sig_clim;       % stdev of obs error
   Y = 0.25:0.25:40;
% obs at every grid and at 1 additional points in between two grids
 case 2.0
   obsloc=1:modelresolution*2;         % locations of  obs.
   sig_obs =sig_obs_fac*sig_clim;       % stdev of obs error
   Y = 0.5:.5:40;
 % obs at every gridpoint
 case 1.0
   obsloc=1:modelresolution;          % locations of  obs.
   sig_obs =sig_obs_fac*sig_clim;     % stdev of obs error
   Y = 1:1:40;
otherwise
  disp('Unknown.');
end % select
%
% Initialize model and observation error covariance matrix.
nobs=size(obsloc,2);
R = sig_obs^2 * eye(nobs,nobs);% obs. error term
Q = sig_obs^2 * eye(modelresolution,modelresolution);
% observations vector holds the observations.
Z=zeros(nobs,1);
% H is the observation operator matrix which will transform from the
% model space into the observation space.
H=zeros(nobs,modelresolution);
% Kalman Gain
K=zeros(modelresolution,nobs);
% Creating the H operator
if (iobs == 4)
  mdim = 1;
  for i = 1:modelresolution-1
     H(mdim,i) = 1;
     H(mdim+1,i) = 0.75; H(mdim+1,i+1) = 0.25;
     H(mdim+2,i) = 0.50; H(mdim+2,i+1) = 0.50;
     H(mdim+3,i) = 0.25; H(mdim+3,i+1) = 0.75;
```

```
      mdim = mdim + 4;
   end
   i = modelresolution;
   H(mdim,i) = 1;
   H(mdim+1,i) = 0.75; H(mdim+1,1) = 0.25;
   H(mdim+2,i) = 0.50; H(mdim+2,1) = 0.50;
   H(mdim+3,i) = 0.25; H(mdim+3,1) = 0.75;

elseif (iobs == 2 )
    ndim = 1;
    for mdim=1:nobs
       if (mod(mdim,2) ~= 0)
          H(mdim, ndim) = 1;
          ndim = ndim + 1;
       else
          if (ndim > modelresolution)
             H(mdim, 1) = 0.50;
          else
             H(mdim, ndim) = 0.50;
          end
          H(mdim, ndim-1) = 0.50;
       end
    end
else

    for iobs=1:nobs
       H(iobs, obsloc(iobs)) = 1;
    end
end
%
% Initialze for data archival
truth_timeseries = ones(modelresolution,assim_time);
Xf_timeseries =ones(modelresolution,assim_time);
Pf_timeseries=ones(modelresolution,modelresolution,assim_time);
Xhat_timeseries=ones(modelresolution,assim_time);
Phat_timeseries=ones(modelresolution,modelresolution,assim_time);
Z_timeseries=ones(nobs,assim_time);
%
% Extended Kalman Filter
eps_obs=R*randn(nobs,1);
tic % starts a stopwatch timer.
for t=1:assim_time
     if t == 1
       Xa0 = X0;
       Pa0 = P0;
     else
```

```
        Xa0 = Xa1;
        Pa0 = Pa1;
        TR0 = TR1 ;
      end
% Forecast Step          I
  [TR1]=model_fcst(TR0,timestepforcing);
  [Xf1]=model_fcst(Xa0,timestep,forcing);
  M   = jacobian_fcst(timestep, Xa0);
  Pf1 = M * Pa0 * M' + Q;

%Data assimilation step

  % Generate observations error and observations
  Z=H*TR1;
  Z = Z + eps_obs;
  % Assimilate observations using EKF
  inov = Z - H*Xf1;                % innovation vector
  K = Pf1*H'*inv( R + H * Pf1 * H'); %Kalman gain
  Xa1 = Xf1 + K*inov; %analysis and analysis error covariance matrix
  Pa1 = (eye(modelresolution,modelresolution) - K*H) * Pf1;

  % Save the data
  truth_timeseries(:,t)=TR0(:);
  Pf_timeseries(:,:,t)=Pf1(:,:);
  Xf_timeseries(:,t)=Xf1(:);
  Z_timeseries(:,t)=Z(:);
  Xhat_timeseries(:,t) = Xa1(:);
  Phat_timeseries(:,:,t)=Pa1(:,:);

end % end of assim_time
% toc prints the elapsed time since tic was used.
time = toc % returns the elapsed time.

% rms error calculation
xf_err = zeros(assim_time,modelresolution);
rms_fcst = zeros(assim_time,1);
xa_err = zeros(assim_time,modelresolution);
rms_an = zeros(assim_time,1);
for i = 1:assim_time
   xf_err(i,:)   = Xf_timeseries(:,i) - truth_timeseries(:,i);
   xf_err2(i,:)  = xf_err(i,:) .* xf_err(i,:);
   rms_fcst(i) = sqrt(sum(xf_err2(i,:))/modelresolution );
   %
   xa_err(i,:)   = Xhat_timeseries(:,i) - truth_timeseries(:,i);
   xa_err2(i,:)  = xa_err(i,:) .* xa_err(i,:);
   rms_an(i) = sqrt(sum(xa_err2(i,:))/modelresolution );
```

```
end
%
figure(figid);
plot(truth_timeseries(:,1),'g-','LineWidth',1);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Grid Points'); ylabel('Truth / Observations /Forecast / Analyses');
title(['After ',num2str(1),' time steps of DA'])
hold on
plot(Xf_timeseries(:,1),'b-','LineWidth',1);
axis([1 40 -15 15]);
plot(Y',Z_timeseries(:,1),'k*','LineWidth',1);
plot(Xhat_timeseries(:,1),'r-','LineWidth',1);
hold off
figid = figid + 1;
%
figure(figid);
plot(truth_timeseries(:,assim_time/2),'g-','LineWidth',1);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Grid Points'); ylabel('Truth / Observations /Forecast / Analyses');
title(['After ',num2str(assim_time/2),' time steps of DA'])
hold on
plot(Xf_timeseries(:,assim_time/2),'b-','LineWidth',1);
axis([1 40 -15 15]);
plot(Y',Z_timeseries(:,1),'k*','LineWidth',1);
plot(Xhat_timeseries(:,assim_time/2),'r-','LineWidth',1);
hold off
figid = figid + 1;
%
figure(figid);
plot(truth_timeseries(:,assim_time),'g-','LineWidth',1);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Grid Points'); ylabel('Truth / Observations /Forecast / Analyses');
title(['After ',num2str(assim_time),' time steps of DA'])
hold on
plot(Xf_timeseries(:,assim_time),'b-','LineWidth',1);
axis([1 40 -15 15]);
plot(Y',Z_timeseries(:,1),'k*','LineWidth',1);
plot(Xhat_timeseries(:,assim_time),'r-','LineWidth',1);
hold off
figid = figid + 1;
% plot average errors
figure(figid);
xtime= (1:assim_time) * timestep * 5;
plot(xtime',rms_fcst,'b-','LineWidth',1);
title('RMS Error During Data Assimilation')
xlabel('Time in Days'); ylabel('rms error');
```

```
hold on;
plot(xtime',rms_an,'r-','LineWidth',1);
hold off;
figid = figid + 1;
% Contour Plots of Phat
figure(figid);
[C,h] = contour(Phat_timeseries(:,:,assim_time),10);
clabel(C,h);
title(['Update Covariance after the last assimilation cycle'])
set(h,'ShowText','on','TextStep',get(h,'LevelStep')*2)
colormap cool
figid = figid + 1;
%
figure(figid);
[C,h] = contour(Pf_timeseries(:,:,assim_time),10);
title(['Forecast Covariance before the last assimilation cycle'])
set(h,'ShowText','on','TextStep',get(h,'LevelStep')*2)
colormap cool;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:             eif.m
%
% Description:  Extended Information Filter (State Representation)
%                   is implemented using the 40 dimensional Lorenz model.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;
clear all;
close all;
format short;
%
% modelresolution is the number of points of the model around a sphere.
modelresolution=40;
% forcing is the forcing term for the Lorenz 96 model.
forcing=8.;
% timestep is set to represent 0.05 days each time interval.
timestep=0.01;
% obs_percent variable sets the number of observations relative to the
% model resolution.
obs_percent=100.;
%obs_percent=200.;
%obs_percent=400;
```

```
% approximate model error stdev
sig_q=0.3;
sig_obs_fac = 0.25;
% number of assimilation time steps.
assim_time = 500;
figid = 1;
%
% initialize the truth and the state for the Lorenz model
fid = fopen('C:\Experiments\init.txt', 'r');
mean_clim = fscanf(fid, '%g\n', [1,1]);
sig_clim = fscanf(fid, '%g\n', [1,1]);
fscanf(fid,'\n');
[X0, number_of_values_read] = fscanf(fid,'%g\n',[40,1]);
fscanf(fid,'\n');
[TR0, number_of_values_read] = fscanf(fid,'%g\n',[40,1]);
fscanf(fid,'\n');
[P0, number_of_values_read] = fscanf(fid,'%g\n',[40,40]);
fclose(fid);
I0 = inv(P0);
%
% 100% represents observation at every model grid point. The number of
% observations is set based upon the modelresolution and obs_percent.
%
iobs = obs_percent/100.;
% Observations based on observation densities
switch iobs
 % obs at every grid and at 3 additional points in between two grids
 case 4.0
    obsloc=1:modelresolution*4;          % locations of  obs.
    sig_obs =sig_obs_fac*sig_clim;       % stdev of obs error
    Y = 0.25:0.25:40;
 % obs at every grid and at 1 additional points in between two grids
 case 2.0
    obsloc=1:modelresolution*2;          % locations of  obs.
    sig_obs =sig_obs_fac*sig_clim;       % stdev of obs error
    Y = 0.5:.5:40;
 % obs at every gridpoint
 case 1.0
    obsloc=1:modelresolution;          % locations of  obs.
    sig_obs =sig_obs_fac*sig_clim;     % stdev of obs error
    Y = 1:1:40;
otherwise
  disp('Unknown.');
end % select
%
```

```
% Initialize model and observation error covariance matrix.
nobs=size(obsloc,2);
R = sig_obs^2 * eye(nobs,nobs);% obs. error term
Q = sig_obs^2 * eye(modelresolution,modelresolution);
IQ = inv(Q);
IR = inv(R);
% observations vector holds the observations.
Z=zeros(nobs,1);
% H is the observation operator matrix which will transform from the
% model space into the observation space.
H=zeros(nobs,modelresolution);
% Kalman Gain
K=zeros(modelresolution,nobs);
% Creating the H operator
if (iobs == 4)
  mdim = 1;
  for i = 1:modelresolution-1
     H(mdim,i) = 1;
     H(mdim+1,i) = 0.75; H(mdim+1,i+1) = 0.25;
     H(mdim+2,i) = 0.50; H(mdim+2,i+1) = 0.50;
     H(mdim+3,i) = 0.25; H(mdim+3,i+1) = 0.75;
     mdim = mdim + 4;
  end
  i = modelresolution;
  H(mdim,i) = 1;
  H(mdim+1,i) = 0.75; H(mdim+1,1) = 0.25;
  H(mdim+2,i) = 0.50; H(mdim+2,1) = 0.50;
  H(mdim+3,i) = 0.25; H(mdim+3,1) = 0.75;

elseif (iobs == 2 )
   ndim = 1;
   for mdim=1:nobs
      if (mod(mdim,2) ~= 0)
         H(mdim, ndim) = 1;
         ndim = ndim + 1;
      else
         if (ndim > modelresolution)
            H(mdim, 1) = 0.50;
         else
             H(mdim, ndim) = 0.50;
         end
         H(mdim, ndim-1) = 0.50;
      end
   end
else
```

```
    for iobs=1:nobs
       H(iobs, obsloc(iobs)) = 1;
    end
end
%
% Initialze for data archival
truth_timeseries = ones(modelresolution,assim_time);
Xf_timeseries =ones(modelresolution,assim_time);
If_timeseries=ones(modelresolution,modelresolution,assim_time);
Xhat_timeseries=ones(modelresolution,assim_time);
Ihat_timeseries=ones(modelresolution,modelresolution,assim_time);
Z_timeseries=ones(nobs,assim_time);
%
% Extended Information Filter
tic % starts a stopwatch timer.
eps_obs=R*randn(nobs,1);
for t=1:assim_time
    if t == 1
      Xa0 = X0;
      Ia0 = I0;
    else
      Xa0 = Xa1;
      Ia0 = Ia1;
      TR0 = TR1;
    end

  % Forecast Step

     [TR1]=model_fcst(TR0,timestep,forcing);
     [Xf1]=model_fcst(Xa0,timestep,forcing);
     M   = jacobian_fcst(timestep, Xa0);
     If1 = IQ - IQ*M*inv(Ia0 + M'*IQ*M)*M'*IQ;

%Data assimilation step

     % Generate observations error and observations
     Z=H*TR1;
     Z = Z + eps_obs;
     % Assimilate observations using EIF
     Ia1 = If1 + H' * IR * H; % analysis error covariance matrix
     K = inv(Ia1)*H'*IR;      % Kalman gain
     inov = Z - H*Xf1;        % innovation vector
     Xa1 = Xf1 + K*inov;

  % Save the data
    truth_timeseries(:,t)=TR0(:);
```

```
    If_timeseries(:,:,t)=If1(:,:);
    Xf_timeseries(:,t)=Xf1(:);
    Z_timeseries(:,t)=Z(:);
    Xhat_timeseries(:,t) = Xa1(:);
    Ihat_timeseries(:,:,t)=Ia1(:,:);
  end % end of assim_time
% toc prints the elapsed time since tic was used.
time = toc % returns the elapsed time.
%
% rms error calculation
xf_err = zeros(assim_time,modelresolution);
rms_fcst = zeros(assim_time,1);
xa_err = zeros(assim_time,modelresolution);
rms_an = zeros(assim_time,1);
for i = 1:assim_time
   xf_err(i,:)   = Xf_timeseries(:,i) - truth_timeseries(:,i);
   xf_err2(i,:)  = xf_err(i,:) .* xf_err(i,:);
   rms_fcst(i) = sqrt(sum(xf_err2(i,:))/modelresolution );
   %
   xa_err(i,:)   = Xhat_timeseries(:,i) - truth_timeseries(:,i);
   xa_err2(i,:)  = xa_err(i,:) .* xa_err(i,:);
   rms_an(i) = sqrt(sum(xa_err2(i,:))/modelresolution );
end
%
figure(figid);
plot(truth_timeseries(:,1),'g-','LineWidth',1);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Grid Points'); ylabel('Truth / Observations /Forecast / Analyses');
title(['After ',num2str(1),' time steps of DA'])
hold on
plot(Xf_timeseries(:,1),'b-','LineWidth',1);
axis([1 40 -15 15]);
plot(Y',Z_timeseries(:,1),'k*','LineWidth',1);
plot(Xhat_timeseries(:,1),'r-','LineWidth',1);
hold off
figid = figid + 1;
%
figure(figid);
plot(truth_timeseries(:,assim_time/2),'g-','LineWidth',1);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Grid Points'); ylabel('Truth / Observations /Forecast / Analyses');
title(['After ',num2str(assim_time/2),' time steps of DA'])
hold on
plot(Xf_timeseries(:,assim_time/2),'b-','LineWidth',1);
axis([1 40 -15 15]);
plot(Y',Z_timeseries(:,1),'k*','LineWidth',1);
```

```
plot(Xhat_timeseries(:,assim_time/2),'r-','LineWidth',1);
hold off
figid = figid + 1;
%
figure(figid);
plot(truth_timeseries(:,assim_time),'g-','LineWidth',1);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('Grid Points'); ylabel('Truth / Observations /Forecast / Analyses');
title(['After ',num2str(assim_time),' time steps of DA'])
hold on
plot(Xf_timeseries(:,assim_time),'b-','LineWidth',1);
axis([1 40 -15 15]);
plot(Y',Z_timeseries(:,1),'k*','LineWidth',1);
plot(Xhat_timeseries(:,assim_time),'r-','LineWidth',1);
hold off
figid = figid + 1;
% plot average errors
figure(figid);
xtime= (1:assim_time) * timestep * 5;
plot(xtime',rms_fcst,'b-','LineWidth',1);
title('RMS Error During Data Assimilation')
xlabel('Time in Days'); ylabel('rms error');
hold on;
plot(xtime',rms_an,'r-','LineWidth',1);
hold off;
figid = figid + 1;
% Contour Plots of Phat
figure(figid);
Phat = inv(Ihat_timeseries(:,:,assim_time));
[C,h] = contour(Phat,10);
clabel(C,h);
title('Update Covariance after the last assimilation cycle')
set(h,'ShowText','on','TextStep',get(h,'LevelStep')*2)
colormap cool
figid = figid + 1;
%
figure(figid);
Pf = inv(If_timeseries(:,:,assim_time));
[C,h] = contour(Pf,10);
title('Forecast Covariance before the last assimilation cycle')
set(h,'ShowText','on','TextStep',get(h,'LevelStep')*2)
colormap cool;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function Dm=jacobian( x )
%
% Input:
% x - Lorenz model(a vector)
%
% Return:
% Dm - jacobian of x
%
% File:          jacobian.m
%
% Description:   Compute Jacobian of RHS of L95 system, where x is the
%                argument of the RHS
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Dm=jacobian( x )

n = size(x,1);    % dimension of state space from row
% indices for the model's circular grid points
j=1:n;
jm1=j-1;  jm1=jm1+n*(jm1<1);
jm2=j-2;  jm2=jm2+n*(jm2<1);
jp1=j+1;  jp1=jp1-n*(jp1>n);
%
Dm  = -eye(n,n);
k  = jp1;
mu=j+n*(k-1);
Dm(mu) = Dm(mu) + x(jm1)';
k  = jm2;
mu=j+n*(k-1);
Dm(mu) = Dm(mu) - x(jm1)';
k  = jm1;
mu=j+n*(k-1);
Dm(mu) = Dm(mu) + ( x(jp1) - x(jm2) )';
%
return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [xx]=model_fcst(x0,dt,scheme,F)
%
% Input:
% x0 - Lorenz model initial condition(a vector)
% dt - time step
% scheme - the numerical scheme to be used to forward the model in time
% F  - Forcing parameter
%
% Return:
% xx - Lorenz model forecast after integrating forward in time one time step
%
% File:          model_fcst.m
%
% Description:    Integrate forward Lorenz 95 model by 1 timestep
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xx]=model_fcst(x0,dt,scheme,F)

n=size(x0,1);

switch lower(scheme)
  % -------
  case 'euler'
  % -------
     [xx]=euler(dt,x,F);
  % -------
  case 'rk4'
  % -------
     [xx]=rk4(dt,x,F);
    otherwise
       disp('Unknown scheme.');

end
%
return
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function M =jacobian_fcst(dt,x0)
%
% Input:
% dt - time step
% x0 - Lorenz model(a vector)
%
% Return:
% M - Forecast covariance using euler scheme
%
% File:          jacobian_fcst.m
%
%
% Description:    Forecast covariance matrix using euler sheme
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function M = jacobian_fcst(dt,x0)

n =size(x0,1);    % dimension of state space

M=eye(n,n);       % initialize propagator
%
x  = x0;
A  = jacobian( x );
M = (eye(size(A))  + dt * A) ;
%
return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [x1] = rk4(dt,x0,F)
%
% Input:
% dt - time step
% x0 - Lorenz model(a vector)
% F  - Forcing
%
% Return:
% x1 - Lorenz model after integrating forward in time one time step
%
% File:        rk4.m
%
%
% Description:   Integrate forward Lorenz 95 model by 1 timestep
%                using 4th order Runge-Kutta method
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x1] = rk4(dt,x0,F)

n=size(x0,1);
xa=[x0(n-1); x0(n); x0; x0(1); x0(2) ];

% indices in xa
nm=n-1;
jp1=4:4+nm;
jm1=2:2+nm;
jm2=1:1+nm;
%
% RHS(x0==xa):
% nonlinear advection + linear terms
%
L95= (xa(jp1) - xa(jm2) ) .* xa(jm1) - x0 + F*ones(n,1);
k1  = dt*L95;
%
% RHS(x0 + 0.5*k1)
%
xx=x0 + 0.5*k1;
xa=[xx(n-1); xx(n); xx; xx(1); xx(2) ];
L95= (xa(jp1) - xa(jm2) ) .* xa(jm1) - xx + F*ones(n,1);
k2  = dt*L95;
%
% RHS(x0 + 0.5*k2)
%
```

```
xx=x0 + 0.5*k2;
xa=[xx(n-1); xx(n); xx; xx(1); xx(2) ];
L95= (xa(jp1) - xa(jm2) ) .* xa(jm1) - xx + F*ones(n,1);
k3  = dt*L95;
%
% RHS(x0 + k3)
%
xx=x0 + k3;
xa=[xx(n-1); xx(n); xx; xx(1); xx(2) ];
L95= (xa(jp1) - xa(jm2) ) .* xa(jm1) - xx + F*ones(n,1);
k4  = dt*L95;
%
% linear combination of k1,k2,k3 and k4
%
fac=1./6.;
x1=x0 + fac*( k1 + 2*k2 + 2*k3 + k4 );
%
return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```