University of Reading

School of Mathematics, Meteorology and Physics

Ensemble Data Assimilation: How Many Members Do We Need?

Joanne A. Pocock

August 2009

This dissertation is submitted to the Department of Mathematics and Meteorology in partial fulfilment of the requirements for the degree of Master of Science.

Abstract

Data Assimilation is the incorporation of observational data into a numerical model to produce a model state which accurately describes the observed reality. It is applicable to many dynamical systems as it can provide a complete set of accurate initial conditions for input into a numerical model. One situation where data assimilation is used is numerical weather prediction, where the size of the state is very large (of order 10^7).

In this thesis we consider a data assimilation scheme known as the particle filter. The particle filter uses information on the probability density of the model to forecast an estimate of an ensemble of model forecasts, the state and its uncertainty. Unfortunately the number of ensemble members required to accurately forecast the state grows exponentially as the size of the state increases. For this reason the idea of mode tracking is introduced. When mode tracking, the state is split into two sections. One section is forecast using the particle filter, the other is treated so its values are equal to the mode of the marginal pdf. Previous work has shown that introducing mode tracking to the particle filter reduces the number of particles required to achieve a given filter accuracy. This hypothesis is tested by solving the stochastic Lorenz equations using both the particle filter and particle filter with mode tracking with varying numbers of particles. For both the particle filter and particle filter with mode tracking it is found that increasing the number of particles increases the accuracy of the the solution obtained. However contrary to previous work it is found that for small ensemble sizes the particle filter can provide more accurate results than the particle filter with mode tracking. The results appear to be sensitive to the specification of the observation and model errors.

Contents

List of Figures

List of Tables

List of Symbols

Latin

a (superscript)	Analysis
b (superscript)	Background
B	Noise Parameter
d	Observation
F	Constant matrix
F (superscript)	Forecast value
G	Constant
G_g	Constant
H	Observation operator
i (superscript)	Particle
J	Cost function
J_o	Observatoin term of the cost function
J_q	Background term of the cost function
K	Gain matrix
N	Number of Particles
n	Size of state vector
o (superscript)	Observation
p	Size of the observation vector
Q	Model error covariance matrix

R	Observation error covariance matrix
t (subscript)	Time
t	Time
T (superscript)	Truth
w	Weight
$\tilde{w}$	Ordered weight
W	Value of Wiener process
x	Lorenz Variable
X	Stochastic Lorenz variable
$\bar{x}$	Particle mean for x dimension
y	Lorenz Variable
Y	Stochastic Lorenz variable
$\bar{y}$	Particle mean for y dimension
z	Lorenz Variable
Z	Stochastic Lorenz variable
$\bar{z}$	Particle mean for z dimension

Greek

α	Order of weak convergence
β	Lorenz parameter
Δ	Timestep size
Δ (superscript)	Timestep size
ϵ	Error
ε_t	Observation error
γ	Order of strong convergence
η	Model error

λ	Complex number
ρ	Lorenz parameter
σ	Lorenz parameter
ψ	State vector
$\mathbf{\Psi}$	State vector
$\tilde{\Psi}$	Numerical approximaion
Ψ_r	Part of the state to be mode tracked
Ψ_s	Part of the state not to be mode tracked
ω	Normalised weight

List of Abbreviations

DA	Data Assimilation
EM	Euler-Maruyama
EnKF	Ensemble Kalman Filter
KF	Kalman Filter
LHS	Left hand side
MT	Mode Tracking
NWP	Numerical Weather Prediction
pdf	Probability density function
RHS	Right hand side
PF	Particle Filter
PFMT	Particle Filter with Mode Tracking
RH	Rank Histogram
RMSE	Root Mean Squared Error
SIS	Sequential Importance Sampling

Chapter 1

Introduction

1.1 Background

Data Assimilation (DA) is the incorporation of observational data into a numerical model to produce a model state which accurately describes the observed reality [Kalnay, 2002]. It is applicable to many situations as it provides a complete set of accurate initial conditions for input into a numerical model.

One situation where DA is used is numerical weather prediction (NWP) [UKMO, 2009]. To create an accurate weather forecast a numerical model must be run from accurate initial conditions. We have some information in the form of observations taken by satellites, radiosondes and weather stations etc., however these observations may contain measurement errors. There are also some areas of the globe where observations cannot be made. DA combines a previous weather forecast along with these incomplete observations to provide a complete and accurate set of data. These are initial conditions from which the forecast is run.

Unfortunately the exact state of the atmosphere cannot be determined, and just small perturbations in these initial conditions can cause large differences in the forecast. We must take into account the uncertainty in the initial conditions. One approach is to use a number of model realisations with different initial conditions known as ensembles or set of particles. These particles are sampled from a probability density function (pdf) that is associated with

the most likely initial conditions. Many ensemble DA methods only allow this pdf to be Gaussian, however often real data is often non-linear and non-Gaussian, so we require an ensemble DA scheme that allows us to have a non Gaussian pdf. A group of schemes that allow this are known as particle filters (PF) [van Leeuwen, 2009], these are described in detail in section 2.3.

A PF is an assimilation scheme that takes a sample of points from an a priori pdf of the state. Each of these points in phase space correspond to a set of model initial conditions. These points are then forecast forward in time. These forecast points provide us with a new prediction pdf at this new time. This prediction pdf is combined with the pdf of an appropriate observation to give us a pdf of the system at the new time. Each particle is given a weight according to how close it is to the observations. Particles close to the observations are given higher weights than those that are further away from the observations. This weighting, discussed in section 2.3.2.2, allows us to allocate importance onto the particles we are most interested in [Doucet et al., 2000a].

Unfortunately this weighting of particles causes us some problems. As we move forward in time, most of the weight goes onto one particle [Snyder et al., 2008]. This means our statistical information is to small to be meaningful. This situation is known as filter degeneracy [van Leeuwen, 2009]. To overcome this situation we must resample our data at each stage. When we resample our data we make multiple copies of particles with high weights and discard some of the particles with the lowest weights. This allows us to redistribute the weights on the particles and still retain the important information, this is discussed in section 2.3.4. This resampling is the final part of the PF scheme. For low dimensional systems the PF works well. However for the NWP problem our system is large, this makes it difficult to solve the problem. A large number of particles are required to gain satisfactory results and this makes the PF scheme computationally costly.

To try and make the computation less costly the idea of mode tracking (MT) is introduced [Vaswani, 2008]. It is thought that combining MT with the PF allows us to obtain comparable results to the PF but with fewer numbers of particles. In MT, the state space is split in to two subspaces. One of these is forecast using the PF and the other is forecast using MT. The

MT is combined with the PF to produce a ensemble DA scheme that allows the accuracy of the PF but is less computationally costly [Vaswani, 2008].

1.2 Aims

The aims of this dissertation are:

- To investigate the particle filter (PF) and the associated problem of ensemble size.
- To investigate the use of mode tracking (MT) with the particle filter (PFMT) and consider whether this can help to reduce the number of particles required.

The aims are achieved by implementing the PF with a simple model, carrying out experiments with varying numbers of particles and comparing the results. The same model is implemented in the PFMT and experiments are carried out to show how use to MT effectively. This includes determining the best way to split the state, and filtering varying numbers of particles. The results of the PF and PFMT are compared to show if MT can help reduce the number of particles required.

1.3 Principal Results

The principal results from this thesis are:

- The PF produced very poor results when too few particles are used. The mean of the pdf lies far from the truth and there is not enough variability in the particles. As the number of particles is increased the mean of the pdf becomes much closer to the truth and the ensemble variability is improved (see Chapter 5).
- The best results from the PFMT are seen when both the x and y dimensions are mode tracked. MT only the z dimension produces the worst results. As with the PF the PFMT the accuracy of the solution increases as the number of particles is increased, however the variability properties of the PFMT are worse than the PF (see Chapter 6).

- The most surprising result is that for small numbers of particles ($N = 5$ and $N = 50$) the PF performs better than the PFMT. It is only when the number of particles is increased to $N = 500$ that the results from the PFMT are usually better than the PF results. This performance of the PFMT appears to be rather sensitive to the choice of observation and model noise (see Chapter 6).

1.4 Outline

We begin this thesis by reviewing previous work that considers DA and PF. This previous work and a more detailed investigation in to how PFs work is considered in Chapter 2. In this chapter we also consider the idea of MT introduced by Vaswani [2008]. In Chapter 3 the model system used for experiments in this thesis is discussed. The initial results from the numerical scheme are also shown. Chapter 4 describes our implementation of the PF and PFMT code. Some of the diagnostics used to asses the PF and PFMT results are also discussed. Chapter 5 shows some results from the PF code. The PFMT results are shown and discussed in Chapter 6. The PF and PFMT results are also compared. In Chapter 7 we conclude the dissertation by summarising the previous chapters and suggesting further work.

Chapter 2

Data Assimilation

In this chapter previous work on the mathematics of data assimilation (DA) is considered. We start by considering conventional DA [Kalnay, 2002] then ensemble DA. After this we discuss the idea of particle filters (PF) [van Leeuwen, 2009][Doucet et al., 2000a] and some of the problems associated with them [Snyder et al., 2008]. Finally we consider how the introduction of mode tracking (MT) [Vaswani, 2008] into the PF can overcome some of the problems encountered.

2.1 Data Assimilation

We first consider the linear dynamical system,

$$\psi_{t+1} = F\psi_t + \eta_t, \tag{2.1}$$

where ψ_t is the state vector of length n at time t, F is a constant matrix of size $n \times n$ and η_t is the model noise at time t. We assume we have observations d_t at time t, these observations are related to the state of the system by,

$$d_t = H\psi_t + \varepsilon_t, \tag{2.2}$$

where d_t is the observation vector of length p at time t, ε_t is the observation error (a vector of size p) at time t and H is known as the observation operator, a matrix of size $p \times n$. The

observation operator maps model space to observation space. In the simple case of direct observations considered here, it picks out the components of the ψ vector that corresponds to the the observations we have.

Next we consider a qualitative description of the DA process. A schematic picture is seen in Figure 2.1.

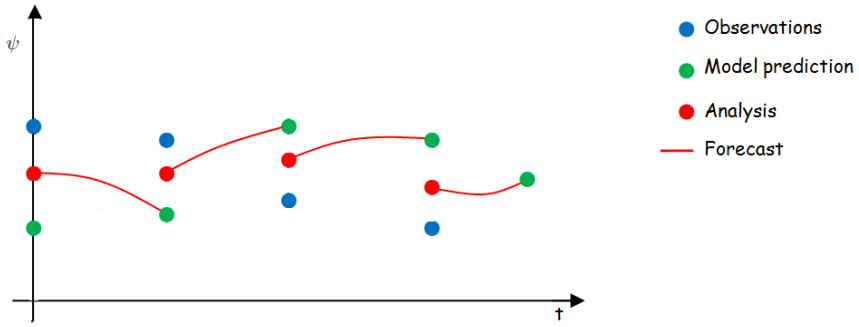

Figure 2.1: A schematic of the sequential DA scheme

Figure 2.1 shows that DA works by taking d_t, an observation at time t and a model prediction, ψ_t^b, known as the background. ψ_t^b and d_t are combined to produce an analysis ψ_t^a, this value is then forecast to the next timestep using equation 2.1. This forecast value becomes the new background state. The process of combining observations and backgrounds to create an analysis which is then forecast is repeated until the time at at which the model state is required.

The Data Assimilation Algorithm [Swinbank et al., 2003]
1. Calculate the analysis: $$\psi_{t+1}^a = \psi_{t+1}^b + K(d_{t+1} - H\psi_{t+1}^b),$$ where K (of size $n \times p$) is known as the gain matrix, and is prescribed by the particular assimilation scheme in use. 2. Given the dynamical system in 2.1 forecast the analysis using $$\psi_{t+1}^b = F\psi_t^a + \eta_t$$ to obtain a background state at the next time.

Table 2.1: A simple DA algorithm.

The equations of the assimilation scheme can also summarised in a simple algorithm [Swinbank et al., 2003], this is seen in Table 2.1.

Although a fairly simple idea DA can be a very computationally costly exercise, this is due to the large state spaces that are often dealt with. For NWP the size of the ψ^a and ψ^b vectors is of the order 10^7, with the d vector being of size 10^6 [Nichols, 2003].

2.2 Ensemble Data Assimilation

So far only one initial state has been assimilated. Unfortunately the exact state of the atmosphere often cannot be determined and as small perturbations from the set of conditions can lead to a large change in the outcome of the forecast it is important that uncertainty in our initial conditions is taken into account. To do this a number of initial states are considered and each one is assimilated. Each of these different states is known as an ensemble or particle. Figure 2.2 shows a number of these ensembles being forecast from the initial probability density function (pdf) to give a pdf at the forecast time.

This forecast pdf has uses throughout DA and we would like to have some idea of it at each

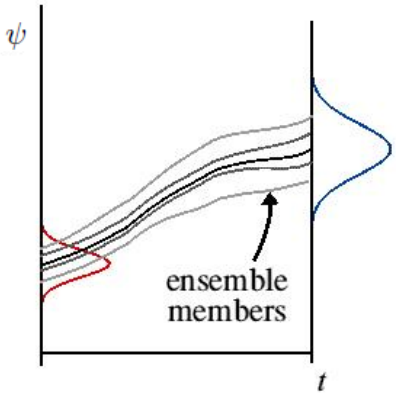

Figure 2.2: Schematic of ensemble DA: Each ensemble is forecast from slightly different initial conditions. The red line represents the initial pdf. The grey lines are the trajectories of the ensembles sampled from the initial pdf, the black line is the ensemble mean trajectory. The blue line represents the forecast pdf. Figure from Bannister [2009]

time step. An assimilation method known as the Kalman filter (KF) [Kalnay, 2002] keeps the pdf information during the assimilation. At each timestep the pdf of the analysis is forecast as well as the analysis. The ensemble Kalman filter (EnKF) [Evensen, 2006] assimilates an ensemble of states and these forecasts are used to determine the background pdf at the new time. However both the KF and EnKF assume that the data is modelled by a Gaussian state space model. Most real data is far more complex. Often real data is non-linear, non-Gaussian and in many dimensions, and hence the assumptions in KF and EnKF are inappropriate. A sequential Monte Carlo method called particle filtering allows us to approximate non-linear, non-Gaussian state space models [Doucet et al., 2000a].

2.3 The Particle Filter

To gain an understanding of how a PF works a schematic diagram for one iteration of the process is considered, this is seen in Figure 2.3. Here the vertical axis represents a model variable, whereas the horizontal axis represents time.

Figure 2.3 (a) shows the prior pdf $p(\psi_t)$ the probability of the state ψ at time t, this is represented by the red line. In Figure 2.3 (b) the black dots represent a set of N particles sampled from this pdf. Figure 2.3 (c) shows these particles being forecast to time $t+1$, their trajectories are shown by the black lines. At this time (Figure 2.3 (d)) there is an observation d, shown as a blue dot, which has an associated pdf $p(d_{t+1}|\psi_{t+1})$ the probability of the observation given the state at time $t+1$. This is known as the observation likelihood and is represented by a blue line. Finally in Figure 2.3 (e) the forecast particles and $p(d_{t+1}|\psi_{t+1})$ are combined to give, $p(\psi_{t+1}|d_{t+1})$, the probability of the model state given the observation at at time $t+1$. This is known as the filtering pdf and is represented by the red line at this time.

To continue the iteration we return to Figure 2.3 (c) forecasting the particles from the new model pdf. The PF algorithm is cycled until the time is reached when we require the information.

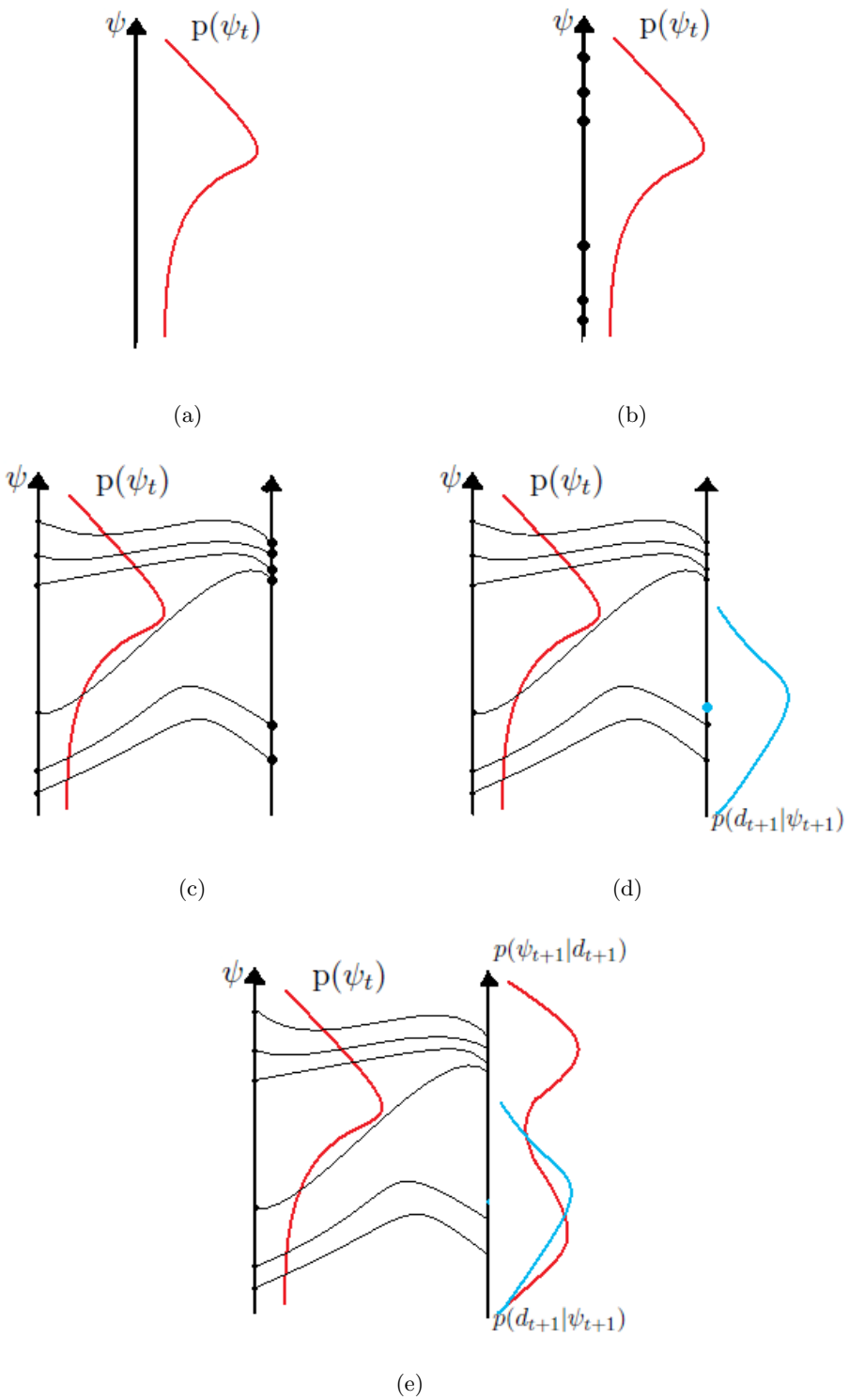

Figure 2.3: A schematic of one iteration of the particle filter process. See text for details.

Now we have seen qualitatively how a PF works we describe the equations used in the algorithm, following van Leeuwen [2009].

2.3.1 The Prior and Sampling From It

To start the PF we require a prior pdf, $p(\psi_0)$. Here ψ_0 is the state vector of length n at the initial time. Due to the vast amount of data required for situations such as NWP it is difficult to store all the information associated with $p(\psi_t)$. There are many values associated with this pdf including is mean, variance and other moments. Both van Leeuwen [2009] and Doucet et al. [2000a] consider the mean of the pdf to simplify the handling of $p(\psi)$. The mean of a function, f, of the state is given by,

$$\overline{f(\psi)} = \int f(\psi)p(\psi)d\psi. \tag{2.3}$$

However this quantity may still be difficult to calculate due to the large size of the state. To overcome this we sample from the initial pdf to gain a set of N particles, $\{\psi^i\}_{i=1}^N \sim p(\psi_t)$. This allows us to represent $p(\psi)$ as

$$p(\psi) = \frac{1}{N}\sum_{i=1}^{N}\delta(\psi - \psi^i), \tag{2.4}$$

. where δ is the Dirac delta function. The mean can now be represented by

$$\overline{f(\psi)} \approx \overline{f_N(\psi)} = \frac{1}{N}\sum_{i=1}^{N} f(\psi^i). \tag{2.5}$$

2.3.2 Sequential Importance Sampling

Now the prior pdf has been considered and sample of particles has been taken from it we must consider how to deal with these particles. The second stage in the PF algorithm is known as sequential importance sampling (SIS). SIS is the method of forecasting these particles, weighting them according to their position relative to the observations and then normalising these weights. The SIS algorithm is summarised in Table 2.2. We define the notation seen in Table 2.2, and later in Table 2.4, as follows. $\psi_{0:k} = \{\psi_0, ..., \psi_t\}$ represents the model states up to time t. $d_{0:k} = \{d_0, ..., d_t\}$ represents the observations up to time t. $\pi(\psi_{0:k}|d_{1,k})$,

The Sequential Importance Sampling Algorithm [Doucet et al., 2000b]

For times $k = 0, 1, 2, ...$

1. For $i = 1, ..., N$, sample $\psi_k^i \sim \pi(\psi_k|\psi_{0:k-1}^i, d_{0:k})$ and $\psi_{0:k^i} = (\psi_{0:k-1}^i, \psi_k^i)$.
2. For $i = 1, ..., N$, evaluate the importance weights up to a normalising constant:

$$w_k^i = w_{k-1}^i \frac{p(d_k|\psi_k^i)p(\psi_k^i|\psi_{k-1}^i)}{\pi(\psi_k|\psi_{0:k-1}^i, d_{0:k})} \tag{2.6}$$

3. For $i = 1, ..., N$, normalise the importance weights:

$$\omega_k^i = \frac{w_k^i}{\sum_{j=1}^N w_k^j} \tag{2.7}$$

Table 2.2: The sequential importance sampling algorithm.

the distribution of the model states up to time k given the observations up to time d, is an arbitrary importance sampling distribution and is known as the importance function. This follows the notation from Doucet et al. [2000a]. The steps of the algorithm are described in sections 2.3.2.1 and 2.3.2.2.

2.3.2.1 The Forecast Step and Bayes Theorem

Step one in the SIS algorithm (Table 2.2) is the forecast step. In the probabilistic language this forecast step is denoted as: sample from the importance function $\psi_k^i \sim \pi(\psi_k|\psi_{0:k-1}^i, d_{0:k})$ the distribution of the state at time k given the states up to time $k-1$ and the observations up to time k . In many PFs the importance function $\pi(\psi_{0:t}|d_{1:t})$ is chosen to be the priori distribution. We shall see in section 2.3.5 that we adopt the prior distribution as the importance function. For this, the forecast step is given as: Sample from $p(\psi_t|\psi_{t-1}^i)$. In practice, for each particle ψ_t^i, we run the stochastic model forward from time $t-1$ to time t using the full nonlinear stochastic model equations.

Once we have forecast these particles we wish to combine them with the observations at

the forecast time. To do this we make use of Bayes Theorem.

Theorem 1 Bayes Theorem

$$p(\psi|d) = \frac{p(d|\psi)p(\psi)}{p(d)} \tag{2.8}$$

So if ψ is the model state and d the observations then Bayes Theorem tells us that the probability of the model state given the observations is equal to the probability of the observations given the model state multiplied by the probability of the model state divided by the probability of the observation. Here $p(d)$ is a normalisation factor.

If the importance function is chosen to be the prior distribution, then by substituting the summation notation in equation (2.4) into Bayes Theorem we obtain equation 2.9,

$$p(\psi_t|d_t) = \sum_{i=1}^{N} w^i \delta(\psi_t - \psi_t^i), \tag{2.9}$$

where w^i is the particle weight. We can now calculate the probability of the model state given the observations. The particle weights, w^i, can be used to give us information on how close each of the particles is to the observation, this is discussed in the following section.

2.3.2.2 Weighting

We are interested in knowing how close each particle, ψ_t^i, to the observation at this time. For this reason the w_t^i from equation (2.9) is known as the weight at time t of particle ψ_t^i.

$$w_t^i = \frac{p(d_t|\psi_t^i)}{\sum_{j=1}^{N} p(d_t|\psi_t^j)}, \tag{2.10}$$

However we must remember that these weights are only for when the importance function is equal to the prior pdf. Weights for a general importance function are shown in equation (2.6) in Table 2.2.

This weighting of the particles gives the relative importance of each particle in the PDF. It is important as we may find that some of the forecast values do not fit with the observations. As weighting represents the importance of the particles, particles that fall in areas of high probability, i.e. near an observation, are given a weighting to make them more important

where as particles that are in areas of low probability are given a low weighting. We consider an example from Fox [2003] to see how weighting works.

Example 1 Weighting Example

A robot is somewhere on the map seen in Figure 2.4 and he is trying to find his location. Doors are shown on the map and the robot can recognise doors but he cannot tell the difference

Figure 2.4: Map of the corridor used for the robot experiment. The white areas represent the corridors where the robot can walk. Figure from Fox [2003].

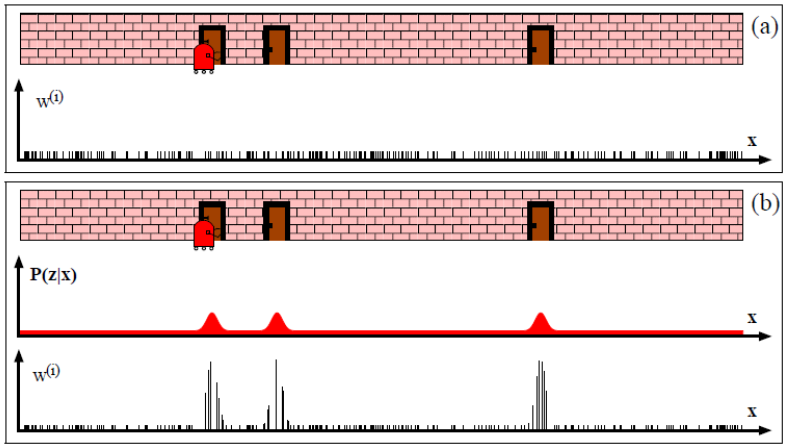

Figure 2.5: Figures showing the position of the robot, graphs of the associated probability of his position and graphs of the particles representing the possible location of the robot. The top of both plots (a) and (b) show the location of the robot relative to the doors. The x vs w plots in (a) and (b) show the distribution of particles across the location seen and their relative weights. Here we use the notation as in Fox [2003], w^i represents thew weight of particle i. Here x is the equivalent of ψ, the state and z is the observation. The remaining plot in (b) shows the observation likelihood $p(d|\psi)$. Figure from Fox [2003]

between each door. The first step in in the particle filter is to sample a number of points where the robot may be. These particles initially all have equal weights and are represented in Figure 2.5 (a) by the small black lines on the x vs. w plot. The robot then arrives at a door, he now knows that on the map he is somewhere in front of a door, but he does not know which door. There is equal chance of him being at each door, this is shown in the x vs. $p(z|x)$ plot. Now we have some information of the robots location we are able to weight the particles accordingly. The particles that represent being by a door are weighted higher as there is a greater chance that this is where the robot is. The particles not near a door are given lower weights as there is a smaller chance that this is where the robot is. We can see these weights represented on the x vs. w plot in Figure 2.5 (b). We shall return to this example later in section 2.3.4 to find out where the robot is.

2.3.3 Problems with Weighting

Unfortunately this weighting of particles causes us some problems, as we move forward in time weight goes onto one particle [Snyder et al., 2008]. This happens because each weight is multiplicatively dependent on a previous weight (equation 2.6). Eventually this leads to all the weight being on one particle. In this case, our statistical information is to small to be meaningful, this situation is known as filter degeneracy [van Leeuwen, 2009]. Snyder et al. [2008] investigated this weighting problem. They calculated the maximum normalised weight at each timestep. This maximum weight is defined as

$$\omega_{max,t} = \max\{\omega_t^i : i = 1, .., N\}. \tag{2.11}$$

Figure 2.6 shows the results of an experiment that was run to see how the weights behave. Is shows a histogram of the number of occurrences of the maximum weights in each interval. It was created by tallying how many times the maximum weight falls in each interval. We see that many of the maximum weights fall into the interval $(0.98, 1)$, showing that on each of these occurrences nearly all the weight is on one particle. This means the information from our other particles is lost. Results included in Appendix A show that the code described in section 4.1 produces similar results to those in Snyder et al. [2008] when the resamiling step

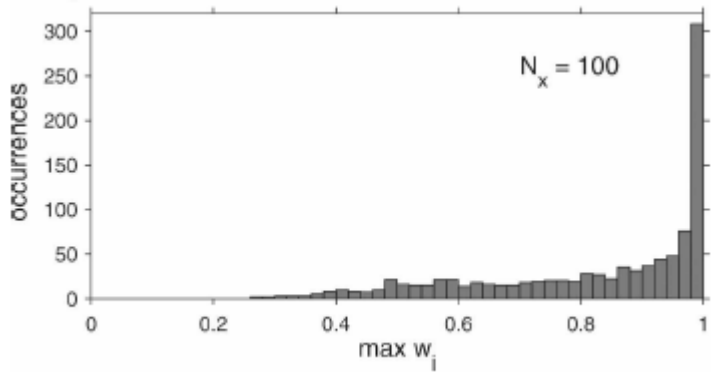

Figure 2.6: A plot of the number of occurrences of the maximum weights in particular intervals. Figure from Snyder et al. [2008]

is omitted.

Now we have seen that weighting the particles can lead to filter degeneracy we must find a way to over come it. One way to do this is to resample the data.

2.3.4 Resampling

We have seen that if we keep all our particles and their weights and continue to evolve them over time all the weight goes onto one particle. To avoid this situation we include a resampling step in our sequential importance sampling algorithm. Essentially resampling the data means that we discard particles with lower weights as they are furthest from the observations. We increase our sample size back to N particles by creating multiple copies of the particles with high weights. When multiple copies of each particle are forecast, due to the stochastic nature of the forecast step, we obtain different forecast results. Although we obtain different forecasts from multiple copies of the same particle, these forecasts are still closer than if the particles had not been resampled. This leads to a loss of diversity in the particles. The effectiveness of the resampling can be seen in the continuation of the Robot Example [Fox, 2003].

Example 2 Resampling Example

Last time we saw the robot, Figure 2.5 (b), he had reached a door and the particles had been weighted. The x vs w plot on Figure 2.7 (c) shows these particles once they have been resampled, normalised and the robot has moved to the right. We see that the resampling

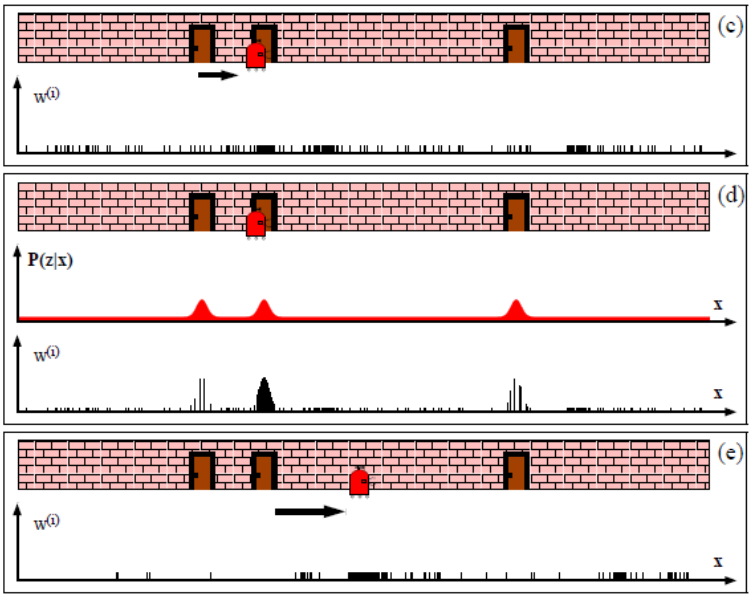

Figure 2.7: Figures showing the position of the robot, graphs of the associated probability of his position and graphs of the particles representing the posible location of the robot. Notation and plots as in Figure 2.5. Figure from Fox [2003]

has increased the number of particles where the weights had been increased and decreased the number of particles where the weights were small. Figure 2.7 (d) shows the robot reaching another door, again the particles are weighted accordingly. It is already easy to see that the area with a high density of high weighted particles represents the actual position of the robot. Figure 2.7 (e) shows the resampled particles with normalised weights once the robot has moved right. It is possible to see that the particles are most densely populated around the position of the robot with only a few particles else where.

Figure 2.8 shows the results of the experiment carried out in [Fox, 2003]. Figure 2.8 (a) shows the results once the robot has moved 5m, we see that the particles are spread across the whole map giving little indication of the robots actual location. By Figure 2.8 (b) the robot has moved slightly further, the particles are now in clusters in four different locations, this limits the position of the robot to these four corners on the map. Finally by Figure 2.8 (c) after the robot has moved approximately 55m we see that all the particles are in one cluster suggesting that the robot is located somewhere amongst them.

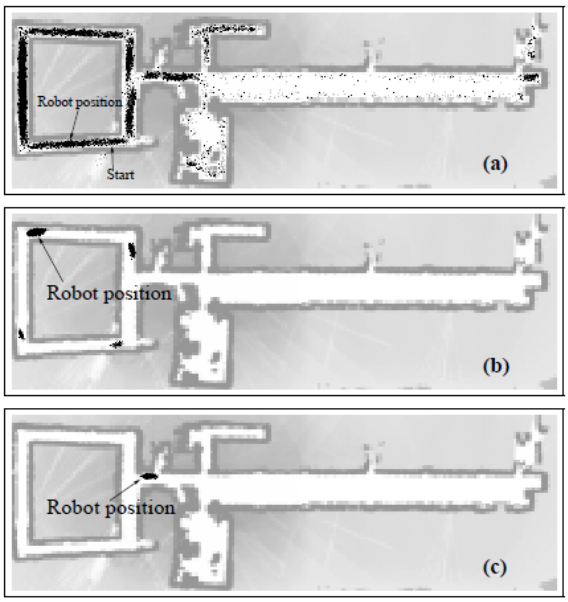

Figure 2.8: Maps showing the possible location of the robot after he has moved 5m (a), more than 5m (b), 55(m). Figure from Fox [2003]

2.3.4.1 Stratified Resampling

There are many ways to resample including Multinomial resampling, Residual resampling, Stratified resampling and Systematic resampling. Douc et al. [2005] show that residual, stratified and systematic resampling in general produce comparable results that are better than those produced by multinomial resampling. Systematic resampling is most simple to implement, but there is a lack of theoretical analysis of its behavior and for this reason the method implemented in this thesis is Stratified resampling. The Stratified resampling algorithm seen in Kitagawa [1996] is presented in Table 2.3.

The Stratified Resampling Algorithm [Kitagawa, 1996]
1. Rearrange the normalised weights $\omega^1, \ldots, \omega^N$ in ascending order. The reordered weights are expressed as $\tilde{\omega}^1, \ldots, \tilde{\omega}^N$.
2. Split the interval in to N equal sections of length $\frac{1}{N}$. Label sections j where $j = 1, 2, ..., N$.
3. Calculate values u_j, where u_j is the midpoint of section j.
4. Set $i = 1$, $j = 1$, $\omega^L = 0$ and $\omega^R = \tilde{\omega}_1$.
5. Test $\omega^L < u_j \leq \omega^R$
If this inequality holds then accept the particle associated with $\tilde{\omega}^i$ and set $j = j + 1$.
If the inequality does not hold set $\omega^L = \sum_{l=1}^{i} \tilde{\omega}^l$, $\omega^L = \sum_{l=1}^{i+1} \tilde{\omega}^l$ and $i = i + 1$.
6. Repeat step 5. until $i = N$ and $j = N$
7. There should now be N particles accepted, some of which are duplicates. These are the resampled particles.

Table 2.3: The Stratified Resampling Algorithm.

We now consider a simple example to explain how stratified resampling works.

Example 3 Stratified Resampling Example

Assume we need to resample five particles with normalised weights $\omega^1 = 0.3$, $\omega^2 = 0.1$, $\omega^3 = 0.05$, $\omega^4 = 0.35$ *and* $\omega^5 = 0.2$ *respectively.*

1. *Sort into order of increasing value to obtain* $\tilde{w}^1 = w^3 = 0.05$, $\tilde{w}^2 = w^2 = 0.1$, $\tilde{w}^3 = w^5 = 0.2$, $\tilde{w}^4 = \omega^1 = 0.3$ *and* $\tilde{\omega}^5 = \omega^4 = 0.35$.

2. *Split the interval* $(0, 1]$ *in to* 5 *equal sections of length* 0.2.

3. *Take* u_j *to be the midpoint of each interval* j. $u_1 = 0.1$, $u_2 = 0.3$, $u_3 = 0.5$, $u_4 = 0.7$ *and* $u_5 = 0.9$.

4. *Now we consider the inequalities containing* u_j *and* w.

Test $\omega^L < u_j \leq \omega^R$ *where* $u_1 = 0.1$, $\omega^L = 0$ *and* $\omega^R = 0.05$.

$0 < 0.1 \leq 0.05 = \tilde{w}^1$

This inequality does not hold so we add $\tilde{\omega}^1$ *to the LHS and* $\tilde{\omega}^2$ *to the RHS.*

Test $\omega^L < u_j \leq \omega^R$ *where* $u_1 = 0.1$, $\omega^L = 0.05$ *and* $\omega^R = 0.15$.

$\tilde{\omega}^1 = 0.05 < 0.1 \leq 0.15 = \tilde{\omega}^1 + \tilde{\omega}^2$

This is true so take the particle associated with $\tilde{\omega}^2$ *and increase the value of* j *to* $j = 2$.

Test $\omega^L < u_j \leq \omega^R$ *where* $u_1 = 0.3$, $\omega^L = 0.05$ *and* $\omega^R = 0.15$.

$\tilde{\omega}^1 = 0.05 < 0.3 \leq 0.15 = \tilde{\omega}^1 + \tilde{\omega}^2$

This does not hold so we add the next weight to each side.

Test $\omega^L < u_j \leq \omega^R$ *where* $u_1 = 0.3$, $\omega^L = 0.15$ *and* $\omega^R = 0.35$.

$\tilde{\omega}^1 + \tilde{\omega}^2 = 0.15 < 0.3 \leq 0.35 = \tilde{\omega}^1 + \tilde{\omega}^2 + \tilde{\omega}^3$

This is true so take the particle associated with $\tilde{\omega}^3$ *and increase the value of* j *to* $j = 3$.

Test $\omega^L < u_j \leq \omega^R$ *where* $u_1 = 0.5$, $\omega^L = 0.15$ *and* $\omega^R = 0.35$.

$0.15 < 0.5 \leq 0.35$

This does not hold so we add the next weight to each side.

Test $\omega^L < u_j \leq \omega^R$ *where* $u_1 = 0.5$, $\omega^L = 0.35$ *and* $\omega^R = 0.65$.

$0.35 < 0.5 \leq 0.65$

This is true so take the particle associated with $\tilde{\omega}^4$ *and increase the value of* j *to* $j = 4$.

Test $\omega^L < u_j \leq \omega^R$ *where* $u_1 = 0.7$, $\omega^L = 0.35$ *and* $\omega^R = 0.65$.

$0.35 < 0.7 \leq 0.65$

This does not hold so we add the next weight to each side.

Test $\omega^L < u_j \leq \omega^R$ *where* $u_1 = 0.7$, $\omega^L = 0.65$ *and* $\omega^R = 1$.

$0.65 < 0.7 \leq 1$

This is true so take the particle associated with $\tilde{\omega}^5$ *and increase the value of* j *to* $j = 5$.

Test $\omega^L < u_j \leq \omega^R$ *where* $u_1 = 0.9$, $\omega^L = 0.65$ *and* $\omega^R = 1$.

$0.65 < 0.9 \leq 1$

This is true so take the particle associated with $\tilde{\omega}^5$. The value of j can no longer be increased so the algorithm is finished.

The new particles are those associated with the original ω^1, ω^2, ω^4, ω^4 and ω^5. We have lost the particle with the lowest weight, but have two copies of the particle with the highest weight.

This resampling is the final step in the PF algorithm. From this point we return to the forecast step, we forecast the resampled particles, weight them and then resample. This process is repeated until the final time is reached.

2.3.5 Summarising the Particle Filter

The algorithm containing this resampling step is known as the Bootstrap Filter [Doucet et al., 2000a], this is shown in Table 2.4.

The Bootstrap Filter
1. Initialisation, t = 0
For $i = 1, \ldots, N$, sample ψ_0^i $p(\psi_0)$ and set $t = 1$.
2. Importance Sampling Step
For $i = 1, \ldots, N$, sample $\widetilde{\psi}_t^i$ $p(\psi_t \vert \psi_{t-1}^i)$ and set
$\widetilde{\psi}_{0:t}^i = (\widetilde{\psi}_{0:t-1}^i, \widetilde{\psi}_t^i)$.
For $i = 1, \ldots, N$, evaluate the importance weights,
$\widetilde{w}_t^i = p(d_t \vert \widetilde{\psi}_t^i)$.
3. Normalise the importance weights.
4. Selection Step
Resample with replacement N particles $(\psi_{0:t}^i; i = 1, \ldots, N)$
from the set $(\widetilde{\psi}_{0:t}^i; i = 1, \ldots, N)$ according to the
importance weights.
Reset weights to $\frac{1}{N}$.
Set $t = t + 1$ and repeat step 2.

Table 2.4: The Bootstrap Filter

The algorithm in Table 2.4 implies that we must resample at every step, however this is not the case. It is possible to resample only at timesteps when the weights differ significantly in value. This will save computational cost as at times when the weights are nearly equal it is possible just to keep each of the particles as this is likely to be what the particle filter would do. The only difficulty with not resampling at each timestep is deciding when resampling should take place. This may be done when the highest weight reaches a particular value or at say every other timestep. As this choice is complex to make and code we chose to resample at every timestep. We see in section 6.1 that this resampling at every timestep leads to a lack of variability in the particles. Resampling less often may be a way to overcome this problem.

2.3.6 Properties of the Particle Filter

Resampling the data removes the problem of all the weight ending up on one particle, and for small scale systems the PF now works well. It is expected that the PF will converge to the true distribution in the limit of large particle samples. However for the NWP problem our system is large, this leads to some more complications. Snyder et al. [2008] shows us that the main issue is the number of particles we require. As the dimension of the state increases the number of particles required for the PF to converge to the true distribution increases exponentially. This is shown in Figure 2.3.6. This large number of particles required makes the PF computationally expensive.

2.4 The Particle Filter with Mode Tracking

To reduce the computational cost of the PF Vaswani [2008] introduced the idea of mode tracking (MT) . It is thought that combining MT with the PF allows us to obtain comparable results to the PF but with fewer numbers of particles. When MT we split the state into $\Psi = [\Psi_s, \Psi_r]$. We treat the Ψ_s part of the state using the ordinary PF, with the pdf of the state being approximated. The Ψ_r part is treated differently. We wish to set Ψ_r equal to the mode of the marginal pdf. We do this by forecasting Ψ_r one timestep with no noise. We then find optimum values for Ψ_r by minimising a cost function that can be related to the 3D-Var

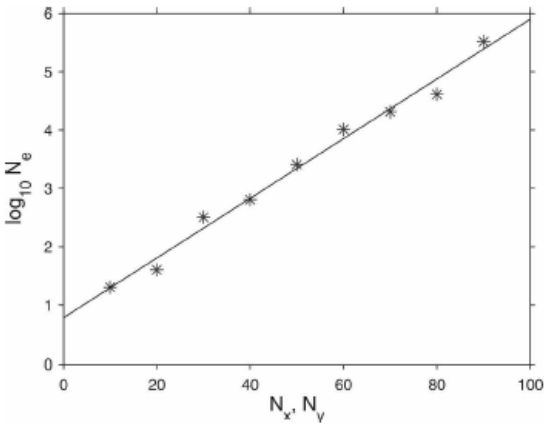

Figure 2.9: Number of Particles Required required 'for the posterior mean estimated by the particle filter is to have averaged square error less than the prior or observations'[Snyder et al., 2008]. As the dimension of the state increases (along the horizontal axis) we see that the number of particles required increases exponentially. N_x and N_y are the state dimensions, N_e is the number of ensembles required. The asterisks represent the the simulation results, a best fit line is then plotted. Figure from Snyder et al. [2008]

[Bannister, 2009].

We present, in Table 2.5 the PFMT algorithm from Vaswani [2008]. We shall consider each stage of the algorithm to see how PFMT works in detail.

The PFMT Algorithm
1. Importance Sample $\Psi_{t,s}$: $\forall i$ sample $\Psi^i_{t,s}$ $p(\Psi^i_{t,s}\|\Psi^i_{t-1})$.
2. Mode track $\Psi_{t,r}$: $\forall i$ set $\Psi_{t,r} = m^i_t$ where
$m^i_t(\Psi^i_{t-1}, \Psi^i_{t,s}, d_t) = \arg\min_{\Psi_{t,r}}[-\log p(d_t\|\Psi^i_{t,s})p(\Psi_{t,r}\|\Psi^i_{t-1}, \Psi^i_{t,s})]$.
3. Weight: $\forall i$ compute $w^i_t = \frac{\omega^i_t}{\sum_{j=1}^N \omega^j_t}$ where $\Psi^i_t = [\Psi^i_{t,s}, \Psi^i_{t,r}]$.
and $\omega^i_t = w^i_{t-1} p(d_t\|\Psi^i_t) p(\Psi^i_{t,r}\|\Psi^i_{t-1}, \Psi^i_{t,s})$.
4. Resample: Set $t \leftarrow t+1$, return to step 1.

Table 2.5: The PFMT Algorithm.

2.4.1 Initialisation and Importance sampling

When mode tracking we must first decide how to split the state. Vaswani [2008] suggests that it is best to split the state so '$\Psi_{t,s}$ contains the minimum number of dimensions ... [such that] $p(\Psi_{t,r}|\Psi^i_{t-1}, \Psi^i_{t,s}, d_t)$ [the probability of the r section of the state at time t given the probability of the state at time $t-1$ and the s part of the state and the observations at time t] is unimodal'. Once we have chosen how to split the state we can begin the PFMT algorithm. The first step shown in Table 2.5 is the importance sampling step. This step is identical to the PF, the only difference being that we only treat the Ψ_s dimensions in this way. Here we sample from the prior pdf and forecast the Ψ_s dimensions one timestep.

2.4.2 Mode Tracking

The second step in the PFMT algorithm is MT. The mode tracking finds values for the Ψ_r part of the state for each particle by minimising the cost function in equation (2.12) with respect to $\Psi_{t,r}$.

$$J^i(\Psi^i_{t,s}, \Psi_{t,r}) = \frac{1}{2}(J_o(\Psi^i_{t,s}, \Psi_{t,r}) + J_q(\Psi_{t,r})), \tag{2.12}$$

where

$$J^i_o(\Psi^i_{t,s}, \Psi_{t,r}) = \left(d_t - H\begin{pmatrix}\Psi^i_{t,s}\\ \Psi_{t,r}\end{pmatrix}\right)^T R^{-1}\left(d_t - H\begin{pmatrix}\Psi^i_{t,s}\\ \Psi_{t,r}\end{pmatrix}\right), \tag{2.13}$$

and

$$J^i_q(\Psi_{t,r}) = (\Psi_{t,r} - f_r(\Psi^i_{t-1}))^T[Q_{rr} - Q_{rs}Q^{-1}_{rr}Q^T_{rs}]^{-1}(\Psi_{t,r} - f_r(\Psi^i_{t-1})), \tag{2.14}$$

where R is the observation error covariance matrix and Q is the model error covariance matrix. Q is split into four sections corresponding to how Ψ has been split. The J_q term above (equation (2.14)) is the the cost function of the model whereas the J_o term (equation (2.13)) is the cost function of the observation. The $(\Psi_{t,r} - f_r(\Psi^i_{t-1}))$ term in equation (2.14) represents how far the background is from the model state, the corresponding term in equation (2.13) represents how far the observation is from the model state.

Now we have forecast values for each part of the state the particles must be weighted.

2.4.3 Weighting and Resampling

The weighting for the PFMT differs from the ordinary PF. The weights are calculated using,

$$w_t^i = \frac{\omega_t^i}{\sum_{j=1}^{N} \omega_t^j} \tag{2.15}$$

where $\omega_t^i = \omega_{t-1}^i p(d_t|\Psi_t^i) p(\Psi_{t,r}^i|\Psi_{t-1}^i, \Psi_{t,s}^i)$. We see these normalised weights are similar to that given in equation (2.7).

Now the particles have been weighted the PFMT continues as the ordinary particle filter. We resample the particles using the same stratified resampling algorithm. Once the particles have been resampled we reset the weights for each particle to $w^i = \frac{1}{N}$. As with the PF we return to the forecast step to continue the assimilation. We repeat the iteration until the final forecast time is reached.

Vaswani [2008] found that the introduction of MT significantly decresed the number of particles required to achieve desirable results. The experiments carried out by Vaswani [2008] showed the 'superior performance of the PFMT ... over the PF-original' when both codes were run with the same number of particles.

2.5 Summary

In this chapter previous work has been considered. Kalnay [2002] showed how DA is used to combine observations and a numerical model to produce a model state which accurately describes the observed reality. Bannister [2009] showed how ensemble DA forecasts a number of states to give us a pdf associated with the forecasts. We have then considered work by van Leeuwen [2009] and Doucet et al. [2000a] to show how the PF makes use of Bayes Theorem to improve on the simple ensemble DA. We considered the idea of weighting, giving particles a value that represent how close they are to the truth, and then use this to resample the particles. From Snyder et al. [2008] we saw that as the state of the system increased the number of particles required to give an accurate solution increased exponentially. For this reason Vaswani [2008] introduced the idea of MT. Vaswani [2008] also showed that the PFMT performed better than the ordinary PF. Therefore this should allow us to reduce the number

of particles required to solve the system and hence decrease the computational cost of the assimilation. We test this hypothesis in Chapter 6. In order to test these ideas we require a model dynamical system, this is seen in Chapter 3.

Chapter 3

The Model

In this chapter we describe a suitable model that can be used with the PF and PFMT algorithms. A fairly simple model has been chosen so it is not complex to implement and not too computationally costly to solve. The chosen model has previously been used with a PF [Pham, 2000][Bengtsson et al., 2003] allowing us to compare the results we obtain.

We consider how to solve the model numerically, some numerical solutions are produced and these are compared to previous work [Pocock, 2008][Pham, 2000][Bengtsson et al., 2003] to validate our implementation. The error in the numerical solution is also considered to check the numerical approximation is converging correctly.

3.1 The Lorenz Model

The model chosen is based on the deterministic Lorenz model [Lorenz, 1963], shown in equations (3.1), (3.2) and (3.3).

$$\frac{dx}{dt} = \sigma(y - x), \tag{3.1}$$

$$\frac{dy}{dt} = x(\rho - z) - y, \tag{3.2}$$

$$\frac{dz}{dt} = xy - \beta z, \tag{3.3}$$

where x, y and z are the dimensions of the state, t is time and σ, ρ and β are constants. The constants σ, ρ and β may take any value greater than zero, however we have chosen $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$ as these are the classic Lorenz parameters [Lorenz, 1963]. Choosing these values allows us to compare our work with previous studies. The choice of $\rho = 28$ causes the system to exhibit chaotic behavior [Lorenz, 1963].

Equations (3.1), (3.2) and (3.3) show a deterministic formulation of the Lorenz model. We wish our model to represent unresolved atmospheric processes with added noise. To allow for this we transform (3.1), (3.2) and (3.3) into stochastic differential equations of the form,

$$d\mathbf{\Psi} = a(\mathbf{\Psi})dt + b(\mathbf{\Psi})dW. \tag{3.4}$$

The first term on the RHS represents the RHS of the original Lorenz equations, the second term is an addition of 'noise' by a Wiener process [Kloden and Platen, 1999]. The stochastic Lorenz equations are given by (3.5), (3.6) and (3.7).

$$dX = \sigma(Y - X)dt + B_X dW_X \tag{3.5}$$

$$dY = (X(\rho - Z) - Y)dt + B_Y dW_Y \tag{3.6}$$

$$dZ = (XY - \beta Z)dt + B_Z dW_Z. \tag{3.7}$$

Now X, Y and Z represent random variables, where as ρ, β and σ are held at the same constant value. dW still represents a value from the Wiener process. The constants B_X, B_Y and B_Z can be changed to alter the importance of the noise in the system. The choice of these parameters is discussed in section 3.6.

A second reason for choosing a stochastic model is that we require some sort of stochastic process for the PF to work. Section 2.3.4 discusses the idea of resampling, the stochastic process is required here to separate copies of particles that are made.

3.2 The Euler-Maruyama Approximation

To solve the model numerically a time discrete approximation must be used. The scheme chosen is a stochastic generalization of Euler's method known as the Euler-Maruyama (EM)

approximation [Kloden and Platen, 1999]. Given a SDE of the form in (3.4) the EM approximation has the form

$$\Psi_{t+1} = \Psi_t + a(\Psi_t)\Delta_t + b(\Psi_t)\Delta W_t, \tag{3.8}$$

where Δ_t is the timestep and $\Delta W_t = W_{t+1} - W_t$. W_t is the value of the Wiener process at timestep t and hence ΔW_t is the difference between the W values at two consecutive timesteps. These random variables ΔW_t are independent and from the distribution $N(0, \Delta_t)$. In practice we take ΔW_t directly from $N(0, \Delta_t)$ rather than calculating values of W_t at each timestep. Thus the size of the timestep affects the noise added to the model. Here the values of Δ_t may vary in length, for simplicity we consider equal length timesteps Δ. The EM approximation for the model is given in equations (3.9) to (3.11),

$$X_{t+1} = X_t + \sigma(Y_t - X_t)\Delta + B_X \Delta W_X, \tag{3.9}$$

$$Y_{t+1} = Y_t + (X_t(\rho - Z_t) - Y_t)\Delta + B_Y \Delta W_Y, \tag{3.10}$$

$$Z_{t+1} = Z_t + (X_t Y_n - \beta Z_t)\Delta + B_Z \Delta W_Z. \tag{3.11}$$

3.3 Convergence and Stability

It is also important to verify that the scheme is numerically stable and converges to the true solution of the continuous problem as Δ goes to zero. We now present some Theorems that give some information on convergence and stability of the scheme.

3.3.1 Convergence of the EM Scheme

We start by considering the convergence criteria of the scheme. Kloden and Platen [1999] and Kliemann and Namachchivaya [1995] state both strong and weak convergence results for the EM approximation. First we define the meanings of strong and weak convergence, then the results for the EM approximation are stated in Theorems 2 and 3. We also include Theorem 4, the convergence results for one timestep of Euler's method.

Definition 1 Strong Convergence

If Ψ_t^T is the true value of Ψ at time t, $\tilde{\Psi}_t^\Delta$ is an approximation of Ψ obtained from a numerical

model with timestep Δ and E is the expectation operator. Then an approximation $\tilde{\Psi}$ converges with strong order $\gamma > 0$ if there exists a finite constant $G > 0$ such that

$$E(|\Psi_t^T - \tilde{\Psi}_t^\Delta|) \leq G\Delta^\gamma$$

for all step sizes $\Delta \in (0, 1)$.

Definition 2 Weak Convergence

If Ψ_t^T is the true value of Ψ at time t, $\tilde{\Psi}_t^\Delta$ is an approximation of Ψ obtained from a numerical model with timestep Δ and E is the expectation operator. Then an approximation $\tilde{\Psi}$ converges with weak order $\alpha > 0$ if for any polynomial g there exists a finite constant $G_g > 0$ such that

$$|E(g(\Psi_t^T)) - E(g(\tilde{\Psi}_t^\Delta))| \leq G_g\Delta^\alpha$$

for all step sizes $\Delta \in (0, 1)$.

Theorem 2 Strong Convergence for the EM approximation

The EM approximation converges with strong order $\gamma = 0.5$.

Theorem 3 Weak Convergence for the EM approximation

The EM approximation converges with weak order $\alpha = 1$

Theorem 4 Convergence for one timestep of Euler's method

For one itteration of Euler's method it is expected that $\epsilon^\Delta = O(\Delta^2)$

We omit the proofs of these Theorems here. However the proof of Theorems 2 and 3 can be found in Kloden and Platen [1999] and the proof of Theorem 4 in Lee and Schiesser [2003].

3.3.2 Stability of the EM Scheme

Both Saito and Mitsui [1996] and Kloden and Platen [1999] discuss the stability of the EM approximation, however only linear systems are dealt with. As stability of nonlinear systems is complex, even in an ODE case, we shall consider the stability of the EM approximation for the linear equation

$$d\Psi_t = \lambda\Psi_t dt + dW_t, \tag{3.12}$$

where λ is a complex number with $\Re(\lambda) < 0$. The EM scheme for equation (3.12) with an equidistant step size Δ is given in equation (3.13),

$$\Psi_{n+1}^{\Delta} = \Psi_n^{\Delta}(1 + \lambda\Delta) + \Delta W_n. \tag{3.13}$$

From this we obtain,

$$|\Psi_n^{\Delta} - \tilde{\Psi}_n^{\Delta}| \leq |1 + \lambda\Delta|^n |\Psi_0^{\Delta} - \tilde{\Psi}_0^{\Delta}|, \tag{3.14}$$

where $\tilde{\Psi}_n^{\Delta}$ is the solution obtained from starting at $\tilde{\Psi}_0^{\Delta}$. From equation (3.14) we see the additive noise terms cancel out, giving us the same region of absolute stability as the deterministic Euler method shown in Figure 3.1.

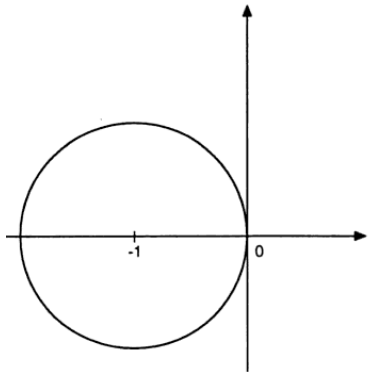

Figure 3.1: The stability region of Euler's Method. The stable region lies within the circle on the complex plane.

As we can not compute the stability region directly for the EM approximation with the Lorenz equations we look to the literature to try and find a stable timestep to use. Xiao et al. [2009] use the Lorenz equations with parameters $\sigma = 10$, $\rho = 30$ and $\beta = \frac{8}{3}$, the only parameters that differ are ρ and the noise parameter we have called B. To solve the equations Xiao et al. [2009] use the EM approximation and a timestep of $\Delta = 0.01$. In our experiments we have also used a timestep of $\Delta = 0.01$ and have seen no evidence of numerical instability with this timestep.

3.4 The Euler-Maruyama Code

The Euler-Maruyama scheme (equations (3.9), (3.10) and (3.11)) was implemented in MATLAB. The noise terms ΔW_X, ΔW_Y and ΔW_Z are independent and from the distribution $N(0, \Delta_t)$. To calculate these values in MATLAB the *randn* command was used [Moler, 2006]. The *randn* command produces pseudo-random numbers from the distribution $N(0, 1)$. As we require values from $N(0, \Delta_n)$ the values of ΔW_X, ΔW_Y and ΔW_Z are calculated scaling the output of *randn* by $\sqrt{\Delta}$, (i.e $\sqrt{\Delta}N(0, 1)$).

3.5 Model Validation

In this section we present results from both quantitative and qualitative experiments to validate the scheme. The sensitivity of the model to the choice of the parameters B_X, B_Y and B_Z is also considered.

3.5.1 Qualitative Results

First the qualitative results are considered. We plot various solutions of the Lorenz systems to check they are as expected. Each Figure (3.2, 3.3, 3.4 and 3.5) contains four separate plots. The graphs on the left are of x (top), y (middle) and z (bottom) against time t. The plot on the right is a plot of x against z, this is the classic butterfly plot produced by the Lorenz Equations.

To gain an initial solution we run the Euler-Maruyama scheme from time $t = 0$ until $t = 20$ with a timestep of $\Delta = 0.01$ and $B = 0$ (no noise) from initial conditions $x = 0.00001$, $y = 0.00001$ and $z = 2.00001$. This allows us to see a smooth numerical solution to the deterministic equations and may help determine a suitable level of noise later. The solution is plotted in Figure 3.2.

It can be seen that the x, y and z values oscillate, small oscillations lead to the spiraling motion in the classic butterfly plot whereas the large oscillations cause the jumps between the left hand and right hand wings. The butterfly plot on the RHS is the classic Lorenz picture we

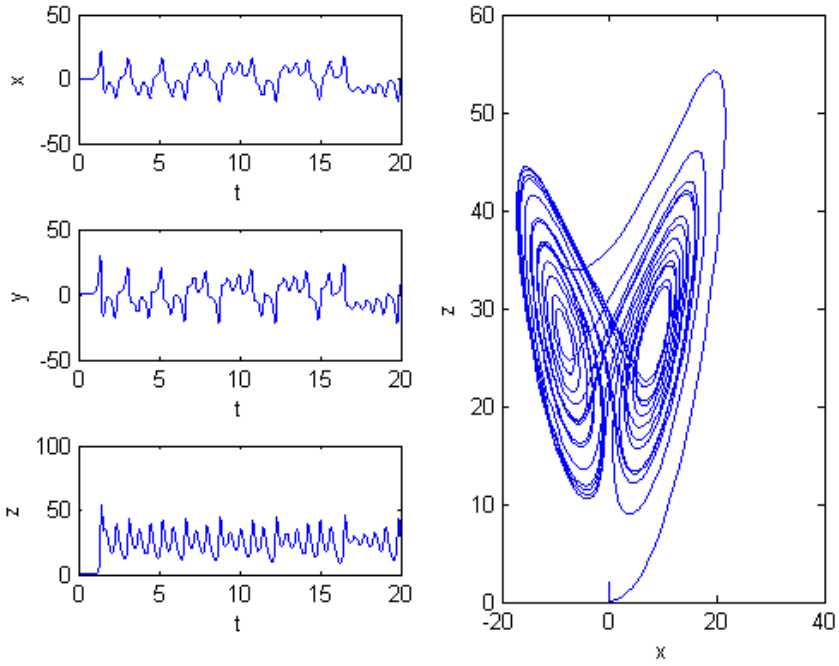

Figure 3.2: Solution to stochastic Lorenz equation with no noise added. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 20$ with a timestep $\Delta = 0.01$.

expect, as seen in [Bengtsson et al., 2003], qualitatively verifying that the basic EM scheme is working.

We now add some noise to the scheme. We run the scheme as before but with $B_X = 1$, $B_Y = 1$ and $B_Z = 1$, the final time is reduced to $t = 0.4$ to simplify the solution and allow us to focus on the added noise. The solution is plotted in Figure 3.3, it is similar to the solution in Figure 3.2 but the trajectory is no longer smooth. The trajectory contains small bumps at each timestep, these are related to the added noise.

The values of $B_X = 1$, $B_Y = 1$ and $B_Z = 1$ put a large amount noise in the scheme and we do not necessarily require the noise to be this large. The sensitivity of the model to these parameters is discussed later in section 3.6. In another experiment we set $B_X = B$, $B_Y = B$ and $B_Z = B$ where $B = 0.1$. Even with this small amount of noise two solutions with identical initial conditions can be completely different over time, with different realisations of

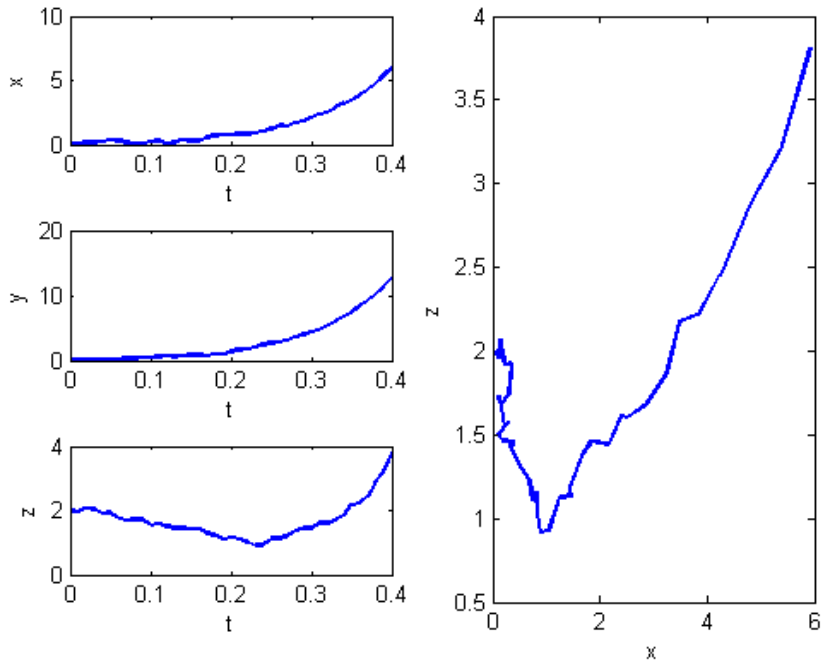

Figure 3.3: Solution to stochastic Lorenz equation. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 0.4$ with a timestep $\Delta = 0.01$. Noise parameters $B_X = 1$, $B_Y = 1$ and $B_Z = 1$.

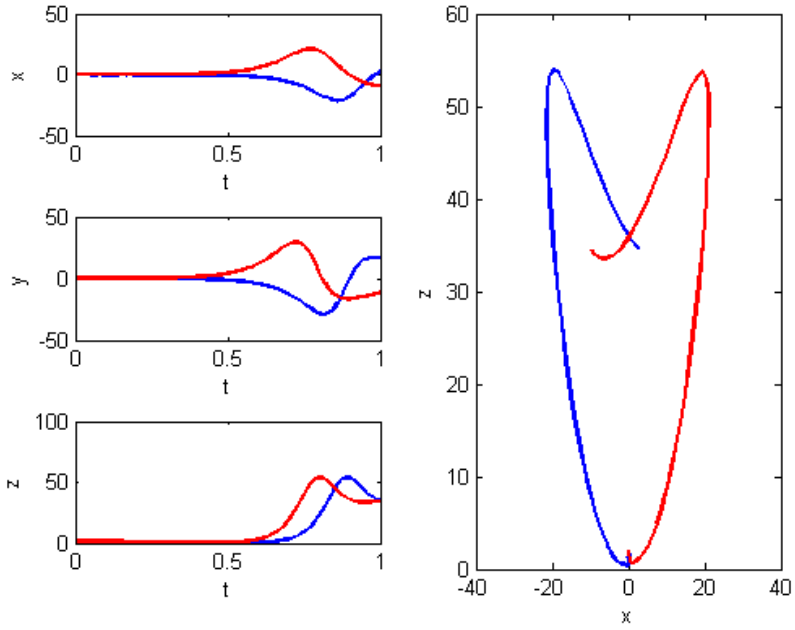

Figure 3.4: Two solutions from identical initial conditions. The red and blue lines represent two solutions from initial conditions $x = 10.00001$, $y = 10.00001$, $z = 20.00001$ but with different realisations of noise. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameters $B = 0.1$.

noise. Figures 3.4 shows how solutions with the same initial conditions can differ.

The location of the initial conditions in phase space also helps to determine how quickly two solutions with different initial conditions differ over time. The initial conditions for Figure 3.4 lie in an unstable area of the system, the small amount of noise added pushes one solution to the left wing of the butterfly and the other to the right hand side. However the initial conditions for Figure 3.5 lie in a more stable part of the system and the trajectories produced follow similar paths over a longer time period.

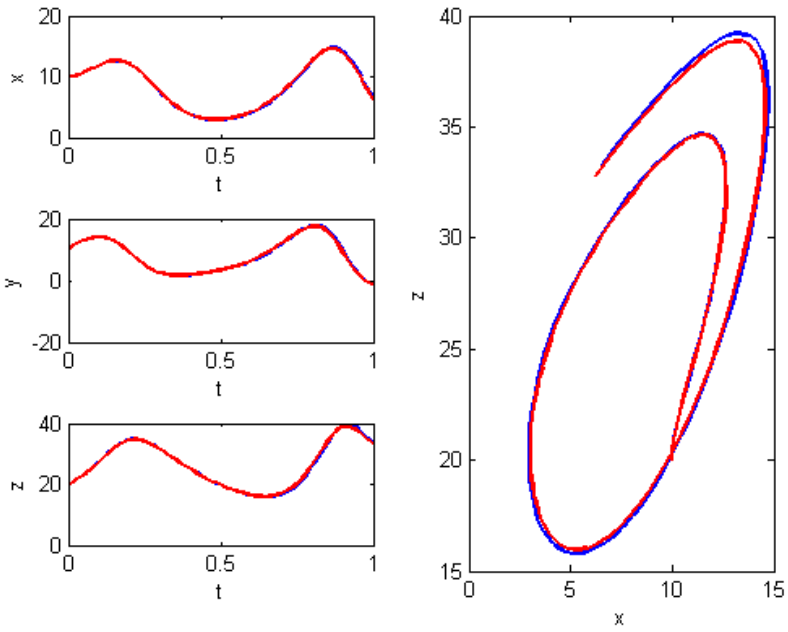

Figure 3.5: Two solutions from identical initial conditions. The red and blue lines represent two solutions from initial conditions $x = 10.00001$, $y = 10.00001$, $z = 20.00001$ but with different realisations of noise. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameters $B = 0.1$.

3.5.2 Quantitative Results

We have now seen that the Euler-Maruyama scheme provides the expected qualitative results, producing the classic butterfly plot we expect from the Lorenz attractor and showing the chaotic nature of the system by giving differing solutions from the same initial conditions. We now look for some quantitative results to support the results already obtained.

Theorem 2 and 3 showed that we expect the EM approximation to converge with strong order $\gamma = 0.5$ and weak order $\alpha = 1$. This suggests that a plot of $\log(\Delta)$ vs. $\log(Error)$ will produce a line with gradient between 0.5 and one. We shall see that this gradient is dependent on the noise parameter B. Setting $B = 0$ simplifies the EM approximation to Euler's method [Ascher and Petzold, 1998], for one timestep of this scheme Theorem 4 shows this error should

be $O(\Delta^2)$. Therefore if we run our scheme with no noise we expect the plot of $\log(\Delta)$ vs. $\log(Error)$ to show a straight line with gradient two.

To be able to plot the $\log(\Delta)$ vs. $\log(Error)$ graph we need to be able to calculate the the error at a given timestep. Normally this would be calculated using equation (3.15)

$$\Psi_t - \tilde{\Psi}_t^{\Delta} = \epsilon^{\Delta}, \tag{3.15}$$

where ϵ^{Δ} is the error obtained from the difference between Ψ is the true solution at time t and $\tilde{\Psi}_t$ the numerical approximation at the same time calculated with timestep Δ. As we do not have an analytic solution we cannot do this directly. We expect our error $\epsilon = O(\Delta^{0.5})$, so we would expect equation (3.15) to be,

$$\Psi_t - \tilde{\Psi}_t^{\Delta} = O(\Delta^{0.5}). \tag{3.16}$$

To calculate our error with a given timestep Δ we compute equation (3.16) using both timestep Δ and timestep 2Δ. This leads to equations (3.17) and (3.18).

$$\Psi_t - \tilde{\Psi}_t^{\Delta} = O(\Delta^{0.5}). \tag{3.17}$$

$$\Psi_t - \tilde{\Psi}_t^{2\Delta} = O((2\Delta)^{0.5}). \tag{3.18}$$

subtracting equation (3.17) from (3.18) gives equation (3.19).

$$\tilde{\Psi}_t^{\Delta} - \tilde{\Psi}_t^{2\Delta} = O((2\Delta)^{0.5}) - O(\Delta^{0.5}) = O(\Delta^{0.5}). \tag{3.19}$$

Therefore to calculate the error at time Δ we calculate the numerical solution using Δ and 2Δ we then subtract one from the other. The resulting value is the error associated with the timestep Δ. We carry out these calculations for 5 different Δ's and plot the results. As we have three dimensions we must decide how to calculate an average of the error, we choose to use a Root Mean Squared Error (RMSE).

In Figure 3.6 $\log(\Delta)$ vs. $\log(Error)$ is plotted, we plot this for various values of B. We include $B = 0$ when the EM scheme is reduced to Euler's Method. The actual error for each Δ is plotted as a cross, we then fit a line of best fit.

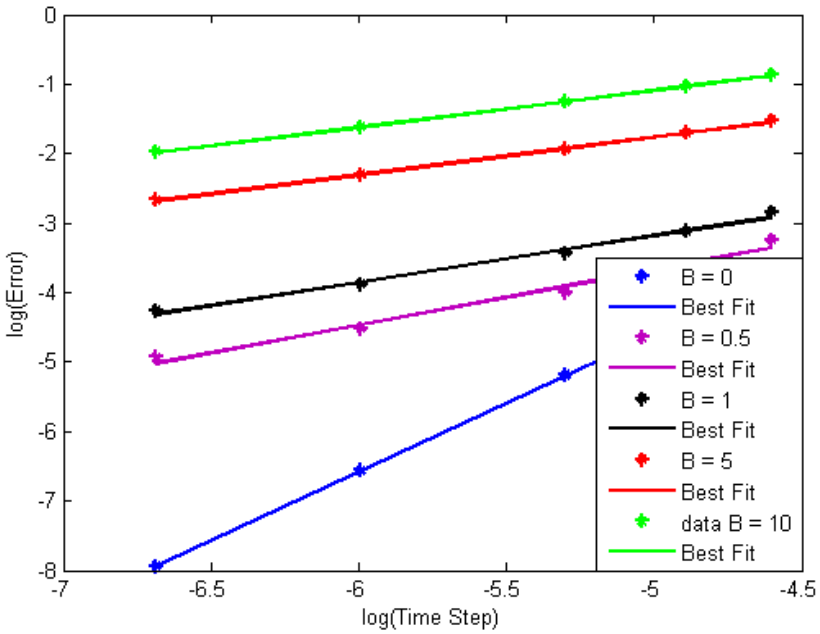

Figure 3.6: Graph showing how the size of the timestep affects the error, for varying values of B. $\log(\Delta)$ is plotted against $\log \epsilon^{\Delta}$ the log of the error produced with a particular size timestep. Crosses represent experimental results obtained as described in the text. A line of best fit is then added to each set of data.

From Figure 3.6 we see that the line produced when $B = 0$ has a gradient of two suggesting the error for one timestep for the Euler's method is $\mathrm{O}(\Delta^2)$ which is what we expect. We see that the remaining plots all have similar gradients, although the gradient reduces slightly as the value of B increases. The gradients of the best fit lines for $B = 5$ and $B = 10$ are both 0.5 suggesting an order of convergence of $\frac{1}{2}$, this means the EM scheme is converging with the expected order seen in Theorem 2. The best fit lines for $B = 0.5$ and $B = 1$ are slightly higher than 0.5, they still fall between the strong and weak convergence values given in Theorems 2 and 3.

3.6 Sensitivity to Noise Parameters

In equations (3.5), (3.6) and (3.7) we see that the parameters B_X, B_Y and B_Z affect the size of the noise added to the equation. For simplicity we set each of these values equal, this value is B ($B_X = B_Y = B_Z = B$). To see how the value of B affects the solution we run the model as in Figure 3.3 but with a final time of $t = 0.1$ with various values of B and with identical noise realisations for each trajectory. These are shown in Figure 3.7. The Figure shows that the smaller the value of B the smoother the trajectory obtained. From now on we choose to set $B = 0.1$, this value is chosen as it allows us to see relatively smooth, yet noisy solutions.

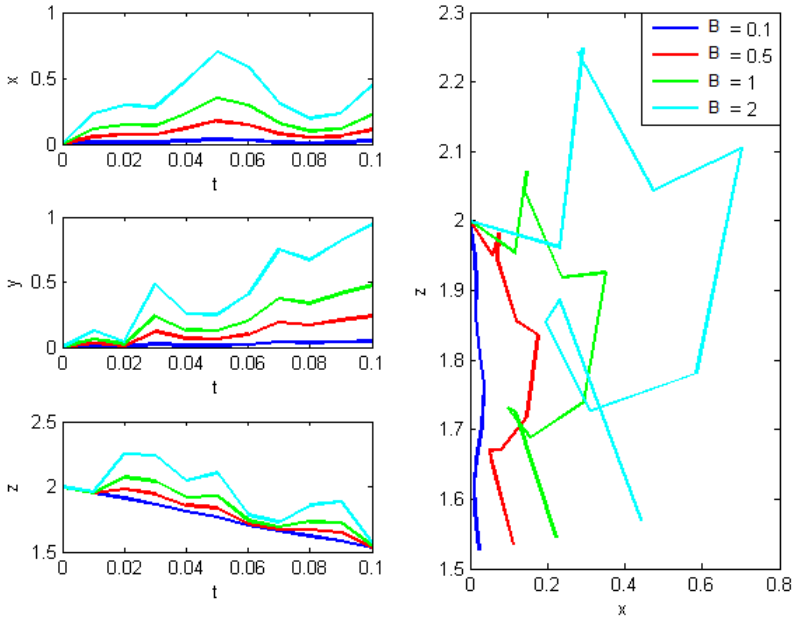

Figure 3.7: Affect on solution of changing B. Each line shows the EM solution to the stochastic Lorenz equations for a different value of B. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 0.1$ with a timestep $\Delta = 0.01$.

3.7 Summary

In this Chapter we have seen the stochastic Lorenz equations ((3.5),(3.6) and (3.7)) that will be solved using the EM approximation and estimated by the PF and the PFMT. We have seen the EM approximation, a simple generalisation of Euler's method, this is used to forecast the particles in the PF. We have seen that the EM approximation gives the expected solution to the Lorenz equations, the x z plot being the classic Lorenz butterfly. We also considered the order of convergence of the EM scheme and showed that the scheme converged as expected. We have considered how the noise in the scheme affects the solution, and how it can cause two completely different solutions from the same initial conditions.

Chapter 4

Implementation and Diagnostics for the PF and PFMT

In this chapter the PF and PFMT codes are discussed. We explain how the codes are implemented and consider some of the parameters used. We also consider some of the diagnostics used to show the accuracy of the results obtained from the code.

4.1 The Particle Filter Code

4.1.1 Initialisation

To be able to compare and compute our numerical solution we require a true solution and a set of observations, creating these is the first stage of the PF code. The true solution is given by one run of the Euler-Maruyama approximation seen in equations (3.9, 3.10 and 3.11), these truth values at time t are referred to as x_t^T, y_t^T and z_t^T. To create the observations the values of the truth need to be perturbed slightly, these slight perturbations represent instrument and human error that is involved with collecting observations. This error is represented by ε_t in equation (2.2), for this code we choose ε_t to be taken from the distribution $N(0, 0.04)$. To perturb the truth values a pseudo-random number from $N(0, 0.04)$ is added to each, this is done using the *randn* MATLAB command that was used in the EM code in section 3.2.

The values output by *randn* are multiplied by $\sqrt{0.04}$ so the perturbations are taken from the correct distribution. We denote these observations at time t using x_t^O, y_t^O and z_t^O.

Now we have a truth and some observations, we are ready to sample our initial particles. The particles are created in a similar way to the observations. For each particle required a small perturbation is added to each of the initial conditions. This time *randn* is multiplied by $\sqrt{\Delta}$ so the perturbations are taken from $N(0, \Delta)$. The final stage in the initialisation is to equally weight each of the particles, $w^i = \frac{1}{N}$.

4.1.2 The Particle Filter Iteration

The main PF is now run, this is an implementation of steps 2 through to 5 of the Bootstrap Filter seen in Table 2.4, these steps are repeated for the number of time steps required. The first stage is to forecast each of these particles one timestep, this is done using the EM approximation. The EM approximation used is identical to that used to create the truth, but for each particle the initial conditions are peturbed as discussed in section 4.1.1. Once each particle has been forecast its weight can be calculated. The w^i for each particle are calculated using equation (4.1),

$$w_t^i = \frac{\exp(-\frac{\sigma_o}{2}((x_t^{F,i} - x_t^O)^2 + (y_t^{F,i} - y_t^O)^2 + (z_t^{F,i} - z_t^O)^2)}{N}, \tag{4.1}$$

where σ_o is the observation variance, $x_t^{F,i}$, $y_t^{F,i}$ and $z_t^{F,i}$ are the forecast values at time t for each particle and x_t^O, y_t^O and z_t^O are the observations at time t. Once these weights have been calculated they are normalised. These weights are then used in the resampling. The resampling is a simple implementation of the stratified resampling algorithm seen in Table 2.3. Now the particles have been resampled the weights are reset and we return to the forecasting stage.

4.2 The PFMT code

The initialisation of the PFMT is the same as the initialisation for the PF code seen in section 4.1.1. Once this initialisation is complete the PFMT iteration can begin, this is an implementation of the PFMT algorithm seen in Table 2.5. The first step in the iteration is

to forecast the particles one time step. However as we are MT we need to treat the state to be mode tracked differently to the rest of the state. We must forecast the state to be mode tracked state with no noise, we do this using the EM scheme with $B = 0$ for one timestep. The EM scheme has been modified so it only forecast the required state. The remaining states are also forecast with this modified EM scheme but with $B = 0.01$. Now each dimension has been forecast we move on to the MT section. We mode track by minimising the cost function in equation (4.2).

$$J^i i(\Psi^i_{t,s}, \Psi_{t,r}) = \frac{1}{2}(J_o + J_b), \tag{4.2}$$

where

$$J^i_q(\Psi_{t,r}) = (\Psi_{t,r} - f_r(\Psi^i_{t-1}))^T Q^{-1}_{rr}(\Psi_{t,r} - f_r(\Psi^i_{t-1}))$$

and J_o is as in equation (2.13). Equation (4.2) is a specific case of the cost function seen in equation (2.12) in section 2.4.2. Here we have been able to simplify the model error matrix to Q_{rr} as our matrix Q contains only zeros off the diagonal. By minimising this equation we gain our forecast values for the dimensions of the state that we wish to mode track. Now we have all the values we require we can weight the particles. The particles are weighted slightly differently to the weighting in the PF code. We used our forecast values to calculate the value of the cost function in equation 4.2 then we use this and

$$w^i = \frac{\exp(-J^i)}{N}, \tag{4.3}$$

to compute the weights. Now the weights have been computed they are normalised and the particles can be resampled. The resampling in the PFMT is identical to the resampling in the PF. Now the particles have been resampled the weights are reset to $w^i = \frac{1}{N}$ and we return to the forecasting stage. The iteration is repeated until the final time is reached. The results are then plotted.

4.3 Diagnostics

4.3.1 Root Mean Squared Error

To show how the PF performance changes as the number of particles changes the RMSE is calculated. The RMSE measures the average magnitude of the error, it is calculated using the true solution and the particle mean. We calculate the particle mean for each dimension of the state. Equation 4.4 shows how the particle mean is calculated using the forecast values $x_t^{F,i}$ and normalised weight w_t^i.

$$\bar{x}_t = \sum_{i=1}^{N} x_t^{F,i} w_t^i \tag{4.4}$$

If the true solutions at each time are x^T, y^T and z^T and the particle means are $\bar{x}$, $\bar{y}$ and $\bar{z}$ then the RMSE is calculated using equation (4.5),

$$RMSE = \sqrt{\frac{(x_t^T - \bar{x}_t)^2 + (y_t^T - \bar{y}_t)^2 + (z_t^T - \bar{z}_t)^2}{3}}. \tag{4.5}$$

The RMSE is calculated at each timestep and its value plotted against t.

However the RMSE only considers one statistic from the PDF. Although we base our analysis of the PF on this, we must remember that the PF represents the whole pdf. It may be possible that if we considered a different diagnostic we would see slightly different results.

4.3.2 Rank Histograms

As well as considering the RMSE we use another diagnostic known as the Rank Histogram (RH) [Hamill, 2000] to asses the quality of the PF. RH are useful for diagnosing errors in the mean and spread of the particles and determining the reliability of ensemble forecast. Unfortunately the RH assumes that each particle is equally weighted, due to this and the resampling we include we must be careful how we interpret our results. A better alternative to the RH may be the Reliability Diagram [Wilkes, 1995], we consider this in our Future work (Section 7.2).

4.3.2.1 Creating the Rank Histograms

RH are created by considering where the value of the observation at time t falls compared to the values of the forecast particles at the same time. The particles at time t are sorted into ascending order so that if $\tilde{x}^i$ are the weighted particles we have $\tilde{x}^1 \leq \tilde{x}^2 \leq ... \leq \tilde{x}^N$. This creates $N+1$ bins the first for values lower than $\tilde{x}^1$, the second for values between $\tilde{x}^1$ and $\tilde{x}^2$ and so on, the final bin is for values higher than $\tilde{x}^N$. The bin, also know as rank, in which the observation at time t falls is noted. This value is tallied over all times t. The resulting tally is plotted and this is known as the RH.

4.3.2.2 Interpreting Rank Histograms

As the RH assumes each particle is equally weighted there at each time there is an equal probability that the observation, d, will fall into any of the $N+1$ ranks. This implies that the resulting rank histogram should have bars of equal height (Figure 4.1 (a)). Histograms that are not uniform can give us information on the quality of the ensemble. Many of the common histograms that are produced from ensemble forecasts are discussed in Hamill [2000], four of the most common histograms obtained are shown in Figure 4.1. A U-shaped histogram (Figure 4.1 (b)) suggests a possible lack of variability in the particle sample, whereas an excess of variability overpopulates the central ranks (Figure 4.1 (c)). Having the left half of the histogram overpopulated (Figure 4.1 (d)) suggests that particles are positively biased,

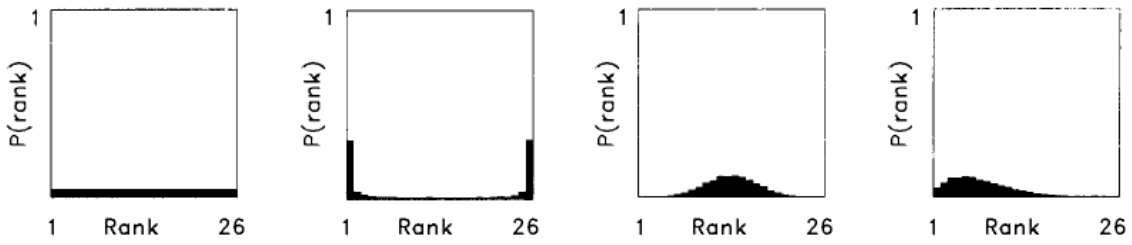

Figure 4.1: Common rank histograms obtained. These histograms show the number of times the observation falls in each bin. Plots from Hamill [2000]

overpopulation in the right half implies the particles are negatively biased. However these are not the only possible reasons for these types of behavior.

4.4 Summary

In this chapter we have seen how the PF and PFMT are implemented. We have seen that they both share the same intialisation which involves a truth run. The values from this are then perturbed to give us the observations. The next stage of the PF code is an implementation of the Bootstrap Filter seen in Table 2.4. Whereas the PFMT code continues with an implementation of the PFMT algorithm seen in Table 2.5. As well as considering the implementations of the PF and PFMT code, the diagnostics for the solutions are also considered. We have seen how to calculate the RMSE and the RH. How to interpret the RH has also been discussed.

Chapter 5

The Particle Filter Solutions

In this chapter we use the PF to estimate solutions of the stochastic Lorenz system. The PF will be run for a number of different situations and the qualitative results will be compared. Primarily we are interested in how the number of particles affects the results (Section 2.3.6) so the experiments shall focus on this. Once the solutions to the Lorenz equations have been seen, the RMSE and Rank Histogram plots are also presented for each of the solutions so we can quantitatively assess the PF.

In section 2.3.3 we saw that if the resampling step was not included in the PF algorithm all the weight ended up on one particle. We wished to check that our code produced the same results. Results included in Appendix A show that if we run the code described in section 4.1 without the resampling step we see similar results to those in Snyder et al. [2008].

5.1 Lorenz Trajectory solutions

The solutions in this section show the trajectories obtained when the PF is run. We are interested in how the solutions differ depending on the number of particles used. For this reason the only input data that differs when the code is run is the number of particles used. The remaining input data stays the same for each run of the model, this data is as follows: Start time $t = 0$, End time $t = 1$, Time step $\Delta = 0.01$, Initial $x = 0.00001$, Initial $y = 0.00001$ and Initial $z = 2.00001$. The output from the code results in the Figures included in the

following section, these figures have the same format as in Chapter 3.

5.1.1 Results for $N = 2$

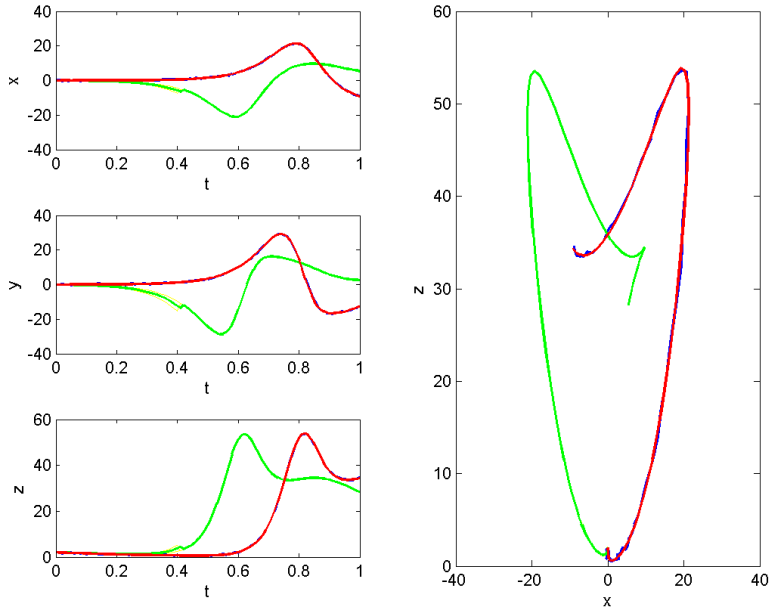

Figure 5.1: PF solution of the Lorenz equations with $N = 2$ particles. The red line is the true solution, the dark blue line is observations, the green line is the particle mean calculated in equation (4.4), the light blue lines represent the particle trajectories obtained for each particle. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$.

In Figure 3.4 we saw how, due to the chaotic nature of the system, two solutions from the same initial conditions can differ. The idea of resampling in the PF is that particles that do not follow the true solution are discarded in favour of particles that are closer to the true solution. Therefore if particles do start to differ from the true solution as in Figure 3.4 we expect to see their trajectories terminate as these particles are 'resampled' and the trajectory restart from a new position corresponding to a different particle. One of the problems with the PF is that if too few particles are used then there is the possibility that none of the particles follow the truth. This is the case in Figure 5.1, where we see that both the particles, and hence the

particle mean, have their solution on the LHS of the Lorenz butterfly whereas the true solution is on the RHS. This illustrates the case when the particles all have a significantly different solution to the truth, this means that the particle with the highest weight does not reflect the true solution so even the resampling cannot bring our solution closer to the truth. For future experiments and verification a larger number of particles will be used so this situation does not occur.

5.1.2 Results for $N = 5$

Figure 5.1 showed that if the PF is run with too few particles the results that are obtained may not follow the true solution, however even with just a few more particles sensible results can be obtained. The results shown in Figure 5.2 show the PF solution to the stochastic Lorenz equations with $N = 5$ particles. In these and subsequent plots it is difficult to see the observations as they lie under the truth, the only points when the observations are visible

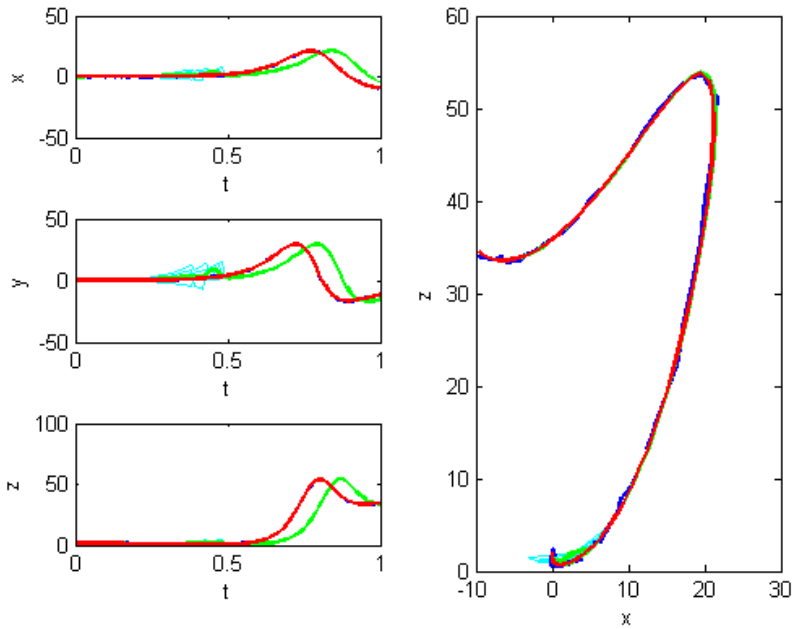

Figure 5.2: PF solution of the Lorenz equations with $N = 5$ particles. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$.

is when the observation error is large. Similarly when the particles collapse down they are hidden beneath the particle mean and the truth if the solution is accurate enough. In Figure 5.2 on the x against z plot we see some particles initially heading away from the true solution. However some of the particles are following the true solution and we see the PF working by terminating the particles that are moving away from the true solution and duplicating particles that are following the true solution. Between $t = 0$ and $t = 0.5$ is when the particle resampling of particles shows most, the resampling is particularly prominent in the x against t and y against t plots. After some time, and due to the relatively low numbers of particles being used, all the particles have very similar values and lie close to the particle mean. Although this time there are enough particles to provide us with a solution that is similar to the truth, after $t = 0.5$ all the x, y, and z particle mean results start to differ from the observations and the true solutions. We see that the x, y and z plots all show that the numerical solution is similar to the true solution but shifted in time. This phase error is masked in the x vs. z plot as this plot has no time dependence.

We would like to know how many particles we need for the particle mean to be sufficiently close to the true solution over our time window. To do this we increase the number of particles to see how this improves the solution.

5.1.3 Results for $N = 50$

The number of particles is now increased to $N = 50$, as there are more particles we expect the particle mean solution to be closer to the exact solution. Although we expect the particle mean to be closer to the true solution, we also expect to see more particles that diverge from the true solution, the PF should terminate these particles and create duplicates of particles that are following the true solution.

Figure 5.3 shows the PF solution to the stochastic Lorenz equations with $N = 50$ particles. As expected we see that more particles do diverge from the true solution, it is also possible to see these particles being resampled and becoming a duplicate of particle closer to the true solution. We also see that in the x, y and z plots the particle mean follows the solution more accurately than in Figure 5.2, this suggests that increasing the number of particles has

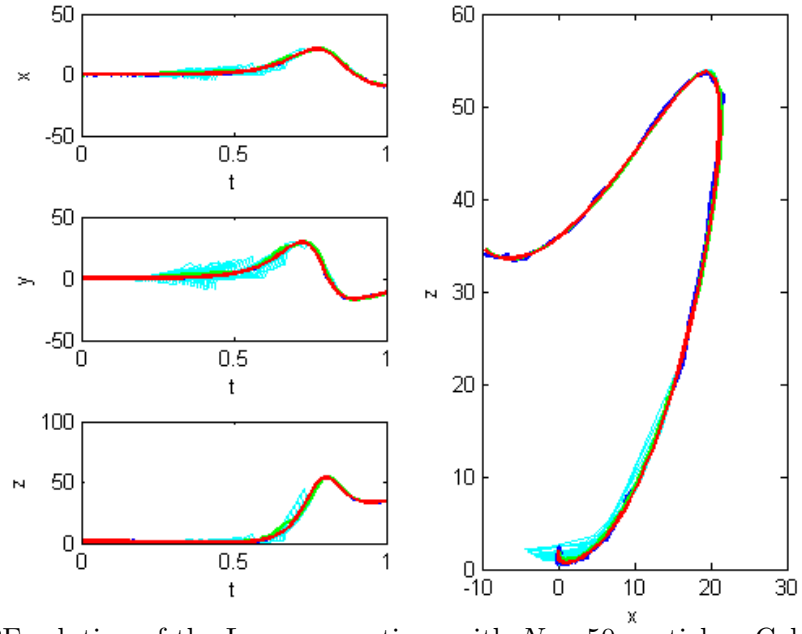

Figure 5.3: PF solution of the Lorenz equations with $N = 50$ particles. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$.

improved the results produced by the PF. Comparing the x, y and z plots from Figure 5.3 and Figure 5.2 it is possible to see that the particles have a larger spread. The resampling discards particles that diverge from the true solution moving all the particles toward the truth. However over time we still see some particles terminating and restarting from a different position, this is due to the stochastic system and the noise that is add in the Euler-Maruyama scheme.

5.1.4 Results for $N = 500$

Finally we consider the results when 500 particles are used, these are shown in Figure 5.4. As expected these are the best PF results obtained, the particle mean lies underneath the true solution and the particles stay close to the true solution.

We have now seen that as the number of particles is increased it appears that the quality of the solution also increases. However all the results we have seen so far are only based on one truth solution. We would expect to obtain qualitatively similar behavior if the PF was

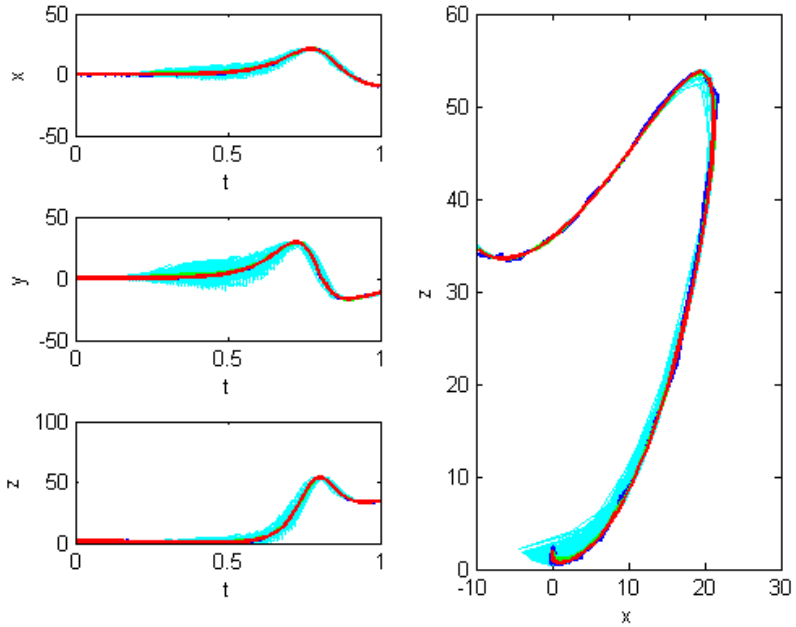

Figure 5.4: PF solution of the Lorenz equations with $N = 500$ particles. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$.

run with a different true solution and noise realisation. Due to time limitation we have not verified this. One other limitation of the PF is that we have compleate observations at every timestep. In reality the set of observations would be incomplete and not avaliable at each timestep. It would be desirable to decrease the number of observations we have to make the situation more realistic, this is discussed in section 7.2.

5.2 Quantitative Results

We now present the quantitative results obtained. The RMSE is calculated and some rank histograms plotted to verify the Lorenz results.

5.2.1 Root Mean Squared Error

In Figure 5.5 the RMSE for $N = 5$, $N = 50$ and $N = 500$ is plotted. The RMSE for $N = 5$, $N = 50$ and $N = 500$ are calculated from the output given by the model runs that were used

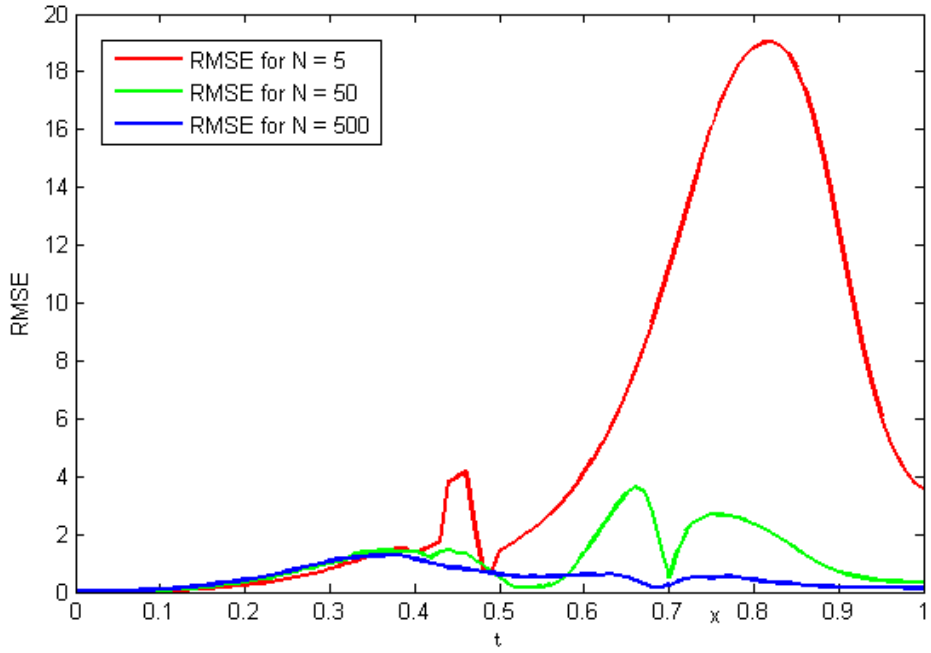

Figure 5.5: RMSE for various number of particles. Calculated from model runs with initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameters $B = 0.1$.

to plot Figure 5.2, 5.3 and 5.4 respectively.

We see that the RMSE for $N = 5$ is much larger than than the RMSE for the other model runs with a very high error between $t = 0.5$ and $t = 1$. There are also some sudden reductions in the RMSE. The $N = 50$ RMSE is a significant improvement on the $N = 50$ RMSE however we still see that the error is highest between $t = 0.5$ and $t = 1$. There are also some occasions where we see a sudden reduction in the error before it begins to rise again. The RMSE for $N = 500$ is the smallest and smoothest, this supports the qualitative plots that the more particles used the more accurate the numerical solution. This is also supported by Pham [2000] and Vaswani [2008] who see the RMSE reduce as the number of particles is increased.

We do not expect the RMSE plot to be smooth as the results are obtained from a set of SDEs and the noise added though the Euler-Maruyama scheme will affect the magnitude

of the RMSE. One reason for the variation in the RMSE is thought to be caused 'when the system state enters a strong nonlinearity section' [Pham, 2000] of the Lorenz Attractor. This is likely to be the cause of the higher RMSE seen between $t = 0.5$ and 1. These fluctuations are also seen in [Pham, 2000] when they consider a RMSE over a longer period of time. The plots they obtain show regions of low RMSE then regions of high RMSE. As this is due to the system chosen we may also expect to see this behavior if the PF was run f0r a longer period of time.

As well as the areas of high and low RMSE that we see, we also observed some sudden reductions in the RMSE before it began to rise again. One possible reason we suggest for this is the affect of the resampling. It is possible that the sudden decrease in the RMSE for $N = 5$ at time $t = 0.5$ is cause by diverging particles being resampled at the previous timestep. This would mean that the resampled particles lie closer to the truth hence the RMSE would rapidly decrease. As these particles are forecast some begin to diverge causing the RMSE to rise again.

5.2.2 Rank Histograms

As well as considering the RMSE we present RH for the $N = 5$, $N = 50$ and $N = 500$ solutions of the Lorenz equations. Figure 5.6 shows the RH for the PF solution to the Lorenz equations with $N = 5$ particles. The top left panel shows the rank histogram for the x dimension, the y dimension is shown in the top right. The bottom left is the rank histogram for the z dimension, the bottom right is a combination of the previous three. The clearest shape in all the histograms is the one for the z dimension, this histogram is clearly U-shaped suggesting a lack of variability in the particles. The x histogram also shows some U-shaped characteristics supporting the suggestion of a lack of variability. This is also the nature suggested by the combined plot. However we mentioned earlier that the rank histogram assumes the particles are all equally weighted, as this is not the case we must consider how it may affect the result. In the PF we resamle particles according to their weights. It is this resampling that changes how we interpret the RH. When we resample the particles we obtain become similar, over time the particles become closer to one another representing the solution closest to the truth,

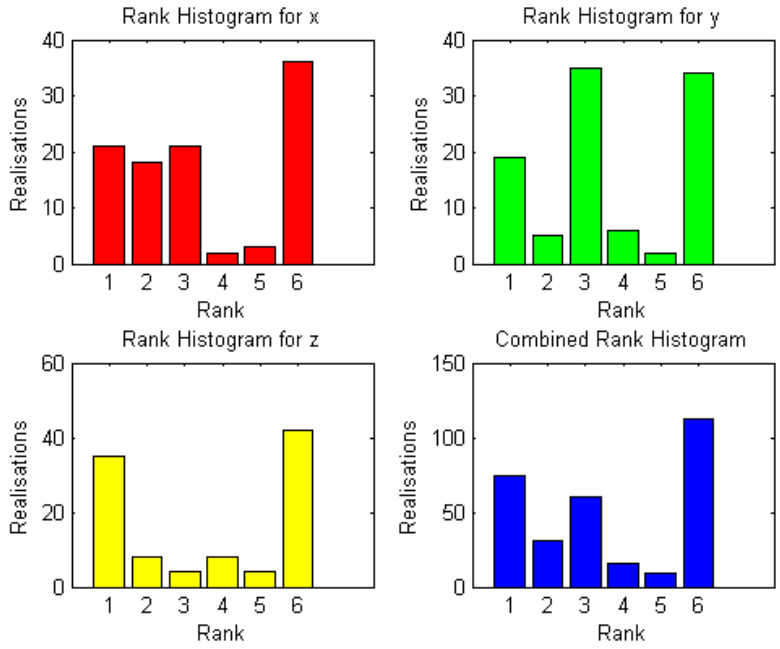

Figure 5.6: Rank Histograms for $N = 5$. Calculated from a model run with initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameters $B = 0.1$. 'Realisations' refers to the number of occurrences of the observation per bin.

this is easily seen in Figure 5.2 where the particle mean hides the actual particle trajectories. This means that over time we expect the variability of our particles to decrease. Also when considering Figure 5.2 we see that initially the particles are fairly similar to the exact solution, however between $t = 0.5$ and $t = 0.8$ all the particles have a value below the observations, and after $t = 0.8$ the particles all have a higher value than the truth. Therefore we expect all the tallies from $t = 0.5$ and $t = 0.8$ to go into the first bin, where as all the tallies after $t = 0.8$ go in the last bin. This means more than half the tallies are guaranteed to go in the first and last bins leading to a U-shaped rank histogram.

We now consider the rank histogram for the PF solution to the Lorenz equations with $N = 50$. From the RH from $N = 5$ we know that the original solution can give us some information on how we may expect the RH to differ, so first we consider Figure 5.3. From the x vs. t plot after $t = 0.8$ we see that all the particles have values higher than the truth so we

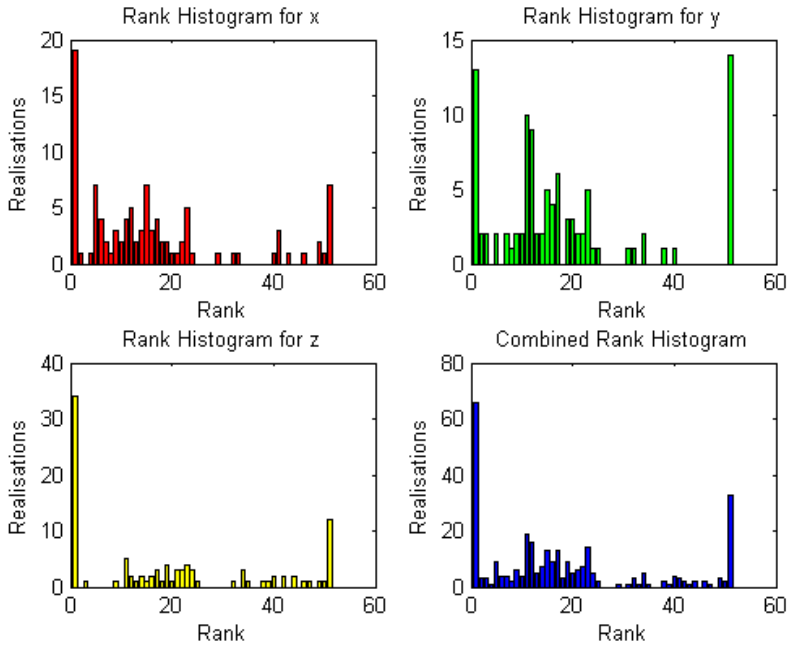

Figure 5.7: Rank Histograms for $N = 50$. Calculated from a model run with initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameters $B = 0.1$. 'Realisations' refers to the number of occurrences of the observation per bin.

expect all the tallies from this time to be in the first bin making this bar higher. From the y vs. t plot between $t = 0.7$ and $t = 0.9$ the particle values are higher than the truth and after $t = 0.9$ the particle values are lower than the truth. This means the y RH is expected to be U-shaped. The z vs. t plot also shows that after $t = 0.7$ all the particles fall to one side of the truth. Until $t = 0.8$ the particles are lower than the truth, after this the particles are higher than the truth, thus we expect to see a U-shaped rank histogram for z, but with the lowest rank being fuller than the highest. The RHs for the solution for $N = 50$ are plotted in Figure 5.7.

We see that the RHs are as expected, with the x RH having large number of tallies in the first bin, the y RH being U-shaped and the x RH being U-shaped but with more tallies in the first bin. Although we have some explanation of the high tallies in the first and last bins we would still like to see the values in the middle bins being uniform. Unfortunately the bins

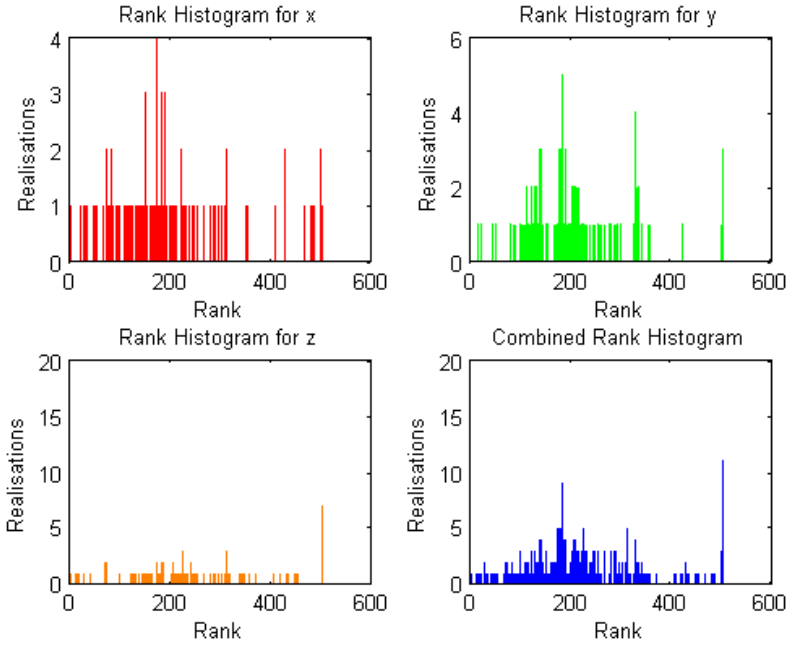

Figure 5.8: Rank Histograms for $N = 500$. Calculated from a model run with initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameters $B = 0.1$. 'Realisations' refers to the number of occurrences of the observation per bin.

in these rank histograms are not as uniform as we expect, this may be due to the ensemble collapse and the lack of variability this causes.

Finally we consider the RH for the PF solutions to the Lorenz equations with $N = 500$ particles. As the number of particles has increased we expect there to be more variability in our particles and hence do not expect to see the U-shaped behavior in the RH. This is supported by Figure 5.4 as we can see that the particles stay spread out through out the window and although the spread is smaller in some places the particles do not collapse down to a single trajectory. We consider the RH for this solution in Figure 5.8. This time we see that each of the bins that contain tallies are much more equal, however we see many bins with no tallies at all. This is because there are only 100 timesteps to tally over but 501 bins that could be filled. Therefore we are only interested in the filled bins being uniform and with the exception of just a few bins this is what we see. The only RH which is not so uniform is the z plot. In Figure 5.4 we see that at the very end of the time window the particles in the z

vs. t plot do collapse down and it is difficult to see the individual particle trajectories. It is possible that it is the values here that contribute the the higher tally in the last bin of the z RH.

5.3 Summary

In this Chapter we have seen both quantitative and qualitative results from the PF. We have seen that if too few particles are used then the PF can give a solution that differers significantly from the truth. Once the number of particles is increased the solution improves. The more particles the better the solution we obtain, and the smaller the RMSE we see. For a small number of particles the PF produces a trajectory with a phase error compared to the true solution. This phase error is obvious in the x, y and z plots, however it is hidden in the x vs. z plot as this plot is independent of time. When the number of particles is increased to $N = 500$ the PF solution follows the truth closely. When a small number of particles is used the RH show that we do not have enough variability in our ensembles. A large number of particles is required to get enough variability across the whole PDF.

Chapter 6

Particle Filters with Mode Tracking

In this chapter we consider the both the quantitative and qualitative results for the PFMT code. These are then used to determine the best way to split the state. The affect of increasing the number of particles is also considered. We then go on to see how the difference between the observation error variance and the background error variance affects the results from the PFMT.

6.1 Splitting the State

When MT we must decide how we split the state. In section 4.1.1 we saw thhat Vaswani [2008] proposed that it is best to split the state so '$\Psi_{t,s}$ contains the minimum number of dimensions ... [such that] $p(\Psi_{t,r}|\Psi^i_{t-1}, \Psi^i_{t,s}, d_t)$ is unimodal'. This suggests that the way we split the state will affect the result obtained. To determine how we should split the state we will consider the Lorenz solutions from the PFMT after splitting the state in each possible way.

6.1.1 Qualitative Results

In this section we only discuss and include the plots for the solutions of the best and worst ways to split the state. The remaining plots are included in Appendix B. The following plots are all from the PFMT code run from initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$, time

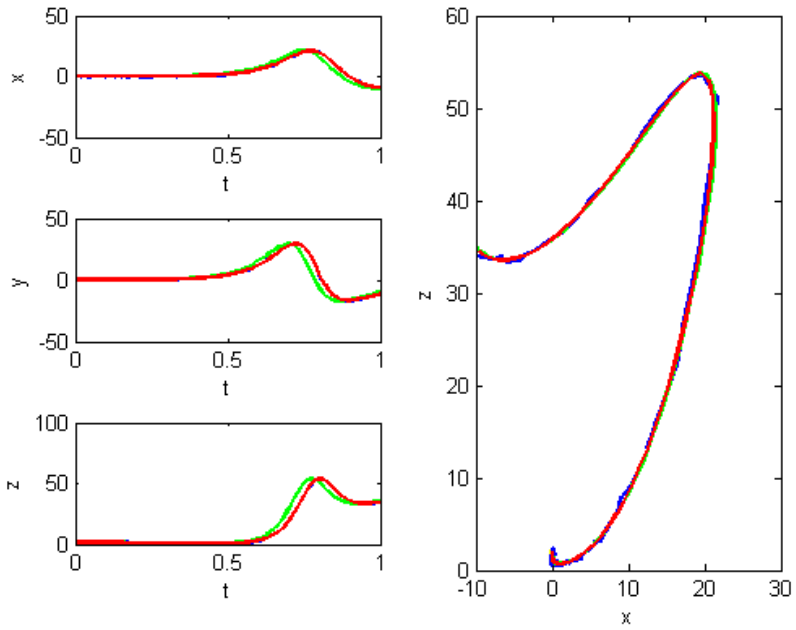

Figure 6.1: PFMT solution of the Lorenz equations with $\Psi_r = (z)$ and $\Psi_s = (x, y)$. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$. $N = 500$ particles.

$t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$, a noise parameter $B = 0.1$ and $N = 500$ particles. It is only the way that the state is split that varies. We have chosen to use $N = 500$ particles as this gave the best results from the PF and assume it will in the PFMT.

Figure 6.1 shows the solution when only the z dimension is mode tracked, $\Psi_r = z$. We see the solution is fairly good, although there is still a small phase error and the particle mean can still be distinguished from the truth between 0.5 and 0.8. This appeared to be the worst of the mode tracked solutions we obtained.

It appears that the best result obtain came from MT the x and y dimensions, $\Psi_r = (x, y)$. This result is shown in Figure 6.2. We see that this solution is better than the one shown in Figure 6.1. The phase error is no longer present and it is difficult to distinguish the particle mean from the truth.

When comparing Figures 6.1 and 6.2 with Figure 5.4, the PF solution with $N = 500$, we clearly see that the particles visible in Figure 5.4 are not visible in Figures 6.1 and 6.2. This

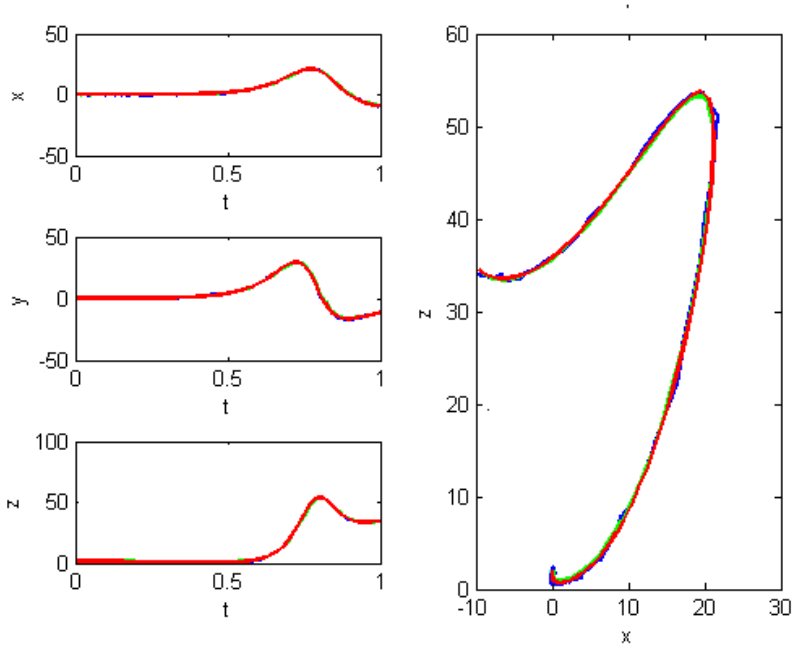

Figure 6.2: PFMT solution of the Lorenz equations with $\Psi_r = (x, y)$ and $\Psi_s = (z)$. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$. $N = 500$ particles.

highlights the lack of particle variability in the PFMT.

We also split the state in four other ways and the solutions for these are shown in Appendix B. These qualitative results were similar in each case all appearing slightly better from the result shown in Figure 6.1 but worse than the solution seen in Figure 6.2.

6.1.2 Quantitative Results

To give a more comprehensive view of how the results differ we consider the quantitative results for each of the possible ways to mode track the state. First the RMSE of each solution is considered, these are plotted in Figure 6.3. We see that until $t = 0.4$ the error from each solution is small and similar. After this we begin to see the errors grow. The errors are highest for each solution at the times expected, in the interval $t = 0.5$ to $t = 0.8$ when the system is in

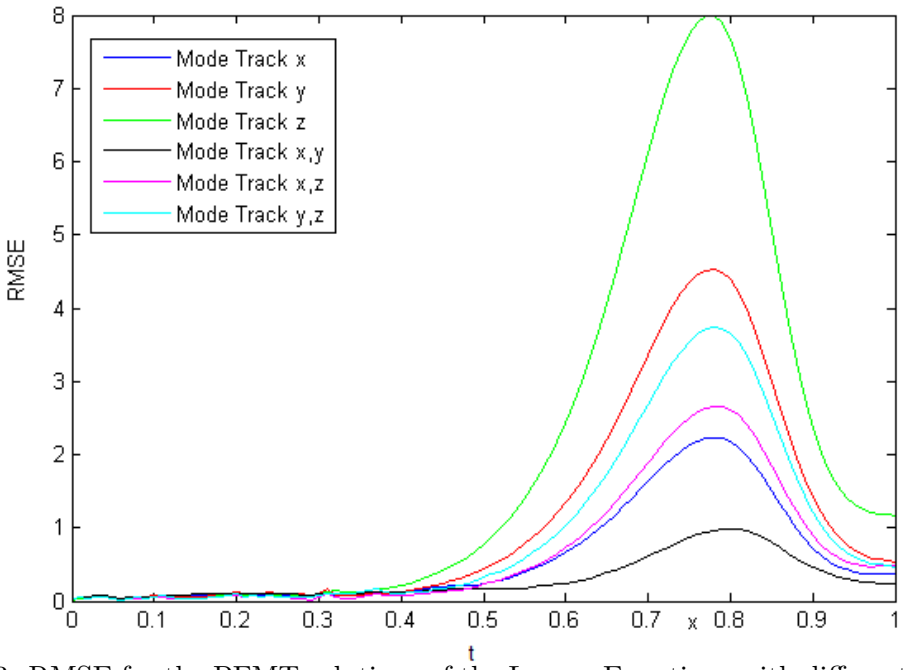

Figure 6.3: RMSE for the PFMT solutions of the Lorenz Equations with different parts of the state mode tracked

a strong nonlinearity region. It is clear that the solution with the highest error is as expected, the one from MT the z dimension. The next highest error is nearly half of the highest error, this is produced from the solution when y is mode tracked. The next highest errors are from the solution when the y and z states and x and z are mode tracked. We have two solutions left for which we have not considered the RMSE. From the quantitative results we could see that the best solution was from the code when the x and y dimensions were mode tracked. So we expect the the next error seen on Figure 6.3 to be the one associated with the solution when x is mode tracked. The smallest error is from the solution obtained from the PFMT solution when x and y and mode tracked. This error is much smaller than the previous errors, and half the error of the x mode tracked solution implying that the best PFMT solution is obtained when both the x and y dimensions are MT.

We look to equations (3.5) to (3.7) to give some idea of why choosing $\Psi_r = (x, y)$ gives the most accurate result. As MT produces the more accurate result we expect the $\Psi_{t,r}$ dimensions to be more accurate. When z is mode tracked its value depends on two non mode tracked

dimensions and hence the result obtained is not so accurate. However when x and y are mode tracked the value of z is dependent on these two mode tracked values, so the PF z value is also kept accurate and hence the over all solution is excellent.

We may have expected that MT two dimensions always gave better results than MT one dimension. However from Figure 6.3 we see that this is not the case. Although we see the most accurate solution is produced by MT the x and y dimensions we see that the next best choice would be to mode track only the x dimension. It is possible this is due to multi-modal behavior of one or more of the dimensions. Vaswani [2008] suggests that MT the wrong dimensions can result in larger RMSE. In section 4.1.1 we saw that Vaswani [2008] proposed that it is best to split the state so '$\Psi_{t,s}$ contains the minimum number of dimensions ... [such that] $p(\Psi_{t,r}|\Psi^i_{t-1}, \Psi^i_{t,s}, d_t)$ is unimodal'. It is possible that when we mode track, say x and z, this does not hold but when we mode track just x it does, hence MT just one dimension would give a better result than MT two dimensions.

We have now determined which is the most accurate solution when $N = 500$, and therefore which states it is best to mode track. We shall now only consider results obtained from the PFMT when both x and y are mode tracked.

6.2 Changing the number of particles

We now consider how results from the PFMT vary when we change the number of particles used. These results will then be compared to the results from the PF code to see the difference that the MT makes. The following plots are all from the PFMT code run from initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$, time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$, a noise parameter $B = 0.1$ and $\Psi_r = (x, y)$. It is only the number of particles that varies.

6.2.1 Qualitative Results

We consider first the result obtained when the number of particles is set to $N = 5$, this is shown in Figure 6.4. The equivalent results for the PF can be seen in Figure 5.2. We see that this result is rather poor. Again we see a phase error in the numerical solution and hence the

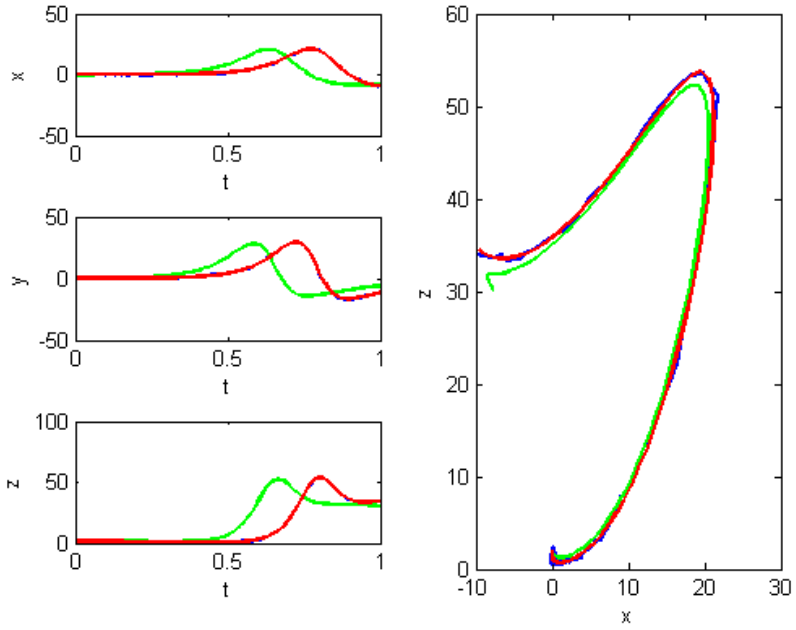

Figure 6.4: PFMT solution of the Lorenz equations with $\Psi_r = (x, y)$ and $\Psi_s = (z)$. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$. $N = 5$ particles.

particle mean differs from the truth significantly. Comparing this result to Figure 5.2 we see that it appears the result from the PFMT is worse than the PF. Again we notice a difference between the PF and PFMT plots, the ensemble variability collapse occurs much sooner in the PFMT results.

Next the results when $N = 50$ are considered, these are seen in Figure 6.5. The equivalent results for the PF can be seen in Figure 5.3. We see that this result is still fairly poor, and more comparable to the $N = 5$ result from the PF seen in Figure 5.2. This suggests the PFMT requires more particles to achieve a more accurate result. This is contrary to the result seen in Vaswani [2008] where it is shown that the PFMT requires a smaller number of particles to produce similar results to the PF.

The result when $N = 500$ is shown in Figure 6.2, the equivalent results for the PF are shown in Figure 5.4. This is as expected the most accurate of the PFMT solutions we have seen as it is run with the most particles and the state is split to give the best result. To see

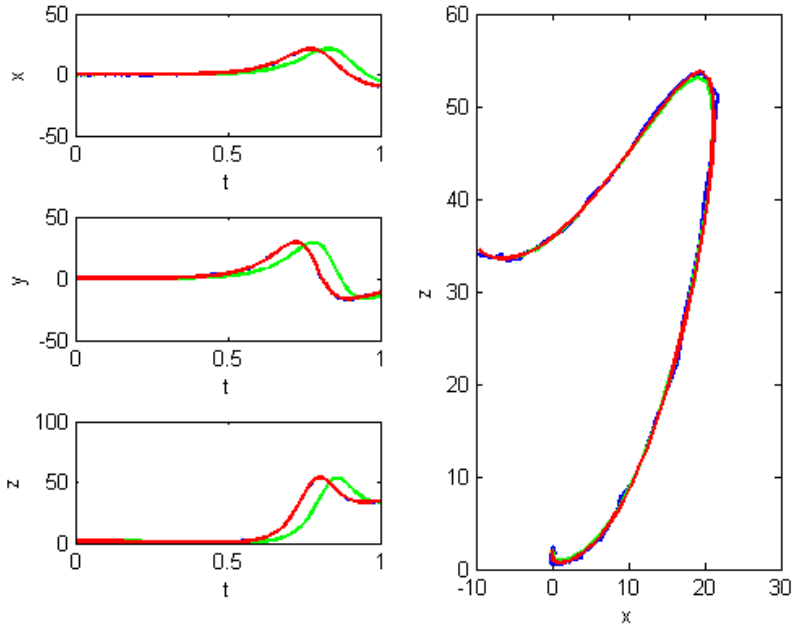

Figure 6.5: PFMT solution of the Lorenz equations with $\Psi_r = (x, y)$ and $\Psi_s = (z)$. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$. $N = 50$ particles.

how the results actually compare with the PF we consider the quantitative results.

6.2.2 Quantitative Results

Figure 6.6 shows the RMSE for $N = 5$, $N = 50$ and $N = 500$ particles for both the PF code and the PFMT code when $\Psi_r = (x, y)$. We see that the two highest errors correspond to the PF and PFMT runs with $N = 5$, however the PFMT error is much higher than the PF error. This corresponds to the qualitative results we saw. It was also suggested that the PFMT $N = 50$ was more comparable to the PF $N = 5$ solution and this is obvious in Figure 6.6 as these errors are very similar. The PF solution with $N = 50$ is much lower than the PF solution with $N = 5$ and is initially similar to the PF solution with $N = 500$. Another difference we see between the PF and PFMT results is that the PFMT RMSE results are much smoother and we do not see the sudden jumps that we see in the PF RMSE solutions. We finally compare the RMSE for the PF and PFMT when $N = 500$. We see that these

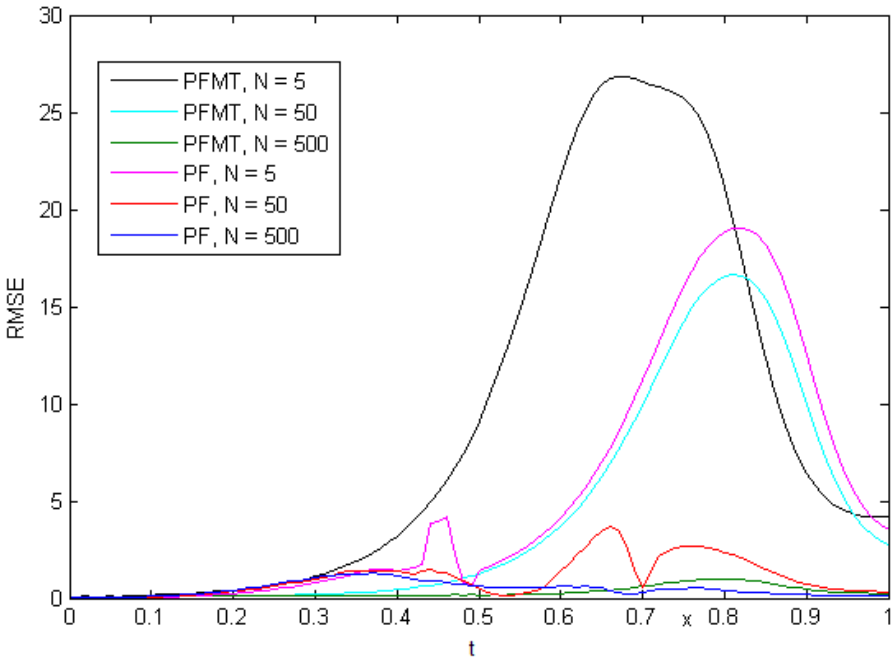

Figure 6.6: RMSE produced from the PFMT ($\Psi_r = (x, y)$) and PF codes with various values of N. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. Time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$.

errors are very similar, the PF error is slightly higher initially where as the PFMT error is higher at the end. However the PF error is higher for longer and overall it is higher than the PFMT error suggesting that the PFMT solution is better than the PF solution. These results contradict results seen in Vaswani [2008] that suggest that the PFMT always produces more accurate solutions than the PF.

6.3 Sensitivity of MT to error variances

When testing the PFMT code it was discovered that the values in the Q and R matrices made a significant difference to the result from the MT. Due to this we shall consider what happens when we change the variance of the observation error, the diagonal elements of the R matrix, as these are high compared to the model error Q. The error in the solution decreases as the number particles increases, for this reason we run the code with $N = 50$ so the difference is

more obvious. The code is run for $\Psi_r = (x, y)$ with initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. Time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$ and noise parameter $B = 0.1$.

6.3.1 Qualitative Results

The PFMT solutions to the Lorenz equations when $N = 50$ were shown in Figure 6.5. For these results the error variance of the observations was set to 0.04. We ran the code again but with the variance set to 0.01, this makes the observation and model error variance equal. As we have decreased an error we expect the results to improve. The results are shown in Appendix C in Figure 7.6. Comparing Figure 6.5 to Figure 7.6 we see a significant improvement and the results when the observation error variance is set to 0.01 are similar to the results produced from the PFMT code when $N = 500$.

6.3.2 Quantitative Results

To compare the results of changing the observation error variance we again consider the RMSE, this can be seen in Figure 6.7.

We see that bringing the values of the background and observation error closer together we significantly improve the results obtained from the PFMT with the error being reduced to below one from above 16. This shows that by reducing the observation error just slightly we can greatly improve the accuracy of our assimilation.

The sensitivity of solutions to the difference between the observation and model errors is discussed in Johnson et al. [2005] in the context of 4D-var. They show that selecting the correct values for the observation and model error variances 'is critical for extracting the maximum amount of useful information from the observations'. This result supports the theory that the PFMT is sensitive to the error variances. It also suggests that it may be possible to improve the results from the PFMT if we find the correct ratio between model and observation variances.

In section 3.6 we considered how to scale the model noise using the noise parameter B. We must take into account the affect this may have on the ratio between the observation and model error. Decoupling the Q matrix from the noise parameter may be one way to overcome

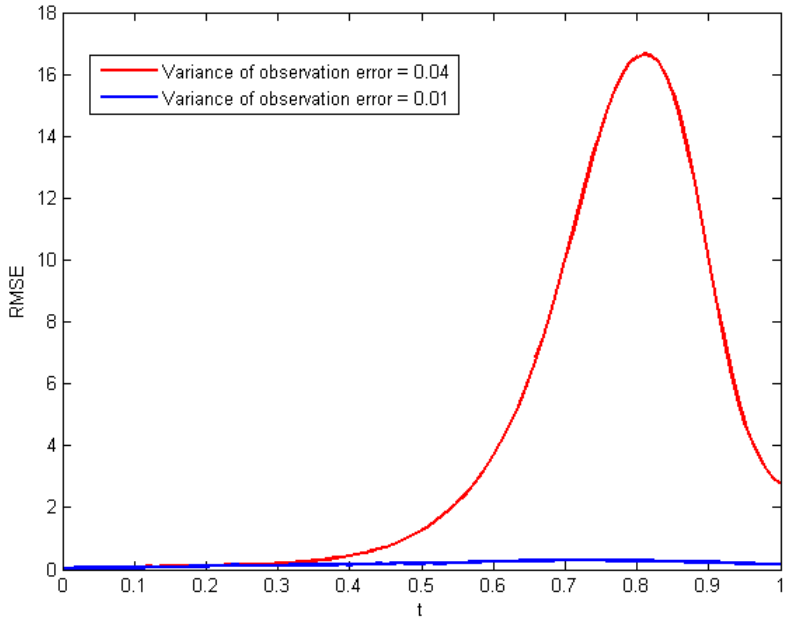

Figure 6.7: RMSE for PFMT solutions to the Lorenz equations with varying error variances

the affect the B parameter has on this ratio. This decoupling is one section discussed in future work (section 7.2).

6.4 Summary

In this Chapter we have seen the results from the PFMT code. We have discovered that the best way to split the state when MT is $\Psi_r = (x, y)$ and $\Psi_s = z$. Splitting the state in this way gives us the most accurate solution, we have shown this in Figures 6.2 and 6.6. We have also seen that the PFMT leads to ensemble variability collapse much sooner than in the PF. Like the PF we also see that the solution improves as we increase the number of particles. The most surprising result of this Chapter, and indeed this thesis, is seen in section 6.2. We see that the PFMT, for small numbers of particles, does not produce a result that is better than the PF. It is only when the number of particles is increased to $N = 500$ that the PFMT produces more accurate solutions than the PF. These results differ to those seen in Vaswani [2008] which suggest that the results obtained from the PFMT are always superior to the

results from the PF. We have also seen that the results from the PFMT appear to be quite sensitive to the ratio between the observation and model noise. This suggestion is supported by Johnson et al. [2005] who show that, in 4D-var, a correct ratio between the observation and model errors is required to achieve the best result.

Chapter 7

Conclusion

7.1 Summary and Discussion

In Chapter 2 the ideas of DA, ensemble DA, PF and PFMT were discussed. We saw that DA combines observations and model data to produce more accurate state estimates. The use of ensemble DA allows estimation of uncertainty in the resulting analysis. The PF allows us to model non-Gaussian pdfs using Bayes Theorem (Theorem 1) to update the prediction pds to give the filtering pdf. Each particle in the pdf has a weight associated with it according to how close they are to the observation. Unfortunately without resampling over time all the weight goes onto one particle making our statistical information meaningless [Snyder et al., 2008]. To overcome this weighting problem we have seen that we must resample the data.

Snyder et al. [2008] showed that the PF works well for small systems but for systems as large as NWP it is very computationally costly, with the number of particles required increasing exponentially as the dimension of the state grows. Vaswani [2008] proposed a scheme that would reduce this computational cost by MT part of the state, this was discussed in section 2.4.

In Chapter 4.1 we considered results obtained from the PF. If too few particles were used then a solution could be obtained that differed significantly from the truth. Once the number of particles was increased we obtained solutions closer to the truth, we also saw that the x, y

and z plots showed phase errors in the solutions. As the number of particles increased we saw that the solution improved. This was seen in both the qualitative and quantitative results and is supported by Pham [2000] and Vaswani [2008] who see the RMSE reduce as the number of particles is increased.

When considering the RMSE we discovered that due to the nonlinearity of the Lorenz system the errors obtained were not smooth, this behavior was also seen in the RMSE obtained by Pham [2000]. We saw that large errors were often associated with the strongly nonlinear regions of phase space [Pham, 2000]. We also suggested that some of the sudden jumps in the RMSE were due to the resampling of the particles. Another quantitative result that was used to determine the accuracy of the PF runs was the RH [Hamill, 2000]. We saw that when all the particles had equal weights a reliable forecast with no errors in its mean and spread should produce a uniform RH. However due to the weighting of particles in the PF we used ensemble means to determine any differences we expected. The RH showed that for small N there was a lack of variability in the ensemble, increasing the value of N increased this variability.

In Chapter 6 we considered the results obtained from the PFMT code and compared these to the PF results. When considering how to split the state we saw that the best results were obtained when the x and y dimensions we mode tracked, whereas the z dimension gave the worst results. This was explained by the model equations and is discussed in section 6.1. It was also apparent that MT two dimensions did not always give better results than MT one dimension. This was likely to be due to the multi-modal behavior of one or more of the dimensions.

When the number of particles used with the PFMT was increased the accuracy of the solution also increased. However a comparison of the PF and PFMT results showed that with small values of N the PFMT produced less accurate results than the PF. Even when the number of particles was increased to $N = 500$ the PFMT did not alway perform better than the PF. This was contrary to the results seen in Vaswani [2008] where the PFMT always produced superior results to the PF.

Finally we considered the sensitivity of the PFMT to the ratio between the model and observation errors. We knew as we were reducing the error variance of the observations the

result would improve, but the improvement seen was more significant than expected. This sensitivity of the model is supported by Johnson et al. [2005], who show that the correct ratio is required to obtain the most accurate solution. It was also mentioned that the choice of noise parameter B, discussed in section 3.6, affected the ratio between the model and observation errors as the model noise is scaled by the parameter B.

7.2 Further Work

We consider three areas identified throughout the thesis that may benefit from further investigation. These include overcoming the limitations of thie work by reducing the number of observations available for use in the PF and PFMT, resampling less often and testing more than one realisation of the model. Using reliability diagrams instead of RH and further investigation into the sensitivity of the Q matrix.

7.2.1 Overcoming Limitations of this work

In section 4.1 we saw that we started the code by creating a truth run, then perturbing these values to give observations. When the main PF iteration was run the forecast values were compared to observations at every timestep. In reality the observations do not occur this often, and even when the observations do occur we may not have an observation for each of the dimensions that needs to be forecast. For this reason it would be interesting to investigate the performance of the PF and PFMT when sparse and incomplete observational data is used. Reducing the number of observations to say one every ten timesteps should be fairly simple, as it requires only a slight adjustment to the code. Instead of forecasting the the particles one timestep using the EM approximation we would need to forecast them say ten timesteps. At this point they could then be compared to the observation, weighted and resampled. The idea of making the observational data incomplete is slightly more complex and would require a greater adjustment of the code.

A second limitation of the code is the inclusion of the resampling at each timestep. This increases the computational cost of the scheme and leads to a lack of variability in the particles.

It would be interesting to see the effect of remove the resampling from each timestep and only include it when necessary.

One further limitation of the experiments seen in Chapters 5 and 6 is that they have only been run for one realisation of the model. It would be beneficial to run the code for different realisations to check that similar results are obtained.

7.2.2 Considering the use of Reliability Diagrams

In section 4.3.2 we used RH to asses the quality of our solutions. However we found that they were not ideal as they assumed equally weighted particles and no resampling. We mentioned that it may be more appropriate to use a reliability diagram [Brocker and Smith, 2007], [Wilkes, 1995] to tell us about the quality of our forecast. The reliability diagram takes the weighting of the particles into account and would therefore provide us with a more accurate diagnostic to allow us to determine the reliability and quality of the forecast.

7.2.3 Investigating the Q matrix sensitivity

In section 6.3 we showed that by reducing the difference between the background error variance and the observation variance we greatly increased the accuracy of the PFMT solution. It would be of interest to see if the improvement in the solution was as great if the same experiment was carried out for the PF. This would allow us to know if this decrease in error was more advantageous for the PFMT. However setting the variances of the background and observation equal was not the only result concerning the Q matrix that improved the solutions. When the code was tested it was discovered that decoupling the Q matrix from the B parameter also increased the accuracy of the solution. Unfortunately there was not time to pursue this discovery, but it would be interesting and maybe important to know why this decoupling increases the accuracy of the solution.

Appendix A

This appendix is included to show that if the resampling step is removed from the code discussed in section 4.1 we obtain results similar to those seen in Snyder et al. [2008]. In section 2.3.3 we saw that Snyder et al. [2008] showed that over time without resampling all the weight ended up on one particle. As we know this problem exists we would like to check that the code we are using produces similar results. Figure 7.1 shows a histogram of maximum weights at each iteration. The graph is generated from code that is discussed in detail in section 4.1, we do not include the resampling step as we wish to see how the weights behave without this.

From Figure 7.1 we see that we also have a high realisation of maximum weights in the interval $(0.98, 1)$ suggesting that over time the weight goes onto one particle. However unlike Figure 2.6 we also see a high realisation of maximum weights in the interval $(0.2, 0.22)$. As there are $N = 5$ particles initially we expect each particle to have a weight of $w^i = 0.2$, it is likely that all the particles stay close to true solution for a short amount of time. During this time we would expect the weights to remain fairly equal, hence giving us a high realisation of maximum weights in the $(0.2, 0.22)$ interval. We have now shown that without the resampling step the weights behave as expected.

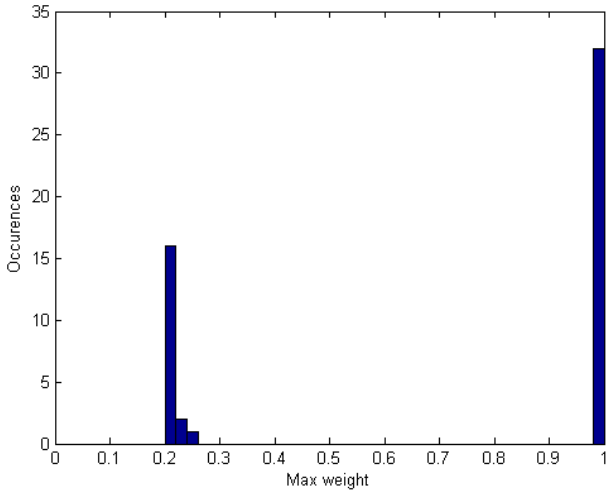

Figure 7.1: How weight goes onto one particle over time. Code parameters: $N = 5$ particles. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$.

Appendix B

This appendix includes a number of plots that support the work, seen in section 6, on splitting the state. Each plot shows the PFMT solutions to the Lorenz equations with different parts of the states mode tracked. These solutions are obtained from code run with the following parameters: Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$. $N = 500$ particles.

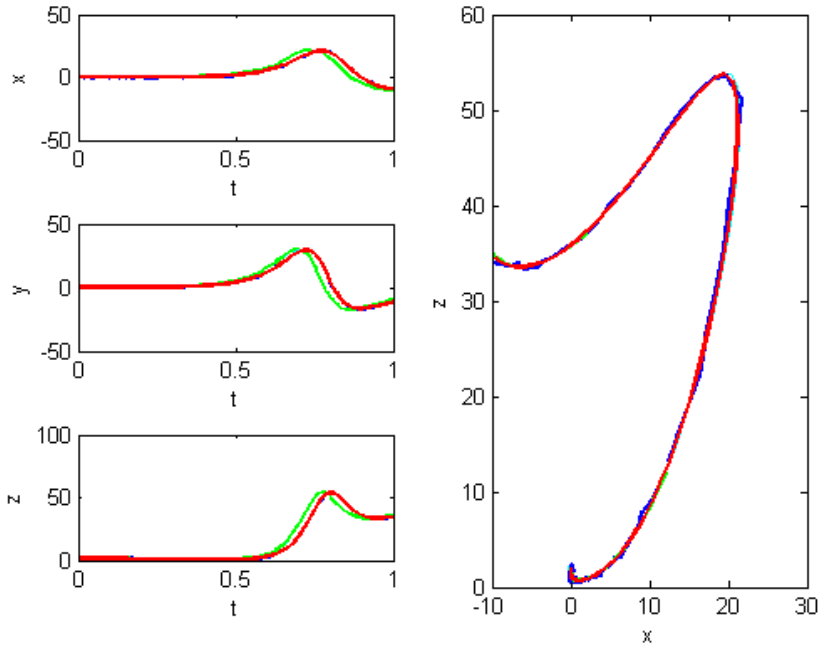

Figure 7.2: PFMT solution of the Lorenz equations with $\Psi_r = (x)$ and $\Psi_s = (y, z)$. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$. $N = 500$ particles.

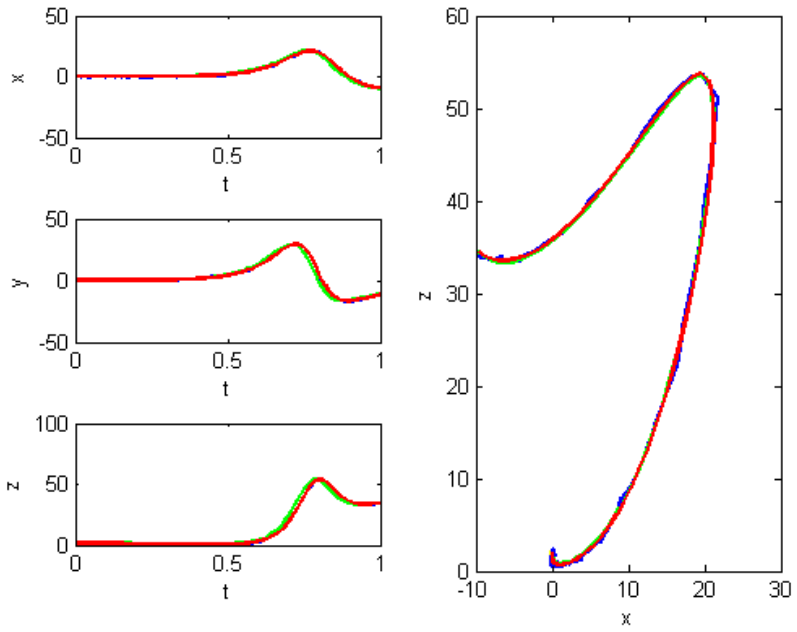

Figure 7.3: PFMT solution of the Lorenz equations with $\Psi_r = (y)$ and $\Psi_s = (x, z)$. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$. $N = 500$ particles.

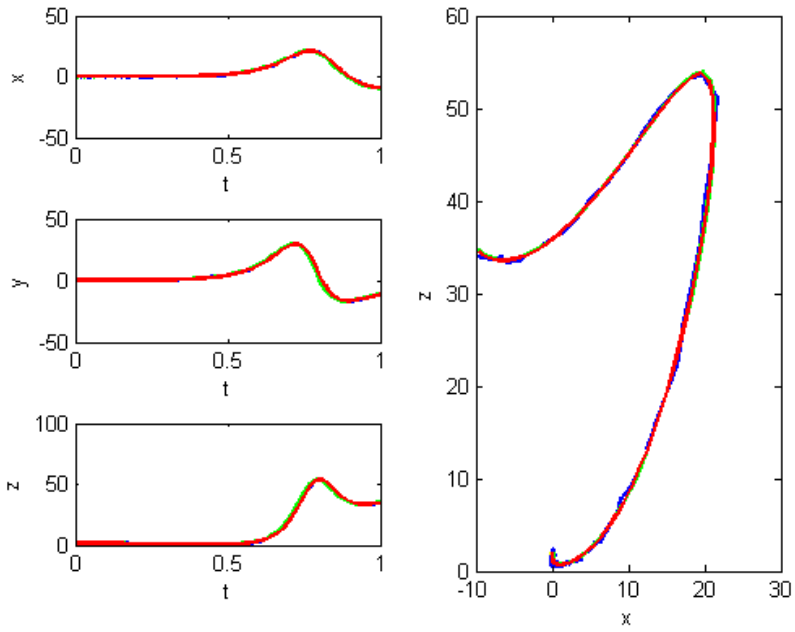

Figure 7.4: PFMT solution of the Lorenz equations with $\Psi_r = (x, z)$ and $\Psi_s = (y)$. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$. $N = 500$ particles.

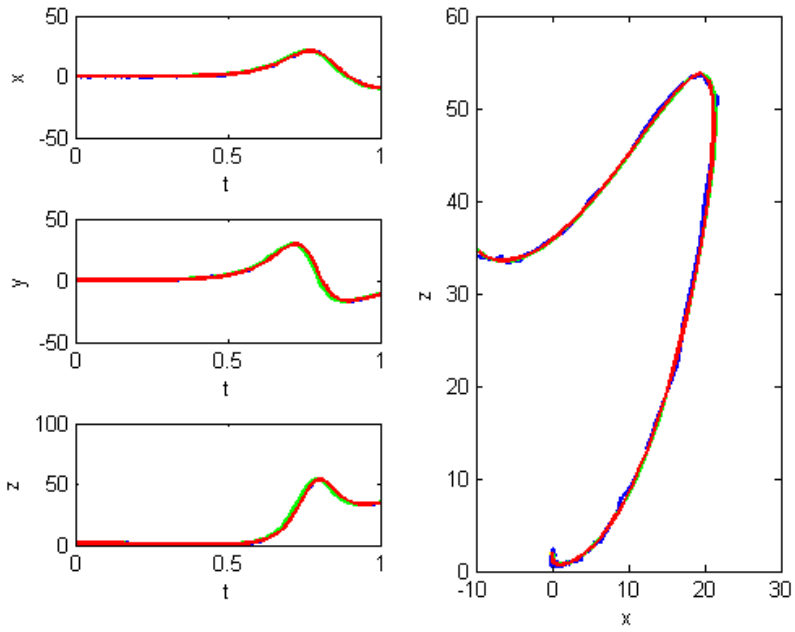

Figure 7.5: PFMT solution of the Lorenz equations with $\Psi_r = (y, z)$ and $\Psi_s = (x)$. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$. $N = 500$ particles.

Appendix C

This appendix includes a plot that supports the work in section 6.3.1. The plot shows the PFMT solutions to the stochastic Lorenz equations with observation error variance = 0.01. This solution is obtained from code run with the following parameters: State splitting $\Psi_r = (x, y)$ and $\Psi_s = (z)$. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$. $N = 50$ particles.

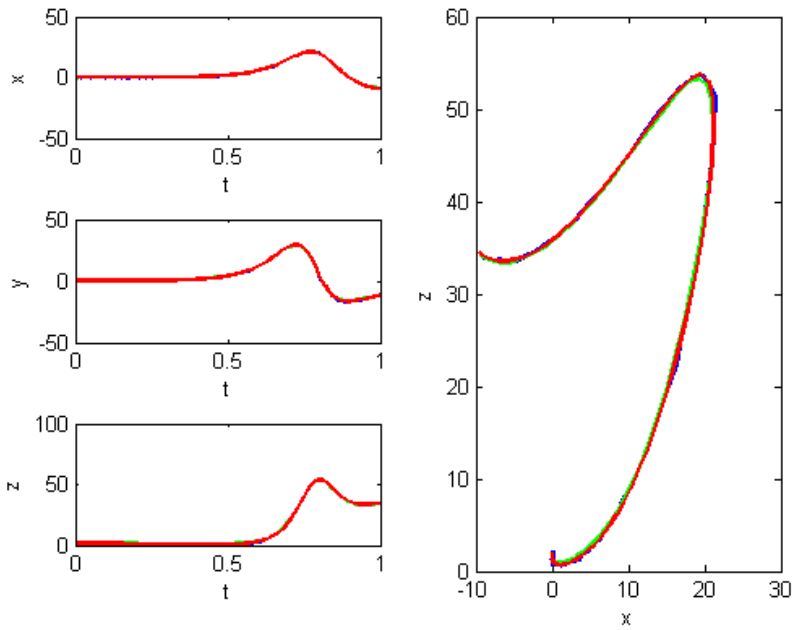

Figure 7.6: PFMT solution of the Lorenz equations with $\Psi_r = (x, y)$ and $\Psi_s = (z)$. Colours as in Figure 5.1. Initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $\Delta = 0.01$. Noise parameter $B = 0.1$. $N = 50$ particles. Observation error variance = 0.01

Bibliography

U. M. Ascher and L. R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*, chapter 3, pages 37 – 39. SIAM, 1998.

R. Bannister. http://www.met.reading.ac.uk/ ross/documents/oxral04.html, Last accessed 19^{th} August 2009.

T. Bengtsson, C. Snyder, and D. Nychka. Toward a nonlinear ensemble filter for high-dimensional systems. *Journal of Geophysical Reasearch*, 108, 2003.

J. Brocker and L. A. Smith. Increasing the reliability of reliability diagrams. *Weather and Forecasting*, pages 651 – 661, June 2007.

R. Douc, O. Cappe, and E. Moulines. Comparison of resampling schemes for particle filtering. *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 64–69, 2005.

A. Doucet, de Freitas N., and N. Gordon. *Sequential Monte Carlo Methods in Practice*, chapter 1. Springer, 2000a.

A. Doucet, S. Godshill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Satistics and computing*, pages 197–208, 2000b.

G. Evensen. *Data assimilation: the ensemble Kalman filter*. Springer-VerlagBerlin and Heidelberg, 2006.

D. Fox. Adapting the sample size in particle filters through kld-sampling. *The Inernational Journal of Robotics Reaserch*, 22:985–1003, 2003.

T. M. Hamill. Interpretation of rank histograms for varifying ensemble forecasts. *Monthly Weather Review*, 129:550–560, 2000.

C. Johnson, N. K. Nichols, and B. J. Hoskins. Very large inverse problems in atmosphere and ocean modelling. *International Journal for Numerical Methods in Fluids*, 47:759–771, March 2005.

E. Kalnay. *Atmospheric Modeling, Data Assimilation and Predictability.* Cambridge University Press, 2002.

G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5:1–25, 1996.

W. Kliemann and N. S. Namachchivaya. *Nonlinear dynamics and stochastic mechanics*, chapter 18.3, pages 439 –440. CRC Press, 1995.

P. E. Kloden and E. Platen. *Numerical Solution of Stochastic Differential Equations.* Springer, 1999.

H. Lee and W. Schiesser. *Ordinary and Partial Differential Equation Routines in C, C++, Fortran, Java, Maple, and MATLAB.* Chapman and Hall/CRC, 2003.

E. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–139, 1963.

C. Moler. *Numerical Computing with MATLAB*, chapter 9. SIAM, 2006.

N. K. Nichols. Data assimilation: Aims and basic concepts. In Data Assimilation for the Earth System, 2003.

D. T. Pham. Stochastic methods for sequential data assimilation in strongly nonlinear systems. *Monthly Weather Review*, 129:1194–1207, 2000.

J. Pocock. Computer techniques and projects, coursework 1, December 2008.

Y. Saito and T. Mitsui. Stability analysis of numerical schemes for stochastic differential equations. *SIAM Journal of Numerical Analysis*, 33(6):2254–2267, December 1996.

C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson. Obsticals to high dimensional particle filtering. *Monthly Weather Review*, pages 4629–4640, 2008.

R. Swinbank, V. Shutyaev, and W. A. Lahoz. *Data Assimilation for the Earth System*. Kluwer Academic Publishers, 2003.

UKMO. http://www.metoffice.gov.uk/science/creating/daysahead/nwp/index.html, Last accessed 19^{th} August 2009.

P. J. van Leeuwen. Particle filtering in geophysical systems. To be publisehed in Monthly Weather Review, 2009.

N. Vaswani. Particle filtering for large-dimensional state spaces with multimodal observation likelihoods. *IEEE Transactions on Signal Processing*, 56:4583–4597, 2008.

D. S. Wilkes. *Statisical Methods in the Atmospheric Sciences*, chapter 7, pages 265–267. Academic Press, 1995.

Y. Xiao, W. Xu, T. S., and X. Li. Adaptive complete synchronization of the noise-perturbed two bi-directionally coupled chaotic systems with time-delay and unknown parametric mismatch. *Journal of Applied Mathematics and Computation*, 231:538–547, 2009.

Declaration

I confirm that this is my own work, and the use of all material from other sources has been properly and fully acknowledged.

Joanne A. Pocock

Acknowledgments

Firstly I would like to thank my supervisor Dr. Sarah Dance for all her enthusiasm, help and support given whilst completing this project. I would also like to thank all those who have made this year enjoyable and possible. Finally I would like to thank NERC for their generous funding.