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 ful lment 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 vi

List of Tables vii

List of Symbols viii

List of Abbreviations xi

1 Introduction 1

1.1 Background 1

1.2 Aims 3

1.3 Principal Results 3

1.4 Outline 4

2 Data Assimilation 5

2.1 Data Assimilation 5

2.2 Ensemble Data Assimilation 7

2.3 The Particle Filter 8

2.3.1 The Prior and Sampling From It 10

2.3.2 Sequential Importance Sampling 10

2.3.2.1 The Forecast Step and Bayes Theorem 11

2.3.2.2 Weighting 12

2.3.3 Problems with Weighting 14

2.3.4 Resampling 15
2.3.4.1 Strati ed Resampling 17
2.3.5 Summarising the Particle Filter 20
2.3.6 Properties of the Particle Filter 21
2.4 The Particle Filter with Mode Tracking 21
2.4.1 Initialisation and Importance sampling 23
2.4.2 Mode Tracking 23
2.4.3 Weighting and Resampling 24
2.5 Summary 24

3 The Model 26
3.1 The Lorenz Model 26
3.2 The Euler-Maruyama Approximation 27
3.3 Convergence and Stability 28
3.3.1 Convergence of the EM Scheme 28
3.3.2 Stability of the EM Scheme 29
3.4 The Euler-Maruyama Code 31
3.5 Model Validation 31
3.5.1 Qualitative Results 31
3.5.2 Quantitative Results 35
3.6 Sensitivity to Noise Parameters 38
3.7 Summary 39

4 Implementation and Diagnostics for the PF and PFMT 40
4.1 The Particle Filter Code 40
4.1.1 Initialisation 40
4.1.2 The Particle Filter Iteration 41
4.2 The PFMT code 41
4.3 Diagnostics 43
4.3.1 Root Mean Squared Error 43

4.3.2 Rank Histograms . . . 43
4.3.2.1 Creating the Rank Histograms . . . 44
4.3.2.2 Interpreting Rank Histograms . . . 44
4.4 Summary . . . 45

5 The Particle Filter Solutions 46
5.1 Lorenz Trajectory solutions . . . 46
5.1.1 Results for $N = 2$ . . . 47
5.1.2 Results for $N = 5$ . . . 48
5.1.3 Results for $N = 50$ . . . 49
5.1.4 Results for $N = 500$ . . . 50
5.2 Quantitative Results . . . 51
5.2.1 Root Mean Squared Error . . . 51
5.2.2 Rank Histograms . . . 53
5.3 Summary . . . 57

6 Particle Filters with Mode Tracking 58
6.1 Splitting the State . . . 58
6.1.1 Qualitative Results . . . 58
6.1.2 Quantitative Results . . . 60
6.2 Changing the number of particles . . . 62
6.2.1 Qualitative Results . . . 62
6.2.2 Quantitative Results . . . 64
6.3 Sensitivity of MT to error variances . . . 65
6.3.1 Qualitative Results . . . 66
6.3.2 Quantitative Results . . . 66
6.4 Summary . . . 67

7 Conclusion 69
7.1 Summary and Discussion . . . 69

7.2 Further Work 71

List of Figures

2.1 A schematic of the sequential DA scheme 6
2.2 Schematic of ensemble DA 7
2.3 A schematic of one iteration of the particle filter process. 9
2.4 Map of the corridor used for the robot experiment 13
2.5 Figures showing the position of the robot 13
2.6 A plot of the number of occurrences of the maximum weights in particular intervals. Figure from Snyder et al. [2008] 15
2.7 Figures showing the position of the robot 16
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] 17
2.9 Number of Particles Required required 22

3.1 The stability region of Euler's Method 30
3.2 Solution to stochastic Lorenz equation with no added noise. 32
3.3 EM solution to stochastic Lorenz equations. 33
3.4 Two differing EM solutions of the Lorenz equations from identical initial conditions 34
3.5 Two similar EM solutions of the Lorenz equations from identical initial conditions 35
3.6 Graph showing how the size of the timestep affects the error 37
3.7 Affect on solution of changing B 38

4.1 Common rank histograms obtained. 44

List of Tables

2.1 A simple DA algorithm. 6
2.2 The sequential importance sampling algorithm. 11
2.3 The Strati ed Resampling Algorithm. 18
2.4 The Bootstrap Filter 20
2.5 The PFMT Algorithm. 22

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

	Order of weak convergence
	Lorenz parameter
	Timestep size
(superscript)	Timestep size
	Error
ε_t	Observation error
	Order of strong convergence
	Model error

	Complex number
	Lorenz parameter
	Lorenz parameter
	State vector
	State vector
$\tilde{}$	Numerical approximaion
$_r$	Part of the state to be mode tracked
$_s$	Part of the state not to be mode tracked
$!$	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 di erences in the forecast. We must

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.

shos(Some)-2T1(results)2321(from)-2TTJ-0-Hz0Ehesults shomediscusstsme Chaptel

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 lters (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 rst consider the linear dynamical system,

$$ _{t+1} = F \ _t + \ _t; \tag{2.1} $$

where $ _t$ is the state vector of length n at time t, F is a constant matrix of size $n \times n$ and $ _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,

$$ d_t = H \ _t + "_t; \tag{2.2} $$

where d_t is the observation vector of length p at time t, $"_t$ 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 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.

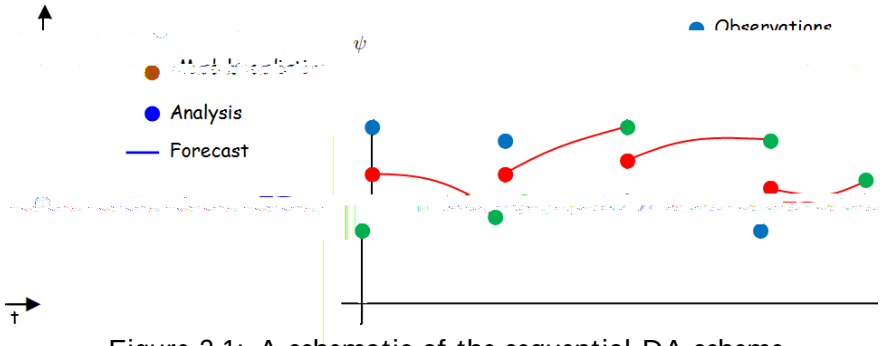


Figure 2.1: A schematic of the sequential DA scheme

Figure 2.1 shows that DA works by taking d_t, an observation at time t and a model prediction, $\overset{b}{t}$, known as the background. $\overset{b}{t}$ and d_t are combined to produce an analysis $\overset{a}{t}$, 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: $\overset{a}{t+1} = \overset{b}{t+1} + K(d_{t+1} - H \overset{b}{t+1})$, 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 $\overset{b}{t+1} = F \overset{a}{t} +$ 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 ^{a} and ^{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 di erent 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

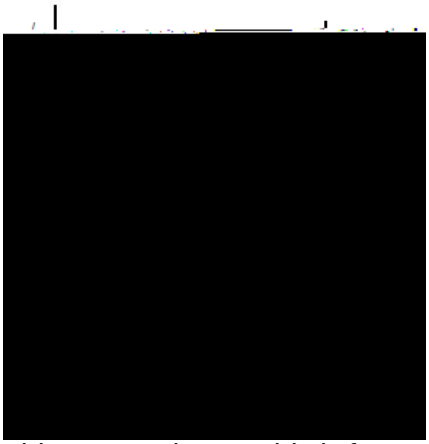
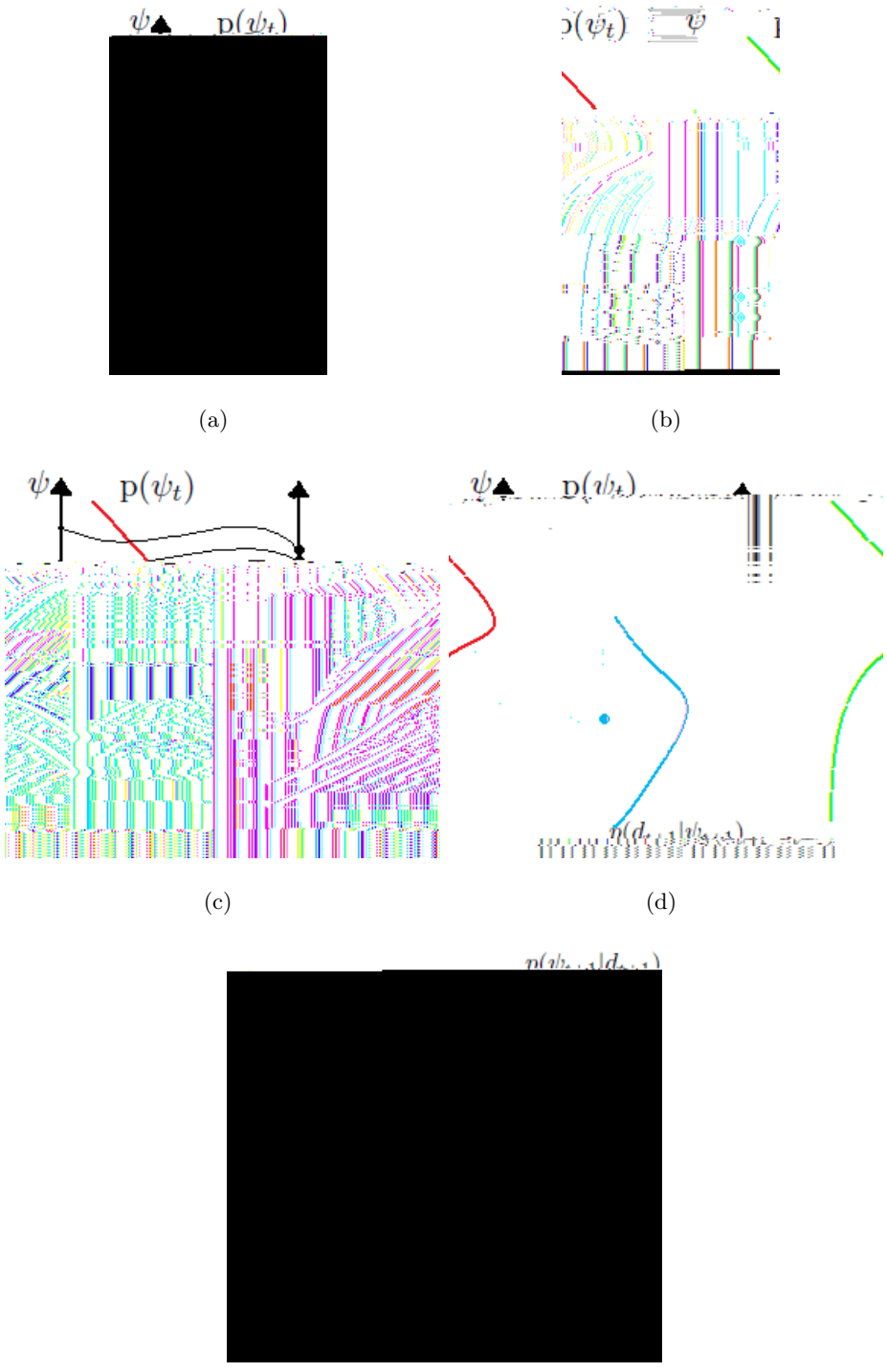


Figure 2.2: Schematic of ensemble DA: Each ensemble is forecast from slightly di erent initial conditions. The red line represents the initial pdf. The grey lines are the trajectories of the ensembles sampled from the initial pdf, the black line is the ensemble mean trajectory. The blue line represents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



(a) (b) (c) (d) (e)

Figure 2.3: A schematic of one iteration of the particle Iter 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, $p(\ _0)$. Here $\ _0$ 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 di cult to store all the information associated with $p(\ _t)$. 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 $p(\ )$. The mean of a function, f, of the state is given by,

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

However this quantity may still be di cult 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, $\{\ ^i\}_{i=1}^N \sim p(\ _t)$. This allows us to represent $p(\ )$ as

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

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

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

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 de ne the notation seen in Table 2.2, and later in Table 2.4, as follows. $\ _{0:k} = \{\ _0;:::;\ _t\}$ represents the model states up to time t. $d_{0:k} = \{d_0;:::;d_t\}$ represents the observations up to time t. $(\ _{0:k}|d_{1;k})$,

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

For times $k = 0; 1; 2; ::::$

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

$$w^i_k = w^i_{k-1} \frac{p(d_k|\ ^i_k)p(\ ^i_k|\ ^i_{k-1})}{(\ _k|\ ^i_{0:k-1}; d_{0:k})} \tag{2.6}$$

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

$$!^i_k = \frac{w^i_k}{\sum_{j=1}^N w^j_k} \tag{2.7}$$

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 describ1 Tf3937.9796nsk

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

Theorem 1 Bayes Theorem

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

So if 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,

$$p(\ _t|d_t) = \sum_{i=1}^{N} w^i\ (\ _t - \ _t^i); \tag{2.9}$$

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, $\ _t^i$, 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 $\ _t^i$.

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

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 nd that some of the forecast values do not t 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

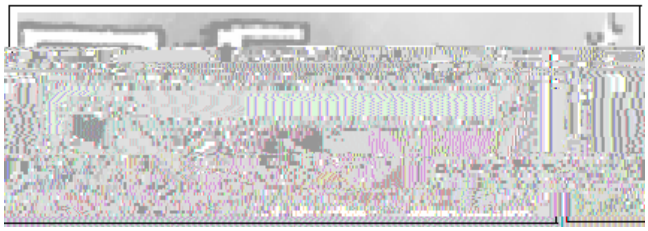


Figure 2.4: Map of the corridor used for the robot experiment. The white areas represent the corridors where the robot can walk. Figure from Fox [2003].

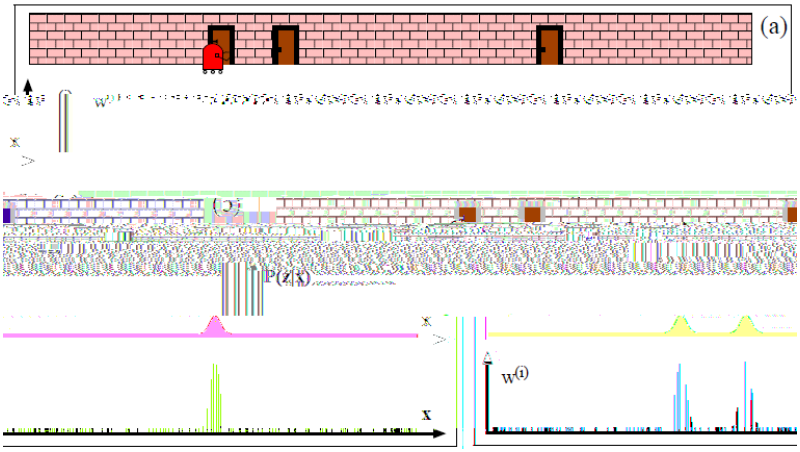
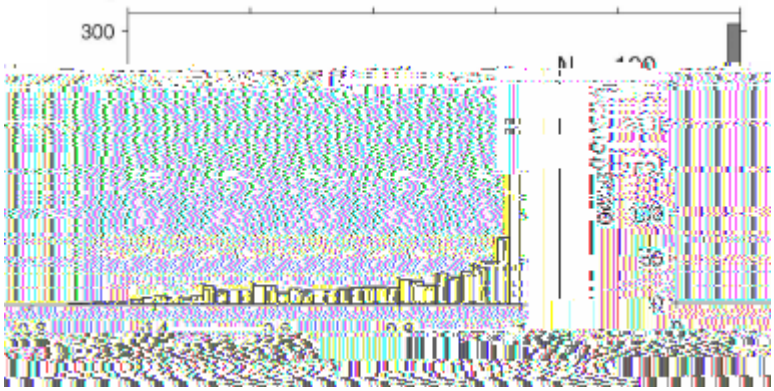


Figure 2.5: Figures showing the position of the robot, graphs of the associated probability of his position and graphs of the particles representing the possible location of the robot. The top of both plots (a) and (b) show the location of the robot relative to the doors. The x vs w plots in (a) and (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 , the state and z is the observation. The remaining plot in (b) shows the observation likelihood $p(d|$). Figure from Fox [2003]

between each door. The rst step in in the particle lter 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


300

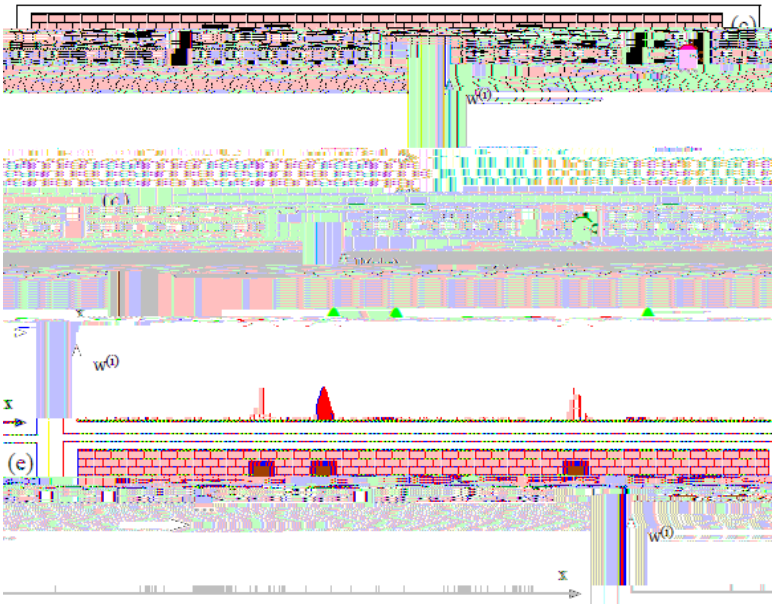


Figure 2.7: Figures showing the position of the robot, graphs of the associated probability of his position and graphs of the particles representing the posible location of the robot. Notation and plots as in Figure 2.5. Figure from Fox [2003]

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

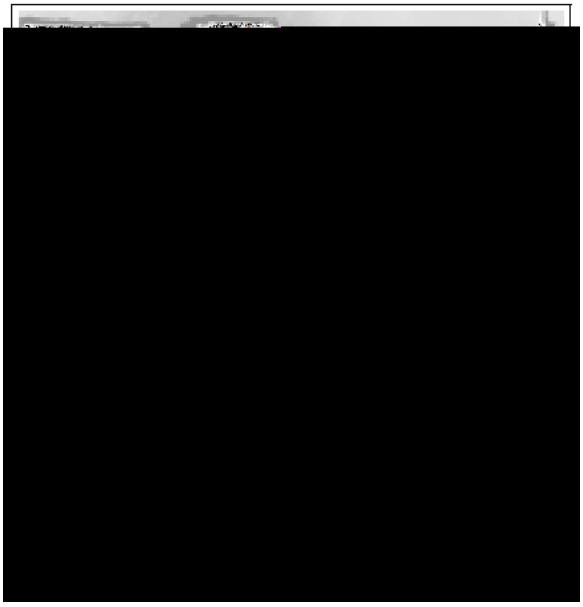


Figure 2.8: Maps showing the possible location of the robot after he has moved 5m (a), more than 5m (b), 55(m). Figure from Fox [2003]

2.3.4.1 Strati ed Resampling

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

The Strati ed Resampling Algorithm [Kitagawa, 1996]
1. Rearrange the normalised weights $!^1;::::;!^N$ in ascending order. The reordered weights are expressed as $\dagger^1;::::;\dagger^N$. 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 $i = 1$, $j = 1$, $!^L = 0$ and $!^R = \dagger_1$. 5. Test $!^L < u_j \leq !^R$ If this inequality holds then accept the particle associated with $\dagger^i$ and set $j = j + 1$. If the inequality does not hold set $!^L = \sum_{l=1}^{i} \dagger^l$, $!^L = \sum_{l=1}^{i+1} \dagger^l$ 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 Strati ed Resampling Algorithm.

+ We nd [(+)-ng8 cm []0 d 0 J 0.398 w 0 0 m 0 20.324 l J/F22 10.9091 Tf 21.212 0 Td [(j)]TJ/F15 10.

Test $!^L < u_j \leq !^R$ *where* $u_1 = 0{:}1$, $!^L = 0$ *and* $!^R = 0{:}05$.

0

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

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

This resampling is the nal 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 nal 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 $i = 1;:::;N$, sample $\overset{i}{0}\ p(\ _0)$ and set $t = 1$.
2. Importance Sampling Step
For $i = 1;:::;N$, sample $\mathrm{e}^i_t\ p(\ _t\|\ \overset{i}{}_{t-1})$ and set
$\mathrm{e}^i_{0:t} = (\mathrm{e}^i_{0:t-1};\ \mathrm{e}^i_t)$.
For $i = 1;:::;N$, evaluate the importance weights,
$\mathrm{w}^i_t = p(d_t\|\ \mathrm{e}^i_t)$.
3. Normalise the importance weights.
4. Selection Step
Resample with replacement N particles $(\ \overset{i}{}_{0:t}; i = 1;:::;N)$
from the set $(\mathrm{e}^i_{0:t}; i = 1;:::;N)$ 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 di er signi cantly 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 lter would do. The only di culty 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 $= [\ _s;\ _r]$. We treat the $_s$

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 $\alpha_{t;s}$: $\forall i$ sample $\alpha^i_{t;s}$ $p(\alpha^i_{t;s} \mid \alpha^i_{t-1})$.
2. Mode track $\alpha_{t;r}$: $\forall i$ set $\alpha^i_{t;r} = m^i_t$ where
$m^i_t(\alpha^i_{t-1}; \alpha^i_{t;s}; d_t) = \arg\min_{\alpha_{t;r}} [-\log p(d_t \mid \alpha^i_{t;s}) p(\alpha_{t;r} \mid \alpha^i_{t-1}; \alpha^i_{t;s})]$.
3. Weight: $\forall i$ compute $w^i_t = \frac{!^i_t}{\sum_{j=1}^{N} !^j_t}$ where $\alpha^i_t = [\alpha^i_{t;s}; \alpha^i_{t;r}]$.
and $!^i_t = w^i_{t-1} p(d_t \mid \alpha^i_t) p(\alpha^i_{t;r} \mid \alpha^i_{t-1}; \alpha^i_{t;s})$.
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 $\grave{}_{t;s}$ contains the minimum number of dimensions ... [such that] $p(\ _{t;r}|\ ^{i}_{t-1};\ ^{i}_{t;s};d_t)$ [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 $_s$ dimensions in this way. Here we sample from the prior pdf and forecast the $_s$ dimensions one timestep.

2.4.2 Mode Tracking

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

$$J^i(\ ^{i}_{t;s};\ _{t;r}) = \frac{1}{2}(J_o(\ ^{i}_{t;s};\ _{t;r}) + J_q(\ _{t;r})); \tag{2.12}$$

2.4.3 Weighting and Resampling

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

$$w_t^i = \frac{! \, {}^i_t}{\sum_{j=1}^{N} ! \, {}^j_t} \tag{2.15}$$

where $! \, {}^i_t = ! \, {}^i_{t-1} p(d_t | \ {}^i_t) p(\ {}^i_{t;r} | \ {}^i_{t-1}; \ {}^i_{t;s})$. 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 lter. We resample the particles using the same strati ed 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 nal forecast time is reached.

Vaswani [2008] found that the introduction of MT signi cantly 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

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]

where x, y and z are the dimensions of the state, t is time and , and are constants. The constants , and may take any value greater than zero, however we have chosen = 10, = 28 and = $\frac{8}{3}$ as these are the classic Lorenz parameters [Lorenz, 1963]. Choosing these

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

$$ _{t+1} = \ _t + a(\ _t)\ _t + b(\ _t)\ W_t; \tag{3.8} $$

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

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

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

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

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 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 de ne 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.

De nition 1 Strong Convergence

If ^T_t *is the true value of* *at time t,* $\tilde{}_t$ *is an approximation of* *obtained from a numerical*

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

$$E(|\ {}^{T}_{t} - \tilde{}_{t}\ |) \leq G\ \Delta^{\gamma}$$

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

De nition 2 Weak Convergence

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

$$|E(g({}^{T}_{t})) - E(g(\tilde{}_{t}))| \leq G_g\ \Delta^{\gamma}$$

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 $\gamma = 1$

Theorem 4 Convergence for one timestep of Euler's method

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

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

$$d\ _{t} = \ _{t}dt + dW_t; \tag{3.12}$$

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

$$_{n+1} = \ _{n}(1 + \ \ ) + \ \ W_n: \tag{3.13}$$

From this we obtain,

$$|\ _{n} - \tilde{}_{n}| \leq |1 + \ \ |^{n}|\ _{0} - \tilde{}_{0}|; \tag{3.14}$$

where $\tilde{}_{n}$ is the solution obtained from starting at $\tilde{}_{0}$. 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.

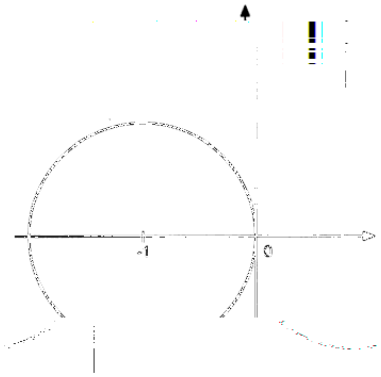


Figure 3.1: The stability region of Euler's Method. The stable region lies within the circle on the complex plane.

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 nd a stable timestep to use. Xiao et al. [2009] use the Lorenz equations with parameters = 10, = 30 and = $\frac{8}{3}$, the only parameters that di er are and the noise parameter we have called B. To solve the equations Xiao et al. [2009] use the EM approximation and a timestep of = 0:01. In our experiments we have also used a timestep of = 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 W_X, W_Y and W_Z are independent and from the distribution $N(0; \ _t)$. 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 $N(0; \ _n)$ the values of W_X, W_Y and W_Z are calculated scaling the output of *randn* by $\sqrt{\ }$, (i.e $\sqrt{\ }N(0; 1)$).

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 butter y plot produced by the Lorenz Equations.

To gain an initial solution we run the Euler-Maruyama scheme from time $t = 0$ until t

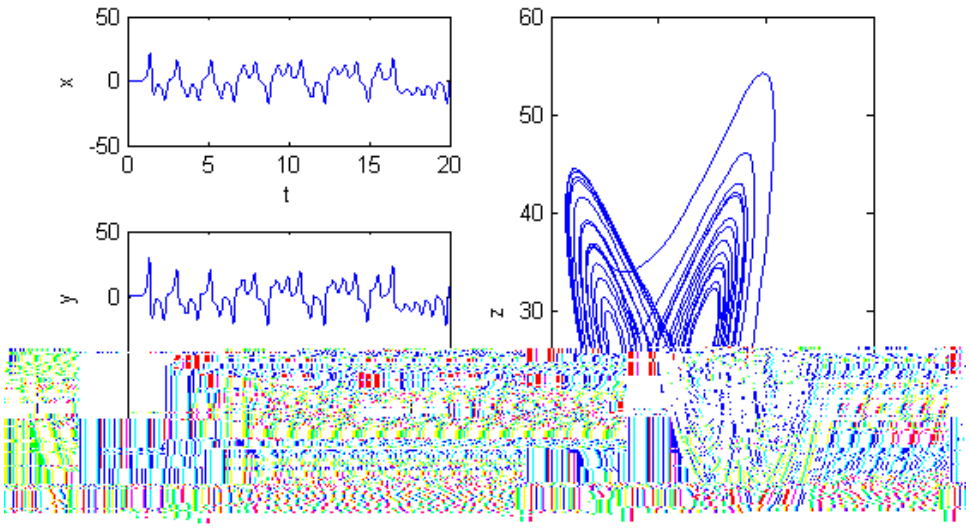


Figure 3.2: Solution to stochastic Lorenz equation with no noise added. Initial conditions $x = 0{:}00001$, $y = 0{:}00001$, $z = 2{:}00001$. From time $t = 0$ to $t = 20$ with a timestep $= 0{:}01$.

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

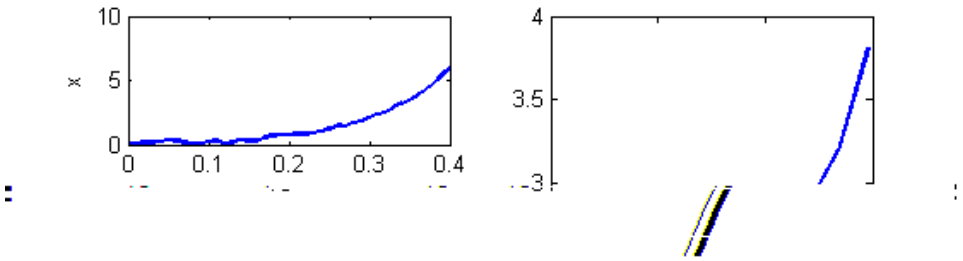


Figure 3.3: Solution to stochastic Lorenz equation. Initial conditions $x = 0{:}00001$, $y = 0{:}00001$, $z = 2{:}00001$. From time $t = 0$ to $t = 0{:}4$ with a timestep $= 0{:}01$. Noise parameters $B_X = 1$, $B_Y = 1$ and $B_Z = 1$.

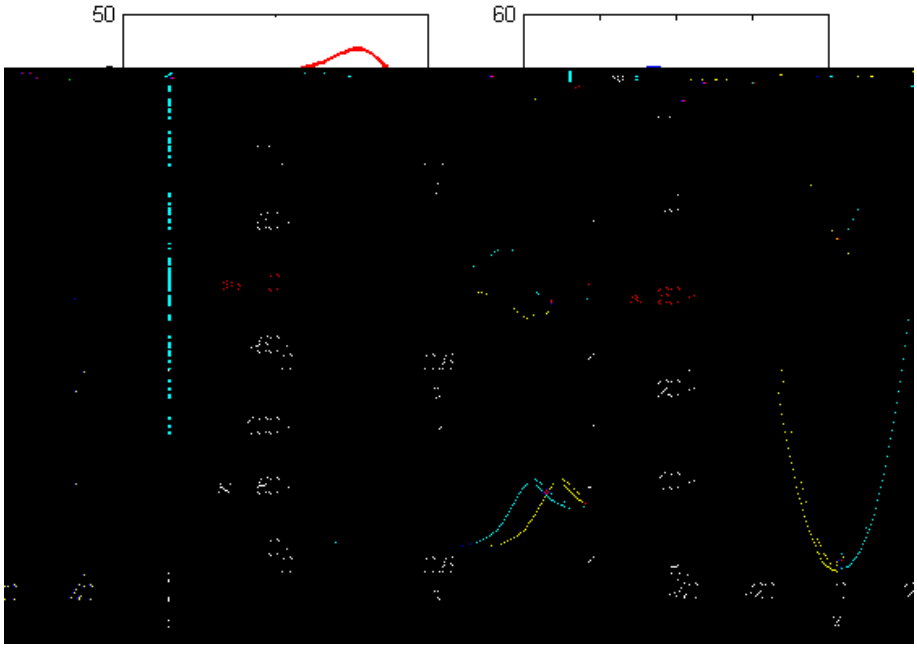


Figure 3.4: Two solutions from identical initial conditions. The red and blue lines represent two solutions from initial conditions $x = 10{:}00001$, $y = 10{:}00001$, $z = 20{:}00001$ but with di erent realisations of noise. From time $t = 0$ to $t = 1$ with a timestep $= 0{:}01$. Noise parameters $B = 0{:}1$.

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

The location of the initial conditions in phase space also helps to determine how quickly two solutions with di erent initial conditions di er 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 butter y 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.

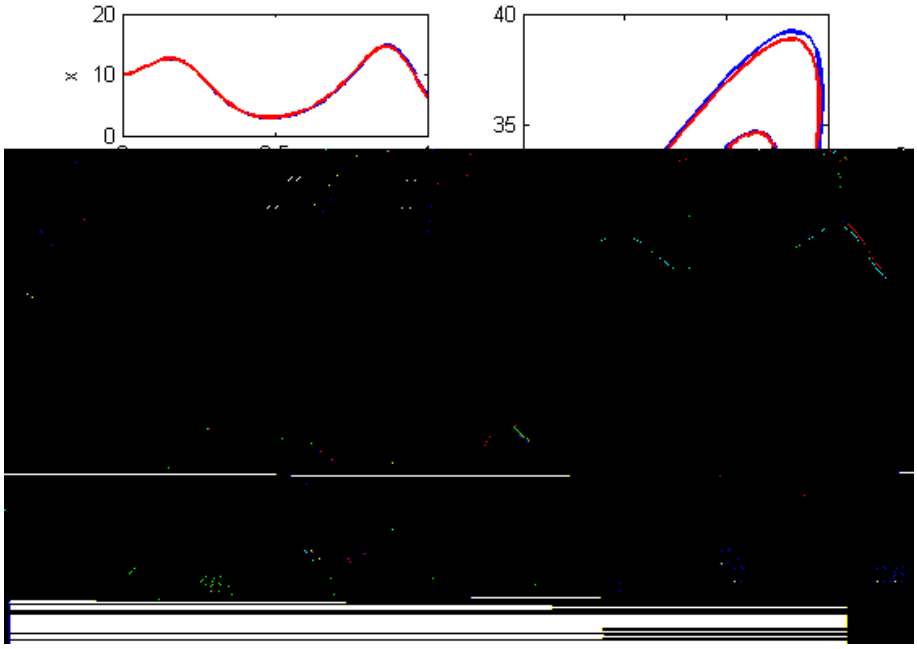


Figure 3.5: Two solutions from identical initial conditions. The red and blue lines represent two solutions from initial conditions $x = 10{:}00001$, $y = 10{:}00001$, $z = 20{:}00001$ but with di erent realisations of noise. From time $t = 0$ to $t = 1$ with a timestep $= 0{:}01$. Noise parameters $B = 0{:}1$.

3.5.2 Quantitative Results

We have now seen that the Euler-Maruyama scheme provides the expected qualitative results, producing the classic butter y plot we expect from the Lorenz attractor and showing the chaotic nature of the system by giving di ering 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 $= 0{:}5$ and weak order $= 1$. This suggests that a plot of log( ) 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 $B = 0$ simpli es 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(^{2}). Therefore if we run our scheme with no noise we expect the plot of log( ) vs. log(*Error*) to show a straight line with gradient two.

To be able to plot the log( ) 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)

$$_t - \tilde{}_t = \quad ; \tag{3.15}$$

where is the error obtained from the di erence between is the true solution at time t and $\tilde{}_t$ the numerical approximation at the same time calculated with timestep . As we do not have an analytic solution we cannot do this directly. We expect our error $= O(^{0:5})$, so we would expect equation (3.15) to be,

$$_t - \tilde{}_t = O(^{0:5}): \tag{3.16}$$

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

$$_t - \tilde{}_t = O(^{0:5}): \tag{3.17}$$

$$_t - \tilde{}^2_t = O((2)^{0:5}): \tag{3.18}$$

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

$$\tilde{}_t - \tilde{}^2_t = O((2)^{0:5}) - O(^{0:5}) = O(^{0:5}): \tag{3.19}$$

Therefore to calculate the error at time we calculate the numerical solution using and 2 we then subtract one from the other. The resulting value is the error associated with the timestep . We carry out these calculations for 5 di erent '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( ) 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 is plotted as a cross, we then t a line of best t.

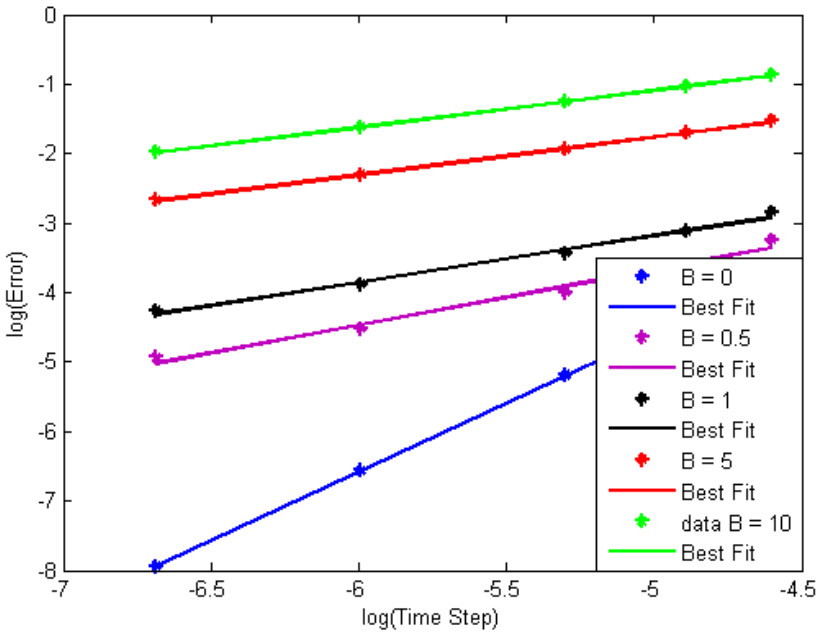


Figure 3.6: Graph showing how the size of the timestep a ects the error, for varying values of B. log( ) is plotted against log the log of the error produced with a particular size timestep. Crosses represent experimental results obtained as described in the text. A line of best t is then added to each 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 O(^{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 t 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 t 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 a ect 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 a ects the solution we run the model as in Figure 3.3 but with a nal 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.

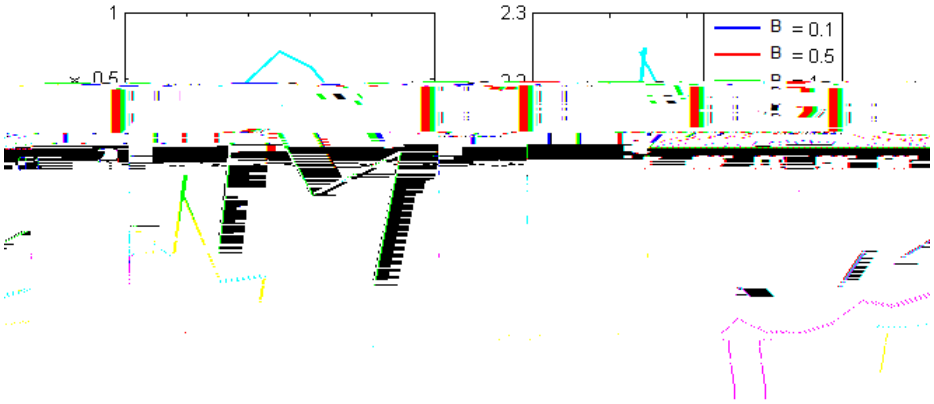


Figure 3.7: A ect on solution of changing B. Each line shows the EM solution to the stochastic Lorenz equations for a di erent value of B. Initial conditions $x = 0{:}00001$, $y = 0{:}00001$, $z = 2{:}00001$. From time $t = 0$ to $t = 0{:}1$ with a timestep = 0:01.

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

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.

to forecast the particles one time step. However as we are MT we need to treat the state to be mode tracked di erently 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 modi ed so it only forecast the required state. The remaining states are also forecast with this modi ed 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).

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

where

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

and J_o is as in equation (2.13). Equation (4.2) is a speci c 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 o 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 di erently 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 nal 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

4.3.2.1 Creating the Rank Histograms

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 di erent situations and the qualitative results will be compared. Primarily we are interested in how the number of particles a ects 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 di er depending on the number of particles used. For this reason the only input data that di ers 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 = 0:01, Initial $x = 0{:}00001$, Initial *yTd [(00001,) -309 (InitiF*

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

5.1.1 Results for $N = 2$

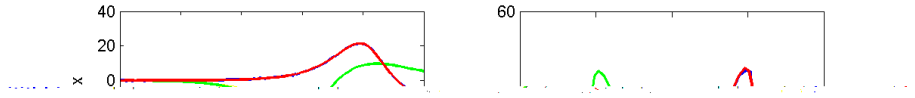


Figure 5.1: PF solution of the Lorenz equations with N = 2 particles. The red line is the true solution, the dark blue line is observations, the green line is the particle mean calculated in equation (4.4), the light blue lines represent the particle trajectories obtained for each particle. Initial conditions x = 0:00001, y = 0:00001, z = 2:00001. From time t = 0 to t = 1 with a timestep = 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 di er. 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 di er 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 di erent 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 butter y whereas the true solution

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 di er 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 su ciently 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

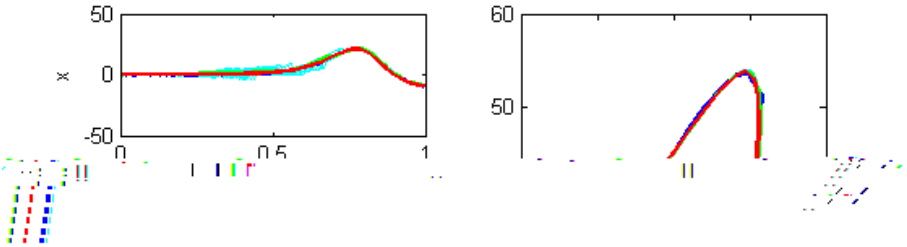


Figure 5.3: PF solution of the Lorenz equations with $N = 50$ particles. Colours as in Figure 5.1. Initial conditions $x = 0{:}00001$, $y = 0{:}00001$, $z = 2{:}00001$. From time $t = 0$ to $t = 1$ with a timestep $= 0{:}01$. Noise parameter $B = 0{:}1$.

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 di erent 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

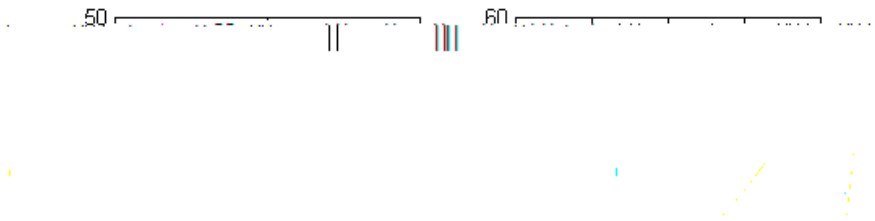


Figure 5.4: PF solution of the Lorenz equations with N = 500 particles. Colours as in Figure 5.1. Initial conditions x = 0:00001, y = 0:00001, z = 2:00001. From time t = 0 to t = 1 with a timestep = 0:01. Noise parameter B = 0:1.

run with a di erent true solution and noise realisation. Due to time limitation we have not veri ed 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

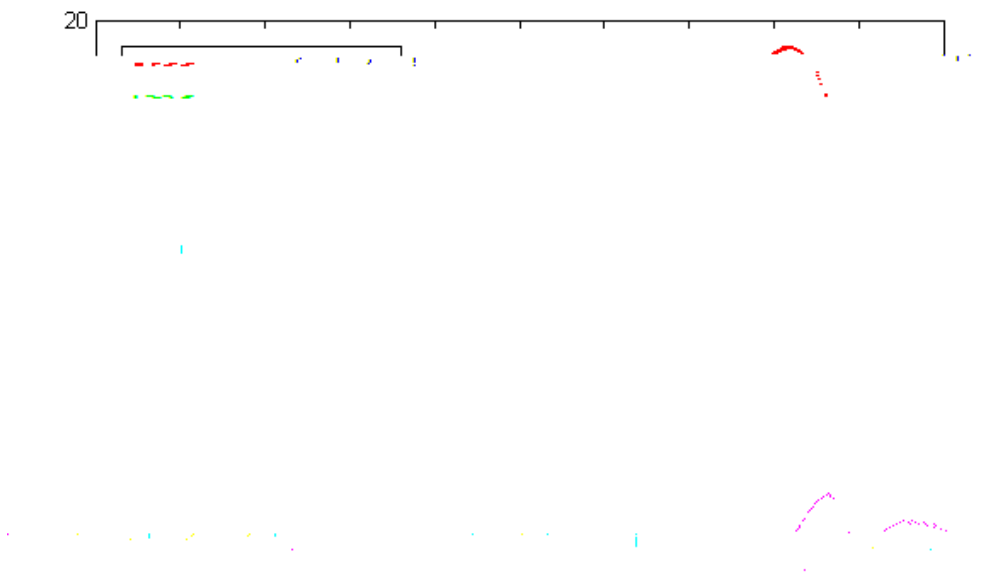


Figure 5.5: RMSE for various number of particles. Calculated from model runs with initial conditions $x = 0{:}00001$, $y = 0{:}00001$, $z = 2{:}00001$. From time $t = 0$ to $t = 1$ with a timestep $= 0{:}01$. Noise parameters $B = 0{:}1$.

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 signi cant 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

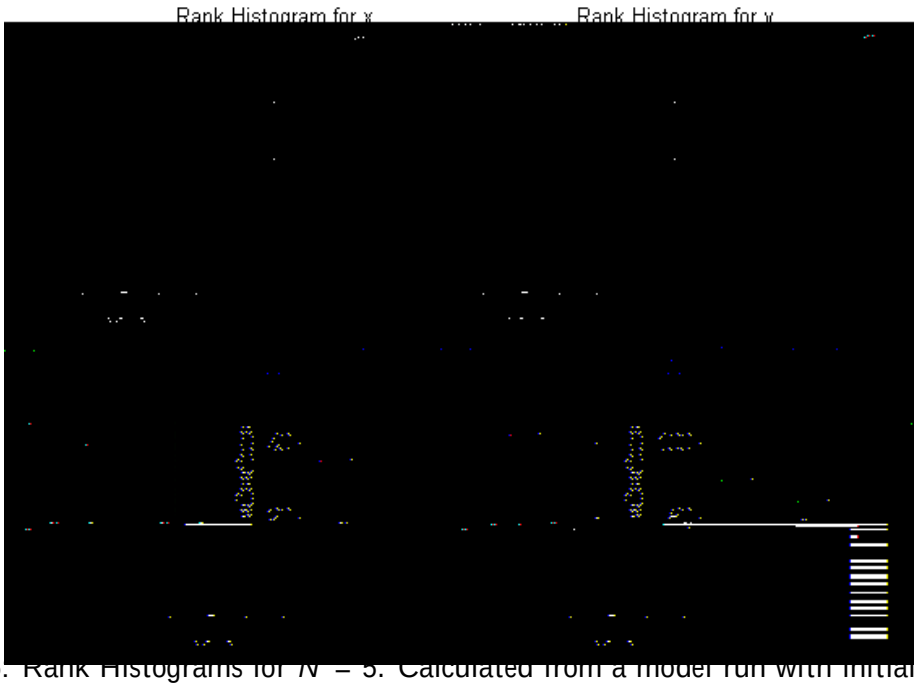


Figure 5.6: Rank Histograms for $N = 5$. Calculated from a model run with initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $= 0.01$. Noise parameters $B = 0.1$. `Realisations' refers to the number of occurrences of the observation per bin.

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

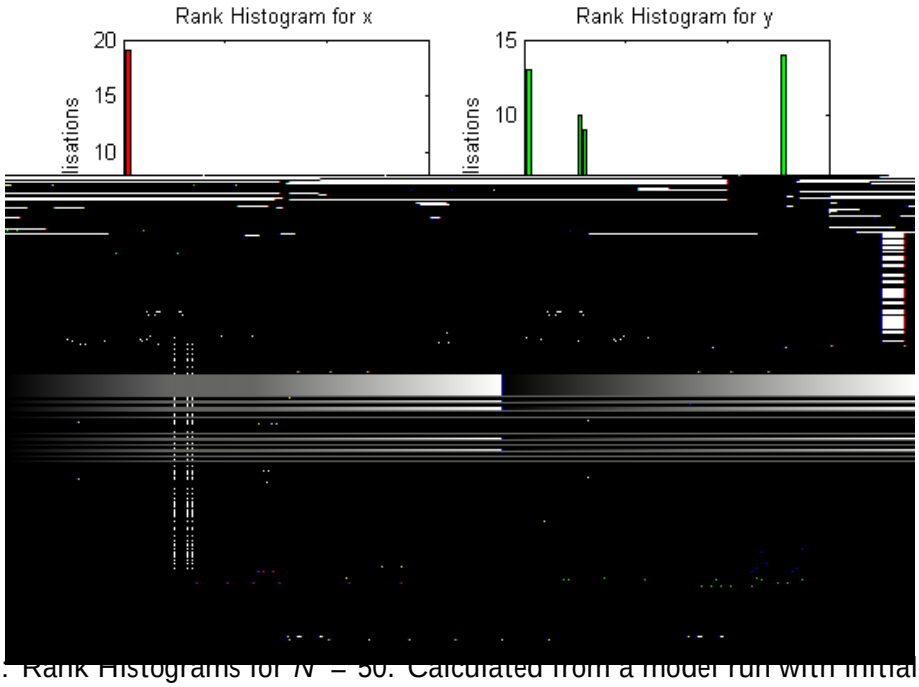


Figure 5.7: Rank Histograms for $N = 50$. Calculated from a model run with initial conditions $x = 0.00001$, $y = 0.00001$, $z = 2.00001$. From time $t = 0$ to $t = 1$ with a timestep $= 0.01$. Noise parameters $B = 0.1$. `Realisations' refers to the number of occurrences of the observation per bin.

expect all the tallies from this time to be in the rst 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 rst bin, the y RH being U-shaped and the x RH being U-shaped but with more tallies in the rst bin. Although we have some explanation of the high tallies in the rst and last bins we would still like to see the values in the middle bins being uniform. Unfortunately the bins

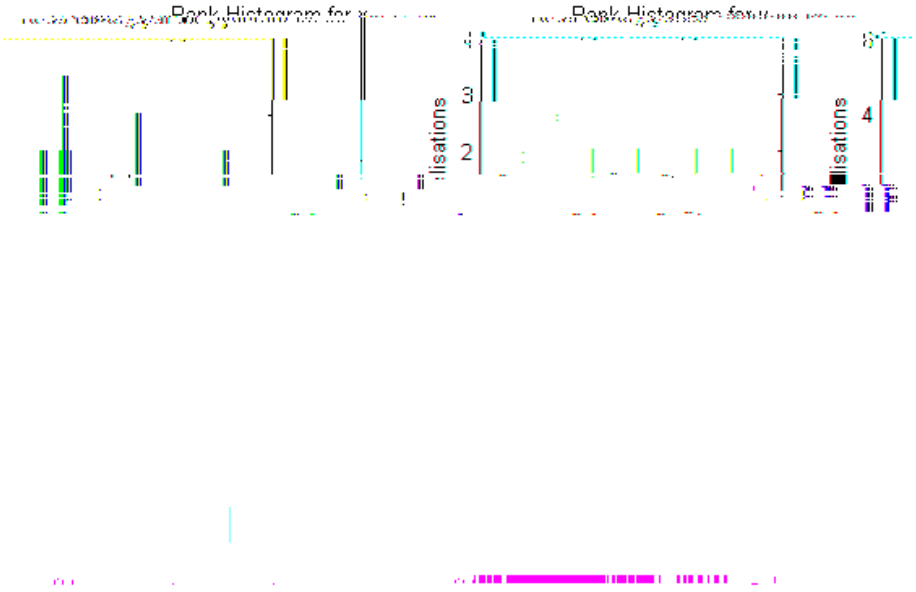


Figure 5.8: Rank Histograms for $N = 500$. Calculated from a model run with initial conditions $x = 0{:}00001$, $y = 0{:}00001$, $z = 2{:}00001$. From time $t = 0$ to $t = 1$ with a timestep $= 0{:}01$. Noise parameters $B = 0{:}1$. `Realisations' refers to the number of occurrences of the observation per bin.

vs. *t* plot do collapse down and it is di cult 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

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 a ect of increasing the number of particles is also considered. We then go on to see how the di erence between the observation error variance and the background error variance a ects 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 $\grave{}_{t;s}$ contains the minimum number of dimensions ... [such that] $p(\ _{t;r}|\ ^{i}_{t-1};\ ^{i}_{t;s};d_t)$ is unimodal'. This suggests that the way we split the state

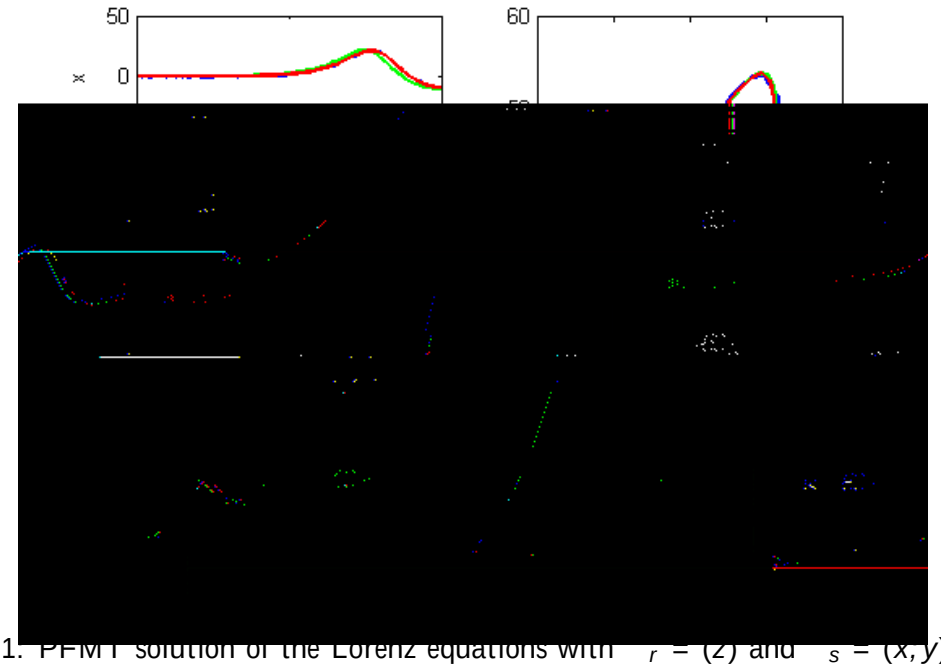
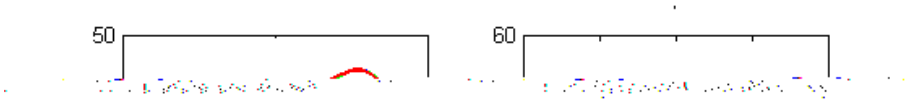


Figure 6.1: PFMT solution of the Lorenz equations with $_r = (z)$ and $_s = (x; y)$. Colours as in Figure 5.1. Initial conditions $x = 0{:}00001$, $y = 0{:}00001$, $z = 2{:}00001$. From time $t = 0$ to $t = 1$ with a timestep $= 0{:}01$. Noise parameter $B = 0{:}1$. $N = 500$ particles.

$t = 0$ to $t = 1$ with a timestep $= 0{:}01$, 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, $_r = z$. 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, $_r = (x; y)$. This result is shown in Figure 6.2. We see that this solution is better than the one shown in


50
60

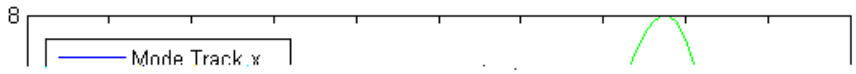
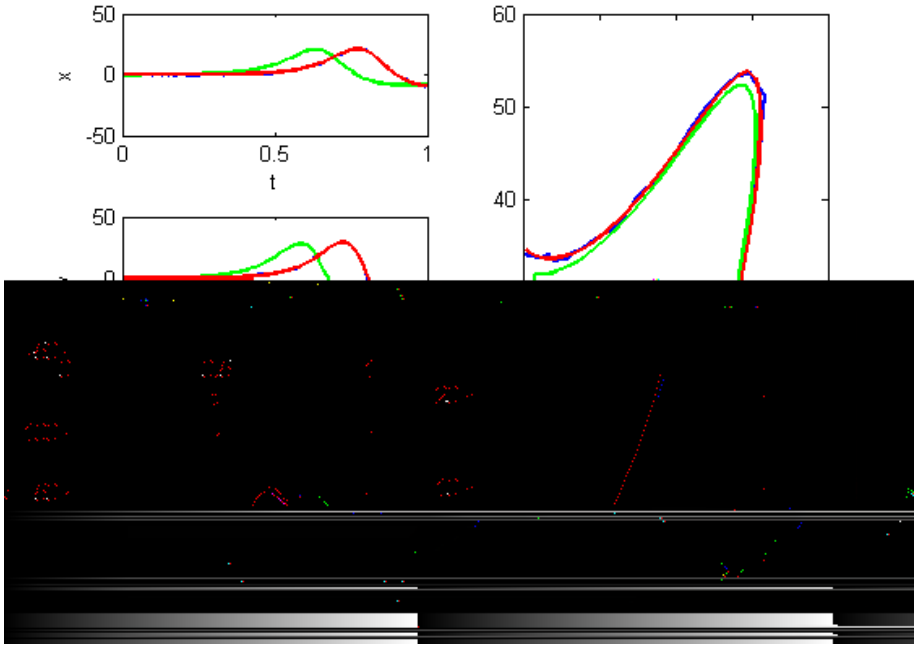


Figure 6.3: RMSE for the PFMT solutions of the Lorenz Equations with different parts of the state mode tracked

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 $_r = (x; y)$ gives the most accurate result. As MT produces the more accurate result we expect the $_{t;r}$ dimensions to be more accurate. When z is mode tracked its value depends on two non mode tracked


50
0
-50
x
0
0.5
1
t
60
50
40

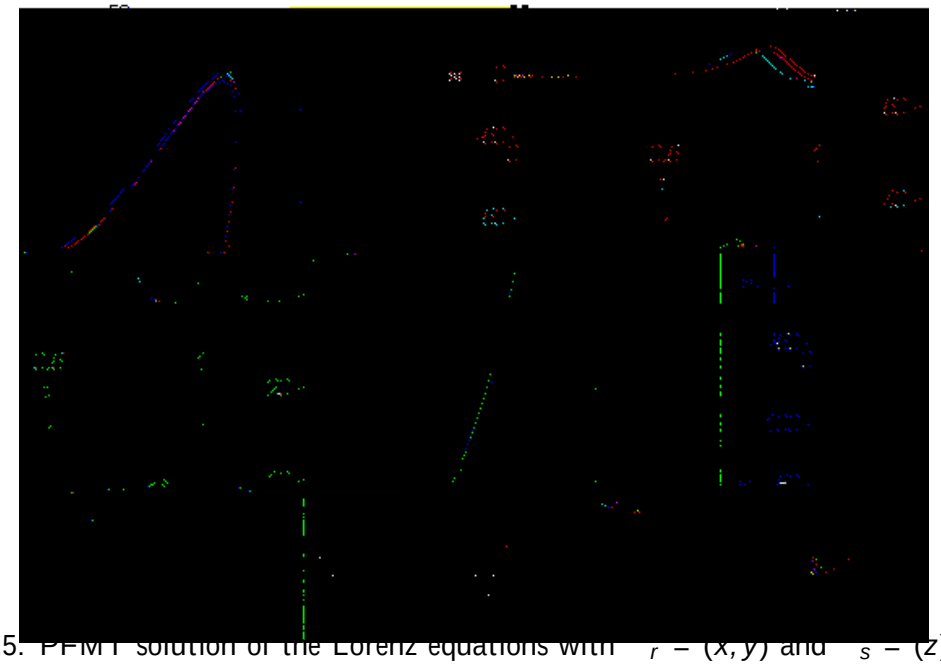


Figure 6.5. PFMT solution of the Lorenz equations with $\Pi_r = (x, y)$ and $\Pi_s = (z)$. Colours as in Figure 5.1. Initial conditions $x = 0{:}00001$, $y = 0{:}00001$, $z = 2{:}00001$. From time $t = 0$ to t

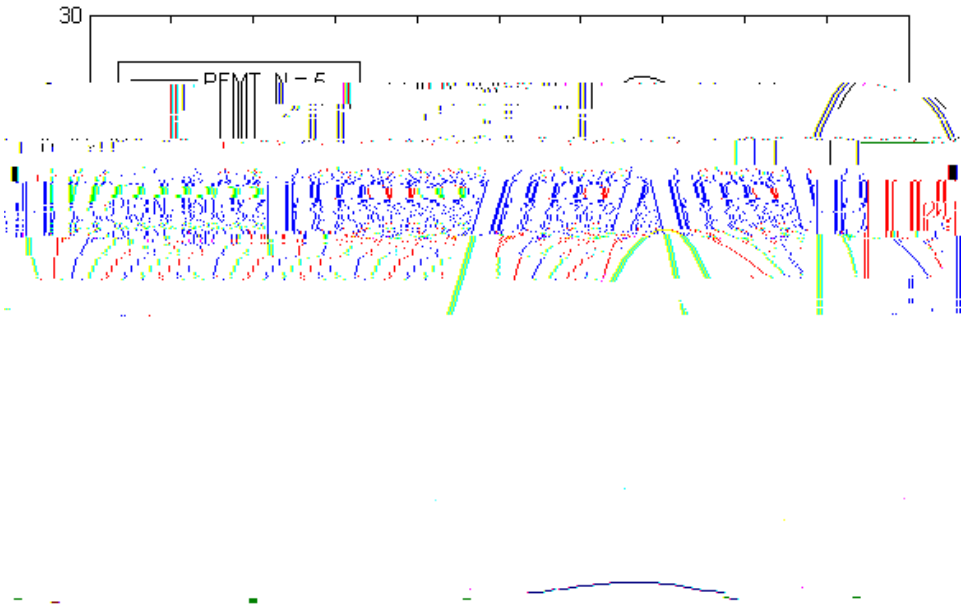


Figure 6.6: RMSE produced from the PFMT ($_r = (x;y)$) and PF codes with various values of N

more obvious. The code is run for $_r = (x; y)$ with initial conditions $x = 0{:}00001$, $y = 0{:}00001$, $z = 2{:}00001$. Time $t = 0$ to $t = 1$ with a timestep $= 0{:}01$ 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 signi cant 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

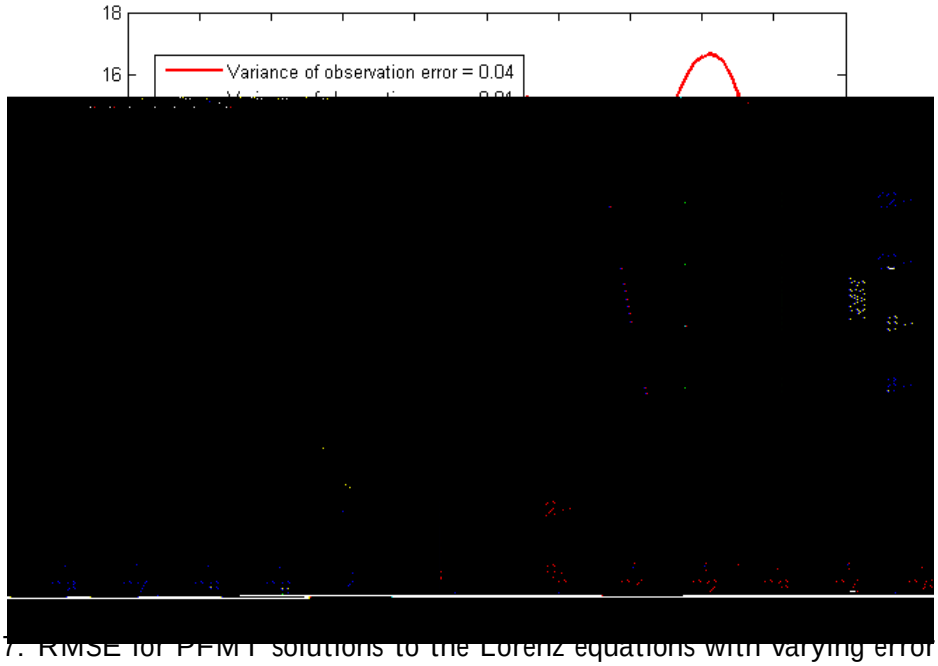


Figure 6.7: RMSE for PFMT solutions to the Lorenz equations with varying error variances

the a ect 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 $_r = (x; y)$ and $_s = z$. 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 di er to those seen in Vaswani [2008] which suggest that the results obtained from the PFMT are always superior to the

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 ltering 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 di ered signi cantly 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 di erences 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

result would improve, but the improvement seen was more signi cant than expected. This sensitivity of the model is supported by Johnson et al. [2005], who show that the correct ratio

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

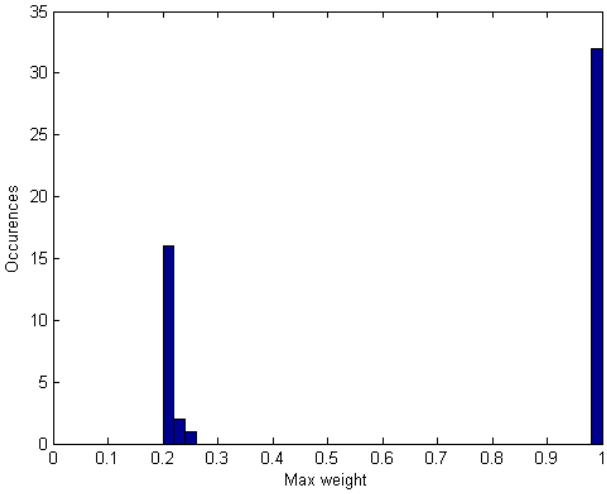
In section 4.3.2 we used RH to asses the quality of our solutions. However we found that they were not ideal as they assumed equally weighted particles and no resampling. We mentioned that it may be more appropriate to use a reliability diagram [Brocker and Smith, 2007], [Wilkes, 1995] to tell us about 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


35
30
25
20
15
10
5
0
Occurences
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Max weight

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 di erent 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 = 0:01. Noise parameter B = 0:1. N = 500 particles.

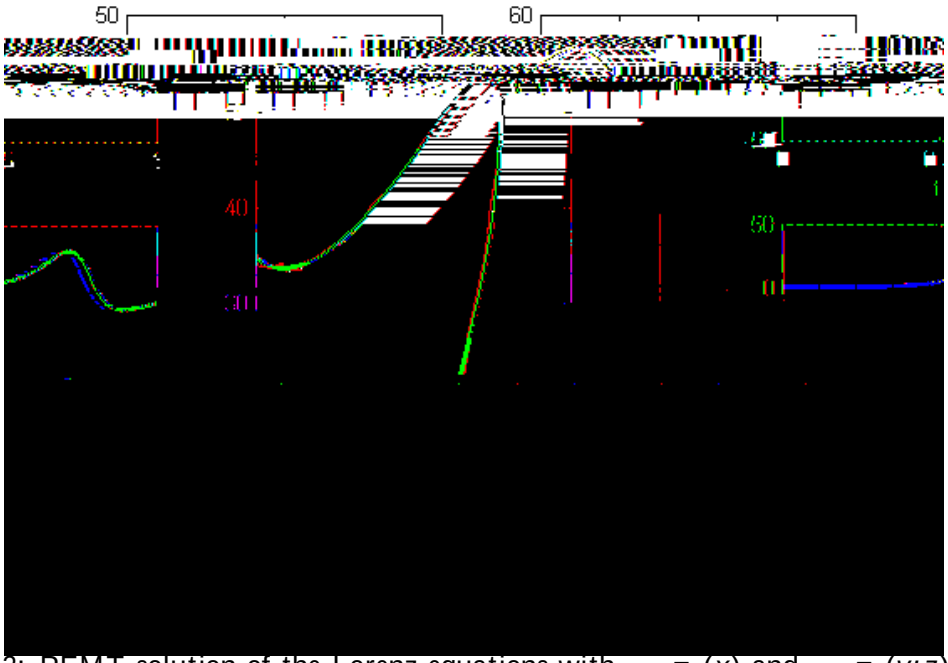


Figure 7.2: PFMT solution of the Lorenz equations with $_r = (x)$ and $_s = (y;z)$. Colours as in Figure 5.1. Initial conditions x = 0:00001, y = 0:00001, z = 2:00001. From time t = 0 to t = 1 with a timestep = 0:01. Noise parameter B = 0:

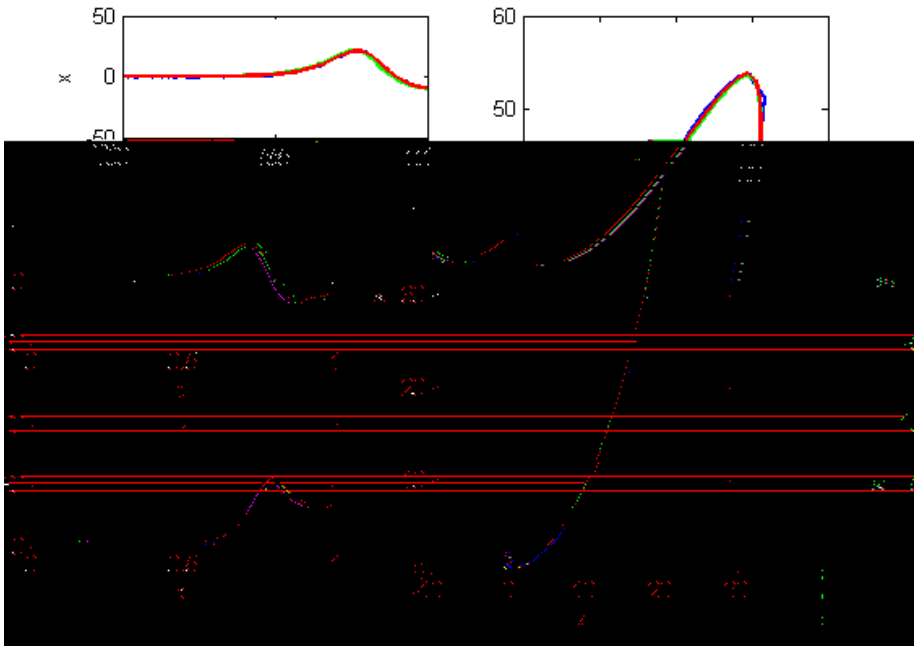


Figure 7.3: PFMT solution of the Lorenz equations with $_r$ = (y) and $_s$ = ($x;z$). Colours as in Figure 5.1. Initial conditions x = 0:00001, y = 0:00001, z = 2:00001. From time t = 0 to t = 1 with a timestep = 0:01. Noise parameter B = 0:1. N = 500 particles.

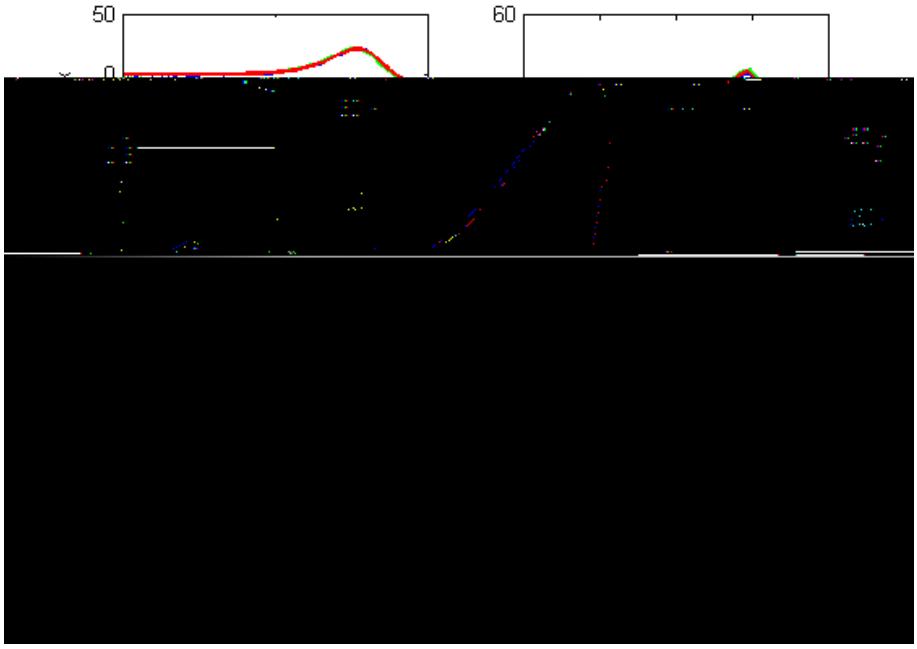


Figure 7.4: PFMT solution of the Lorenz equations with $_r = (x; z)$ and $_s = (y)$. Colours as in Figure 5.1. Initial conditions x

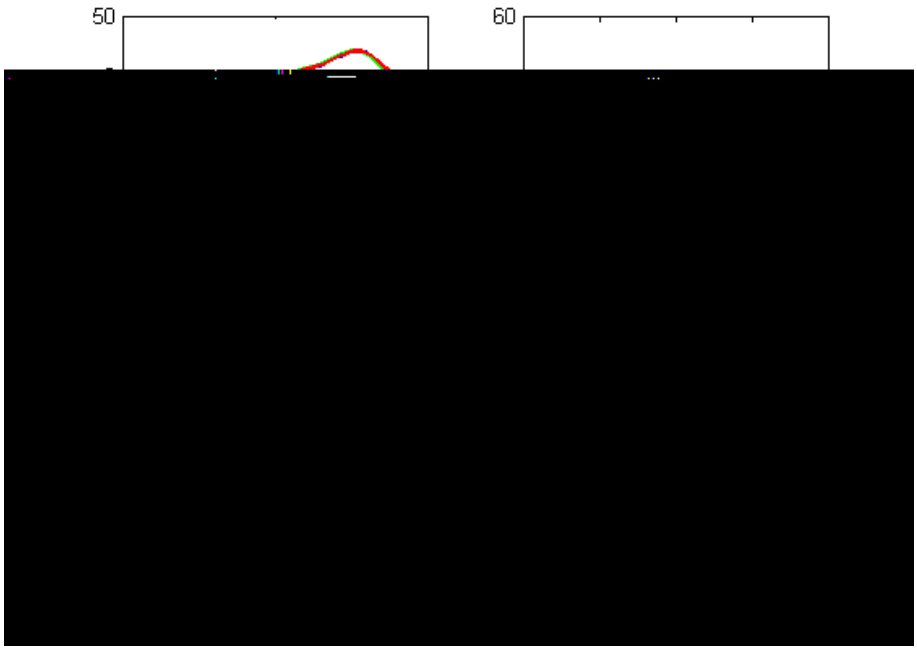


Figure 7.5: PFMT solution of the Lorenz equations with $_r = (y; z)$ and $_s = (x)$. Colours as in Figure 5.1. Initial conditions $x = 0{:}00001$, $y = 0{:}00001$, $z = 2{:}00001$. From time $t = 0$ to $t = 1$ with a timestep $= 0{:}01$. Noise parameter $B = 0{:}1$. $N = 500$ particles.

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 $_r$ = (x;y) and $_s$ = (z). Initial conditions x = 0:00001, y = 0:00001, z = 2:00001. From time t = 0 to t = 1 with a timestep = 0:01. Noise parameter B = 0:1. N = 50 particles.

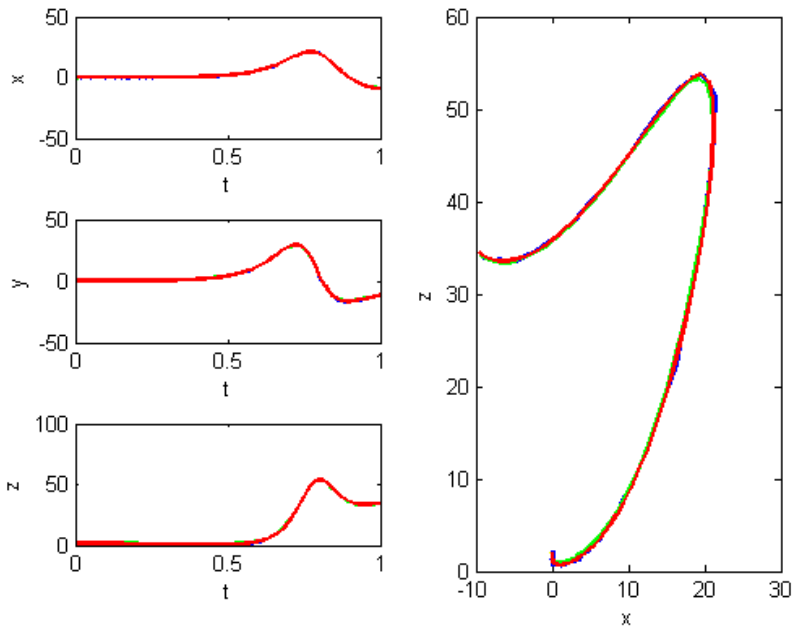


Figure 7.6: PFMT solution of the Lorenz equations with $_r$ = (x;y) and $_s$ = (z). Colours as in Figure 5.1. Initial conditions x = 0:00001, y = 0:00001, z = 2:00001. From time t = 0 to t = 1 with a timestep = 0:01. Noise parameter B = 0:1. N = 50 particles. Observation error variance = 0:01

Bibliography

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

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

T. Bengtsson, C. Snyder, and D. Nychka. Toward a nonlinear ensemble lter 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 ltering. *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 ltering.

D. Fox. Adapting the sample size in particle lters 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 lter 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 Di erential Equations*. Springer, 1999.

H. Lee and W. Schiesser. *Ordinary and Partial Di erential Equation Routi/F46 10.(Parti60es18.3,)-334(pag*

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

Y. Saito and T. Mitsui. Stability analysis of numerical schemes for stochastic di erential 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 ltering. *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.meto ce.gov.uk/science/creating/daysahead/nwp/index.html, Last accessed 19th August 2009.

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

N. Vaswani. Particle ltering 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 con rm 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.