Three Filters and Their Applications: A Comparison Case Study

by

Yan Zhao

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Financial Mathematics

by

Yan Zhao

April 2018

APPROVED:

Zhongqiang Zhang

Professor Zhongqiang Zhang, Major Thesis Advisor

[illegible]

Professor Luca Capogna, Head of Department

Abstract

Filtering has been shown successful in prediction from dynamically changing data. In this thesis, we perform case studies and comparison among three filters: Kalman filter, unscented Kalman filter and particle flow filter.

We consider Kalman filter in the first chapter where we focus on studying the S&P model in a time-discrete dynamics with time-discrete observations for dividend yield and S&P returns. For this filtering problem, Kalman filter performs well only in the first few time steps.

Since the S&P model we consider is nonlinear, we are motivated to apply nonlinear filters and use unscented Kalman filter. The key technique is to approximate non-Gaussian processes (non-linear models) by assigning the so-called sigma points (nonrandom) around the priori mean. We implement it on the S&P model in Chapter 2. We also implement unscented Kalman filter for a two-dimensional tumor growth model. Unscented Kalman filter works reasonably well for both models with capturing the trend and predicting the values.

We consider the recently-developed particle flow filter in Chapter 3. Particle flow filter is a method of moving the particles by partial differential equations generated from proper chosen likelihood functions via the Bayes rule. By solving partial differential equations, one can construct an explicit dynamic model on how to move particles. In this chapter, we implement two models as in Chapter 2. One is the S&P model and the other is perturbed tumor growth model. We compare performance of particle flow filter and unscented Kalman filter for these two models.

Acknowledgements

I would like to express my gratitude to my advisor, Professor Zhongqiang Zhang at Department of Mathematical Sciences at Worcester Polytechnic Institute. The door to Prof. Zhang's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently helped me in understanding mathematical problems, implementing Matlab code and writing the paper.

Contents

List of Figures

List of Tables

Chapter 1

Kalman Filter

Filtering is a simulation method for obtaining optimal estimation from data and an imperfect model. It contains two models – the first is a measurement model and the second is an observation model.

Kalman Filter has been used to minimize the estimation error for unknown variables in a noisy stochastic system, when observation models are linear. Kalman Filter works recursively to update estimation by incorporating observed measurements over time. The measurement model is used to generate prior estimation for current state variables; observation model is used to update the estimation and create posterior estimation. Kalman Filter has broad applications, such as predicting natural weather and prices of traded commodities. It also has been used to monitor complex dynamic systems, like signal processing in GPS and motion monitoring in robotics [3].

1.1 Linear Dynamic Systems in Discrete Time

The linear filtering problem can be presented by combining the following two models [3].

- Measurement model

$$\mathbf{x}_k = \phi_{k-1}\mathbf{x}_{k-1} + w_{k-1}, \qquad w_k \sim \mathcal{N}(0, \mathbf{Q}_k). \tag{1.1}$$

- Observation model

$$\mathbf{z}_k = H_k x_k + v_k, \qquad v_k \sim \mathcal{N}(0, \mathbf{R}_k). \tag{1.2}$$

Here v_k and w_k are assumed as independent Gaussian random processes with the mean of zero. The initial value of $\mathbf{x}_k$ is known, denoted as x_0, with known initial covariance matrix P_0.

The goal is to find estimations of $\hat{\mathbf{x}}_k$ presented by function of $\mathbf{z}_k$ such that the mean-squared error is minimized. Denote $P_{k(-)}$ as the prior covariance matrix for x at time t_k, $P_{k(+)}$ as the posterior covariance matrix for x at time t_k, $\bar{K}_k$ as Kalman gain at time t_k, $\hat{x}_{k(-)}$ as the prior estimate of $\mathbf{x}_k$ and $\hat{\mathbf{x}}_{k(+)}$ as the posterior estimate of $\mathbf{x}_k$. By using orthogonality properties, we can prove the following updating equations [3]:

$$\begin{aligned}
P_{k(-)} &= \phi_{k-1} P_{(k-1)(+)} \phi_{k-1}^T + Q_{k-1}, && (1.3)\\
\bar{K}_k &= P_{k(-)} H_k^T [H_k P_{k(-)} H_k^T + R_k]^{-1}, && (1.4)\\
P_{k(+)} &= [I - \bar{K}_k H_k] P_{k(-)}, && (1.5)\\
\hat{\mathbf{x}}_{k(-)} &= \phi_{k-1} \hat{\mathbf{x}}_{(k-1)(+)}, && (1.6)\\
\hat{\mathbf{x}}_{k(+)} &= \hat{\mathbf{x}}_{k(-)} + \bar{K}_k [\mathbf{z}_k - H_k \hat{\mathbf{x}}_{k(-)}]. && (1.7)
\end{aligned}$$

The Kalman filter works only when the observation model is linear, and the noise is Gaussian distributed, i.e., H_k doesn't depend on $\mathbf{x}_k$ and the covariance matrix R_k doesn't depend on $\mathbf{x}_k$ in (1.2). In the next section, we present an application from finance and show that Kalman filter only works for a very short time.

1.2 A Nonlinear Model – the S&P Model

Changes in the stock market have drawn people's attention for a long time. It is natural to use filters to predict trends in the stock market over time. Predicting dividend yield and real return will be significantly important for investors if we have the historical observation. Consider a stochastic model of S&P with two variables, one variable is dividend yield, and the other variable is the real return [4].

$$\begin{aligned} Z_n &= \begin{pmatrix} X_n \\ \delta R_n \end{pmatrix} = \begin{pmatrix} \frac{1}{1+k} X_{n-1} + \frac{k\theta}{1+k} + \frac{\sigma}{1+k}\sqrt{X_{n-1}}\Delta W_{1,n} \\ \mu X_n + a\sqrt{X_{n-1}}\left(\rho \Delta W_{1,n} + \sqrt{1-\rho^2}\Delta W_{2,n}\right) \end{pmatrix}, \\ Y_n &= \begin{pmatrix} Y_{1,n} \\ Y_{2,n} \end{pmatrix} = \begin{pmatrix} X_n + Q_1 B_{1,n} \\ \delta R_n + Q_2 B_{2,n}. \end{pmatrix}. \end{aligned}$$

Here X_n is dividend yield, δR_n is real return and Y_n is a two-dimensional vector for the observation of X_n and δR_n. Noises $\Delta W_{1,n}, \Delta W_{2,n}$ are independent Brownian motion increments with $\Delta W_{i,n} = W_{i,n+1} - W_{i,n}, i = 1, 2$. $B_{1,n}, B_{2,n}$ are also independent Brownian motion increments. k, θ, σ, μ, a, ρ, Q_1 and Q_2 are parameters with the given values as follows.

Table 1.1: Parameters for the S&P dividend yield and real return model, obtained by applying a maximum likelihood estimation, see [6].

k	θ	σ	μ	a	ρ	Q_1	Q_2
2.0714	2.0451	0.3003	0.1907	0.9197	1.6309	0.0310	-0.8857

Historical observations of Y_n from the year 1945 to 2010 can be found online. It is natural for us to predict year-end yield and return by using observed data from the previous year and to compare it with the real data such that we can clarify whether the method is used to make precise predictions. In the following section, we present how we use the Kalman filter for the S&P model.

1.2.1 Rewrite the S&P Model

It is readily seen that our S&P model does not have the matrix form as shown in (1.1) and (1.2). In order to use Kalman filter method, we rewrite it to matrix form as follows.

$$Z_n = \Phi Z_{n-1} + D + M\sqrt{Z_{n-1}}CW_n, \tag{1.8}$$

$$Y_n = HZ_n + VB_n. \tag{1.9}$$

Here we denote

$$\Phi = \begin{pmatrix} \frac{1}{1+k} & 0 \\ \frac{\mu}{1+k} & 0 \end{pmatrix}, \quad D = \begin{pmatrix} \frac{k\theta}{1+k} \\ \frac{\mu k\theta}{1+k} \end{pmatrix}, \quad M = \begin{pmatrix} 1 & 0 \end{pmatrix}, \quad C = \begin{pmatrix} \frac{\sigma}{1+k} & 0 \\ \frac{\mu\sigma}{1+k} + a\rho & a\sqrt{1-\rho^2} \end{pmatrix}, \quad W_n = \begin{pmatrix} \Delta W_{1,n} \\ \Delta W_{2,n} \end{pmatrix},$$

and

$$H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad V = \begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix}, \quad B_n = \begin{pmatrix} B_{1,n} \\ B_{2,n} \end{pmatrix}.$$

We present how we rewrite observation and measurement model of dividend yield and real return into matrix form in the following. When rewriting Z_n, one can see that X_n is represented by Z_{n-1}, while δR_n is represented by Z_n. Thus, we should rewrite δR_n in terms of Z_{n-1} and

$$\delta R_n = \frac{\mu}{1+k}X_{n-1} + \frac{\mu k\theta}{1+k} + (\frac{\mu\sigma\sqrt{x_{n-1}}}{1+k} + a\rho\sqrt{X_{n-1}})\Delta W_{1,n} + a\sqrt{X_{n-1}}\sqrt{1-\rho^2}\Delta W_{2,n}.$$

Then Z_n becomes to the following,

$$Z_n = \begin{pmatrix} X_n \\ \delta R_n \end{pmatrix} = \begin{pmatrix} \frac{1}{1+k} & 0 \\ \frac{\mu}{1+k} & 0 \end{pmatrix} \begin{pmatrix} X_{n-1} \\ \delta R_{n-1} \end{pmatrix} + \begin{pmatrix} \frac{k\theta}{1+k} \\ \frac{\mu k\theta}{1+k} \end{pmatrix} + \sqrt{X_{n-1}} \begin{pmatrix} \frac{\sigma}{1+k} & 0 \\ \frac{\mu\sigma}{1+k} + a\rho & a\sqrt{1-\rho^2} \end{pmatrix} \begin{pmatrix} \Delta W_{1,n} \\ \Delta W_{2,n} \end{pmatrix}.$$

The rewriting of Y_n in a matrix form is straightforward. We follow the presentation in [3] to derive a solution for our model. After we rewrite the S&P model, we are able to obtain the updated equations.

1.2.2 Updated Equations for the S&P Model

Following the idea of Kalman filter of deriving Kalman gain, prior covariance, posterior covariance, prior estimation and posterior estimation, one can generate the following updated equations for our S&P model.

$$\begin{aligned}
P_{n(-)} &= \Phi_{n-1}P_{n-1(+)}\Phi_{n-1}^T + MZ_{n-1}CC^T, && (1.10)\\
\bar{K}_n &= P_{n(-)}H_n^T(H_nP_{n(-)}H_n^T + V^2)^{-1}, && (1.11)\\
P_{n(+)} &= (I - \bar{K}_nH_n)P_{n(-)}, && (1.12)\\
\hat{Z}_{n(-)} &= \Phi_{n-1}\hat{Z}_{n(+)} + D, && (1.13)\\
\hat{Z}_{n(+)} &= \hat{Z}_{n(-)} + \bar{K}_n(-H_n\hat{Z}_{n(-)} + Y_n). && (1.14)
\end{aligned}$$

Details of deriving these updated equations can be found in Appendix A. These equations from our S&P model are similar to (1.3) to (1.7), but some of them are different because the noise parts of the S&P model is not strictly linear and rely on the previous steps.

By plugging observed yield and return from the year 1945 to the year 2010 to Y_n and setting the initial prior covariance as zero, one can repeat the updated equations from (1.10) to (1.14) for 65 times. Each time we update Kalman gain, prior & posterior covariance and prior & posterior estimation for once, and record the posterior estimation as the predicted value. The Matlab code for implementing Kalman filter on the S&P model can be found in Appendix B. Figure 1.1 and 1.2 show the comparison between real values and estimated values with the same time discretization.

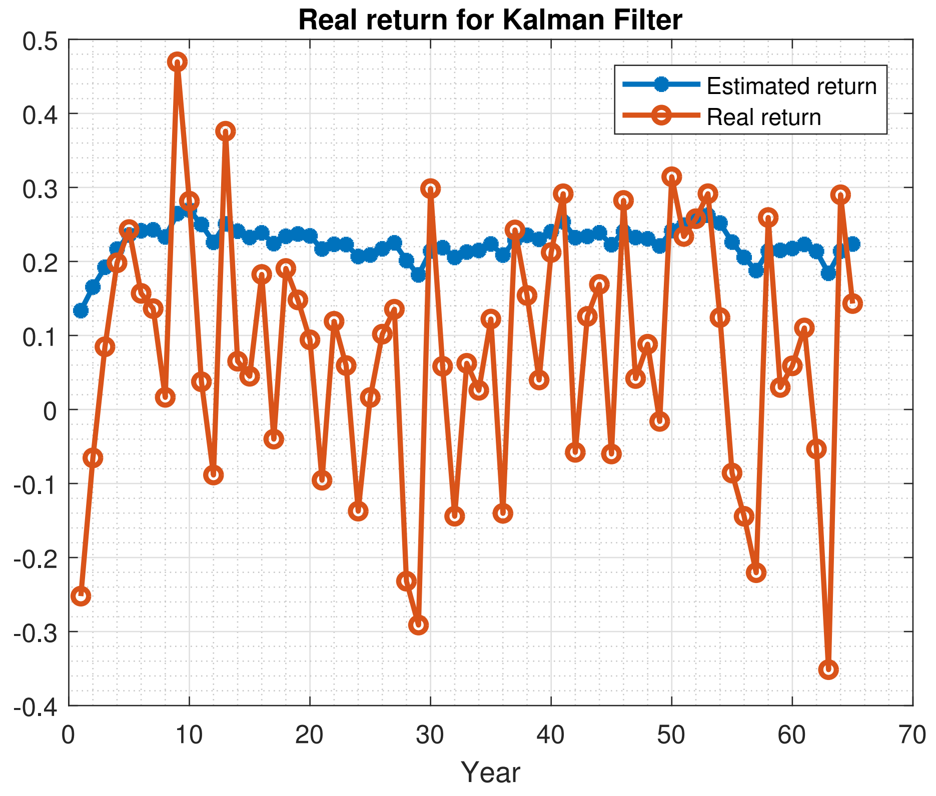

Figure 1.1: Kalman Filter for Real Return in Equation (1.8) and (1.9). Predicted return can well capture the trend but there are gaps between predicted ones and real ones.

1.2.3 Results for the S&P Model

In Figure 1.1 and Figure 1.2, the x-axis represents the year, with year zero substituting the initial year, which is the year 1945 in our model. And the 65th year (the year of 2010) is the last year. The y-axis represents the value of return and yield respectively. The results show that Kalman filter works well in the first five to six years with the same trend of movement and approximated estimation. Although the patterns keep almost the same trend between estimations and real ones after the fifth year, the values are not much closer to each other. The reason for these deviations is that our S&P model is non-linear. These results suggest that Kalman filter works for the first several steps for a non-linear model, but do not have satisfactory performance for the longer term in a non-linear model.

In next two chapters, we consider nonlinear filters and focus on unscented Kalman

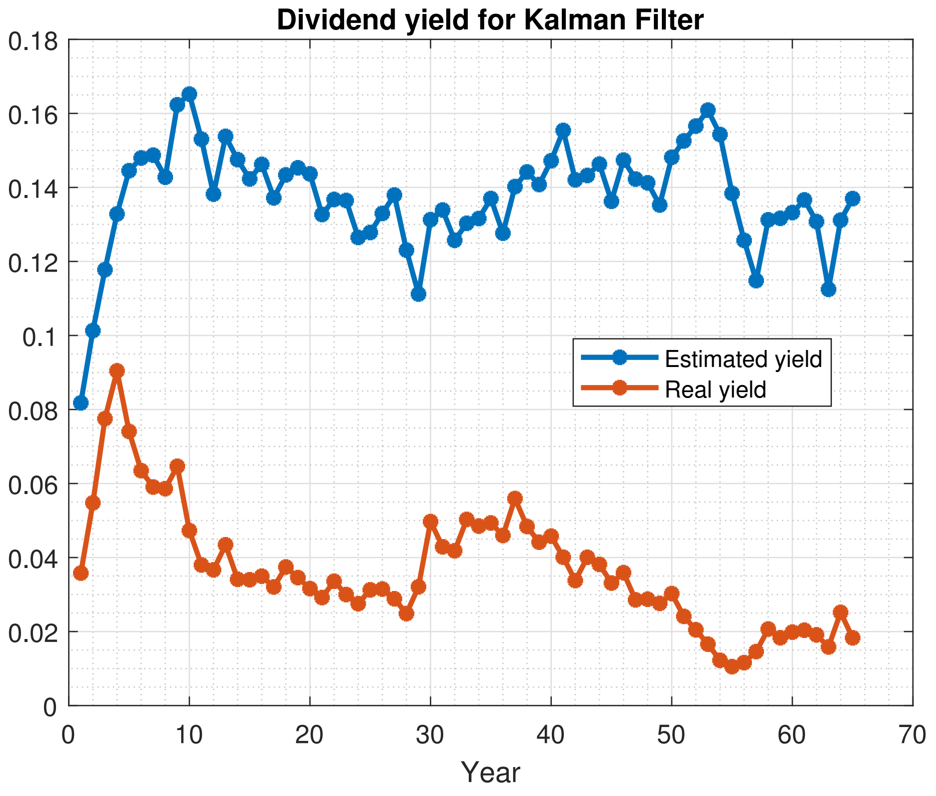

Figure 1.2: Kalman Filter for Yield in Equation (1.8) and (1.9). For the first few years, Kalman filter can well capture the trend of increasing.

filter for the S&P model in the next chapter.

Chapter 2

Unscented Kalman Filter

As shown in last chapter, Kalman filter is inaccurate for nonlinear models with non-Gaussian observations. The Unscented transformation (UT) [5] has been developed as an improvement to utilize information of mean and covariance to accurate results and make it easier to implement.

The UT method is to select sigma points from the distribution with mean μ_x and variance σ_x^2. In a scalar model, equispaced sigma points are chosen form the interval $[\mu_x - 2\sigma_x, \mu_x + 2\sigma_x]$. Then mean and covariance can be updated using those sigma points. Thus, we linearize non-linear models in some sense.

There are at least two advantages of using UT transformation. The first is that sigma points are no longer randomly chosen and they contain the most important information of an unknown distribution, which can be sufficient for statistic computation. The second is that weights for sigma points can be adjusted in ways such that points around mean can be weighted more.

2.1 Generating Sigma Points and Choosing Weights

Consider a set of sigma points S with given mean and covariance, which is often defined as $S = \{i = 0, 1, ...2N_x : X^{(i)}, W^{(i)}\}$ with $(2N_x + 1)$ points and their associated weights.

Here we assume different points have different weights of calculating. By convention, $W^{(0)}$ is the weight for the mean. i.e.

$$cX^{(0)} = \bar{X}_{(0)}, \quad W^{(0)} = W^{(0)}. \tag{2.1}$$

Typically, the expected value holds the highest weights. The other $2N_x$ points are randomly generated within $2\sqrt{N_x}$ standard deviation from the mean with half points on the left side of the mean and half on the right side of the mean. i.e.

$$\begin{aligned} X^{(j)} &= \bar{X} - \big(\sqrt{\frac{N_x}{1-W^0}\Sigma_x}\big)_i, \quad W^{(j)} = \frac{1-W^{(0)}}{2N_x}. && (2.2)\\ X^{(j+N_x)} &= \bar{X} + \big(\sqrt{\frac{N_x}{1-W^0}\Sigma_x}\big)_j, \quad W^{(j+N_x)} = \frac{1-W^{(0)}}{2N_x}. && (2.3)\end{aligned}$$

In the next section, we briefly present how to implement unscented Kalman filter, which is the general algorithm of unscented Kalman filter.

2.2 General Algorithms for Unscented Kalman Filter

We present the algorithm of Unscented Kalman Filter following the presentation in [5].

- **Step 1. Generating sigma points**

  The set of sigma points are generated by following (2.1) to (2.3).

- **Step 2. Generating transformed set**

  The generating transformed set is usually the expectation of X in measurement model,

$$\hat{X}_n^{(i)} = f[X_n^{(i)}, \mu_n].$$

- **Step 3. Computing predicted mean and computing predicted covariance**

$$\begin{aligned}\hat{\mu}_n &= \sum_{i=0}^{p} W^{(i)} \hat{X}_n^{(i)}, \\ \hat{K}_n &= \sum_{i=0}^{p} W^{(i)} \{\hat{X}_n^{(i)} - \hat{\mu}_n\}\{\hat{X}_n^{(i)} - \hat{\mu}_n\}^T.\end{aligned}$$

- **Step 4. Generating predicted observation**

  We get predicted observations by plugging each prediction points into observation model,

$$\hat{Y}_n^{(i)} = g[X_n^{(i)}].$$

- **Step 5. Computing mean of observations** $\hat{Y}_n = \sum_{i=0}^{p} W^{(i)} \hat{Y}_n^{(i)}$.

- **Step 6. Computing covariance of observations**

$$\hat{S}_n = \sum_{i=0}^{p} W^{(i)} \{\hat{Y}_n^{(i)} - \hat{Y}_n\}\{\hat{Y}_n^{(i)} - \hat{Y}_n\}^T.$$

- **Step 7. Updating normal Kalman filter equations**

$$\begin{aligned}\mathcal{V}_n &= Y_n - \hat{Y}_n, \quad W_n = \hat{K}_n \hat{S}_n^{-1}, \\ \mu_n &= \hat{\mu}_n + W_n \mathcal{V}_n, \quad K_n = \hat{K}_n - W_n \hat{S}_n W_n^T.\end{aligned}$$

2.3 Implementations of Unscented Kalman Filter

Here we consider the unscented Kalman filter for two nonlinear models. One is S&P model from [4] and the other is perturbed tumor growth model from [2]. Both examples are non-linear discrete models, with non-Gaussian observation models. In the next two subsections, we present how we implement the algorithm as well as the prediction results. We start with the S&P model.

2.3.1 Implementation and Results for the S&P Model

Here we use the same example of the two-dimensional S&P model which includes dividend yield and real return. We describe the model at page 3. After rewriting the model as standard matrix form, we have the following formula to represent measurement and observation model, in which Z_n is the two-dimensional measurement and Y_n is the observation.

$$\begin{aligned} Z_n &= \Phi_{n-1} Z_{n-1} + D + M\sqrt{Z_{n-1}} C W_n, \\ Y_n &= H_n Z_n + V B_n. \end{aligned}$$

Before we implement the algorithm, we need to set the initial value, in which we set $W^{(0)} = \frac{1}{3}$, $2N_x = 400$ and $K_0 = \mathbf{0}$. Then, one can repeat the algorithms by using steps listed at section 2.2.

There are three points needed to be mentioned when implementing the algorithm for our S&P model. When we generate sigma points, the mean point can be calculated from equation $\mu_n = H_n^{-1}(Y_n - VB_n)$, which has the weight of $\frac{1}{3}$. The remaining 400 sigma points are generated randomly from the range of two standard deviations around mean, and weight $\frac{2}{3}$ in total. Standard deviation at time n comes from the factorization of covariance of the previous step $\hat{K}_{n-1}$. We use the whole measurement function to generate transformed set in our S&P model instead of using expectation because the noisy parameters of Q_1 and Q_2 are relatively high. In each iteration, we need to grantee X_n is positive since the $\sqrt{X_n}$ has been used as a part of the real return.

We set the weight for mean as 1/3, total points as 400 and initial covariance for x_0 as 0. When we are not able to factorize of covariance, we set L as the square root of x. By plugging observed yield and return from the year 1945 to the year 2010 to Y_n, one can repeat the algorithm for 65 times. Each time we update predicted mean and predicted covariance for once, and record the predicted mean as the predicted value. The specific

Matlab code for implementing unscented Kalman filter on the S&P model can be found in Appendix C. Figure 2.1 and 2.2 show the comparison between real value and estimated value with the same time discretization.

Results for the S&P model

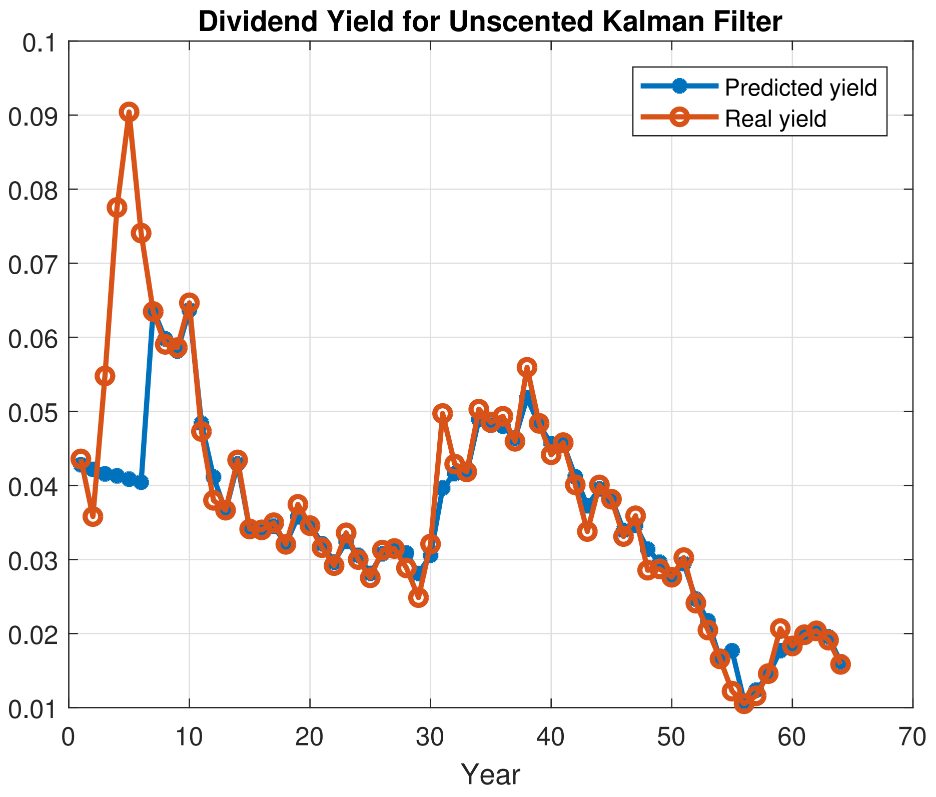

Figure 2.1: Unscented Kalman Filter for Dividend Yield in Equation (1.8) and (1.9). Predicted yield successfully match with the real yield after the first few years.

We also calculate the mean squared error (MSE) each time after generating sigma points. MSE is an estimator of sample variance by measuring the average of squares of the deviations. If $\mathbf{X}_{t_j}^{(i)}$ is the i-th sigma points at time t_j, and $\bar{\mathbf{X}}_{t_j}$ is the mean of the sigma points at time t_j, then the MSE of the predictor is computed as

$$MSE = \sqrt{\frac{1}{n(n-1)} \sum_{i=1}^{n} |(\mathbf{X}_{t_j}^{(i)})^2 - (\bar{\mathbf{X}}_{t_j})^2|}. \tag{2.4}$$

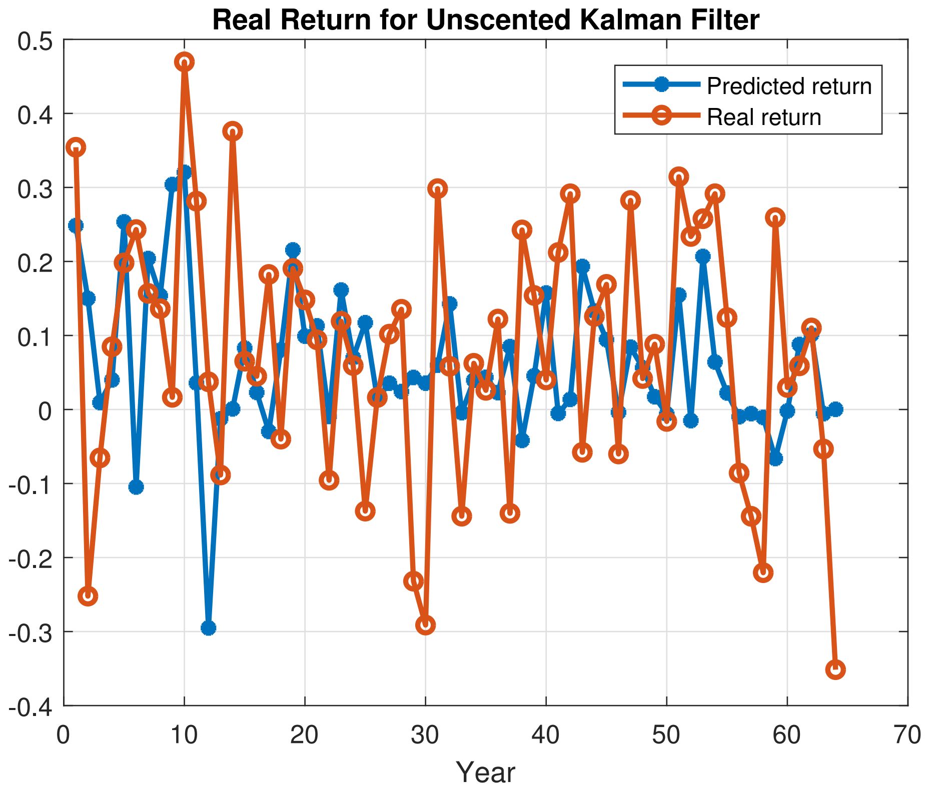

Figure 2.2: Unscented Kalman Filter for Return in Equation (1.8) and (1.9). Predicted return captures the most of trend and mismatch with the real return at some single points.

From Figure 2.1 and Figure 2.2, we see that predicted dividend yield matches highly with the real yield after the first few years, which means the application of unscented Kalman filter for predicting yield are pretty successful. Predicted real return captures the most of trend with the little mismatch at some single points.

Reasons for the different behaviors for yield and return are listed below.

First, the variance for return is larger than yield which we can see from Figure 2.3. The overall MSE for both yield and return are acceptably small, and MSE for return is larger than MSE for yield. That means sigma points for yield are more intensive with the more stable trend.

Second, return highly depends on the prediction of yield from the previous step. The predicted error for yield can be exaggerated further in return.

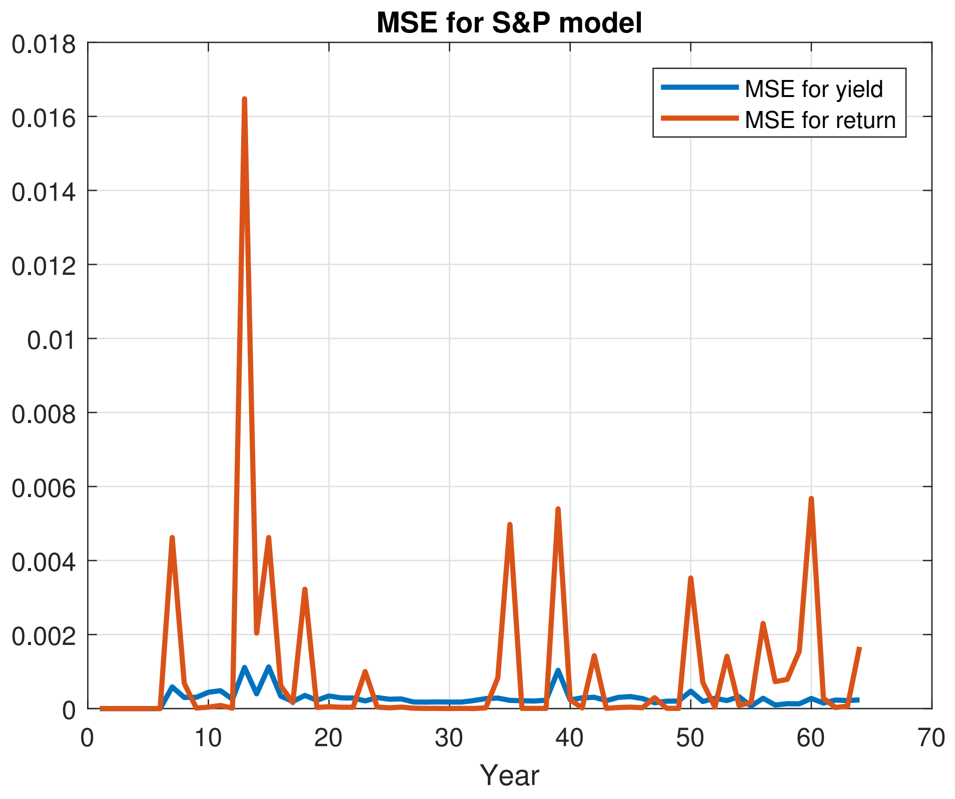

Figure 2.3: MSE for Yield and Return in Equation (1.8) and (1.9) with Unscented Kalman Filter. The overall MSE for both yield and return are acceptably small and MSE for return is larger than MSE for yield.

In the next section, we focus on an application from biology and show that unscented Kalman filter also works well in that model.

2.3.2 Implementation and Results for the Perturbed Tumor Growth Model

In this example, we consider the two-dimensional noise perturbed tumor growth model [2]

$$d\mathbf{X}_t = F(\mathbf{X}_t)dt + \sigma dW_t, \tag{2.5}$$

where W_t is a two-dimensional standard Brownian Motion with

$$\sigma = (0.01, 0.01)^T,\ \mathbf{X}_t = (X_t^1, X_t^2)^T,\ F(X_t) = (f_1(\mathbf{X}_t),\ f_2(\mathbf{X}_t))^T,$$

and f_1 and f_2 are defined by

$$f_1(\mathbf{X}_t) = \alpha_1 X_t^1 \ln(\frac{X_t^2}{X_t^1}),$$
$$f_2(\mathbf{X}_t) = \alpha_2 X_t^1 - \alpha_3 X_t^2 (X_t^1)^{\frac{2}{3}}.$$

Here f_1 models the Gompertzian growth rate if the tumor and f_2 gives the degree of vascularization of the tumor. The measurement model reads

$$Y_k = (X_k^1, X_k^2)^T + R\mathbf{v}_k, \tag{2.6}$$

where $\mathbf{v}_k$ is a two-dimensional zero mean Gaussian white noise with covariance $\Lambda = I\Delta$, I is an identity matrix and $R = (0.1, 0.1)^T$. We are given the step size $\Delta = 0.2$, the number of process $K = 40$, parameters value $\alpha_1 = 1, \alpha_2 = 0.2, \alpha_3 = 0.2$ and initial state $\mathbf{X}_0 = (0.8, 0.3)^T$.

We generate the "observation data" from simulating sample paths by using the forward Euler method which reads

$$\mathbf{X}_k = F(\mathbf{X}_{k-1})\Delta + \sigma\sqrt{\Delta}W_{k-1}. \tag{2.7}$$

We set the weight for mean as 1/3, total points as 1500 and initial covariance for x_0 as 0. When we are not able to factorize of covariance, we set L as the square root of x. By plugging observed X1 and X2 from time 0 to time 8 to Y_n, we can repeat the algorithm for 40 times. In each iteration, we update predicted mean and predicted covariance for once, and record the predicted mean as predicted values. The specific Matlab code for implementing unscented Kalman filter on the perturbed tumor growth model can be found in Appendix D. Figure 2.4 and 2.5 show the comparison between real value and estimated value with the same time discretization.

Results for Perturbed Tumor Growth Model

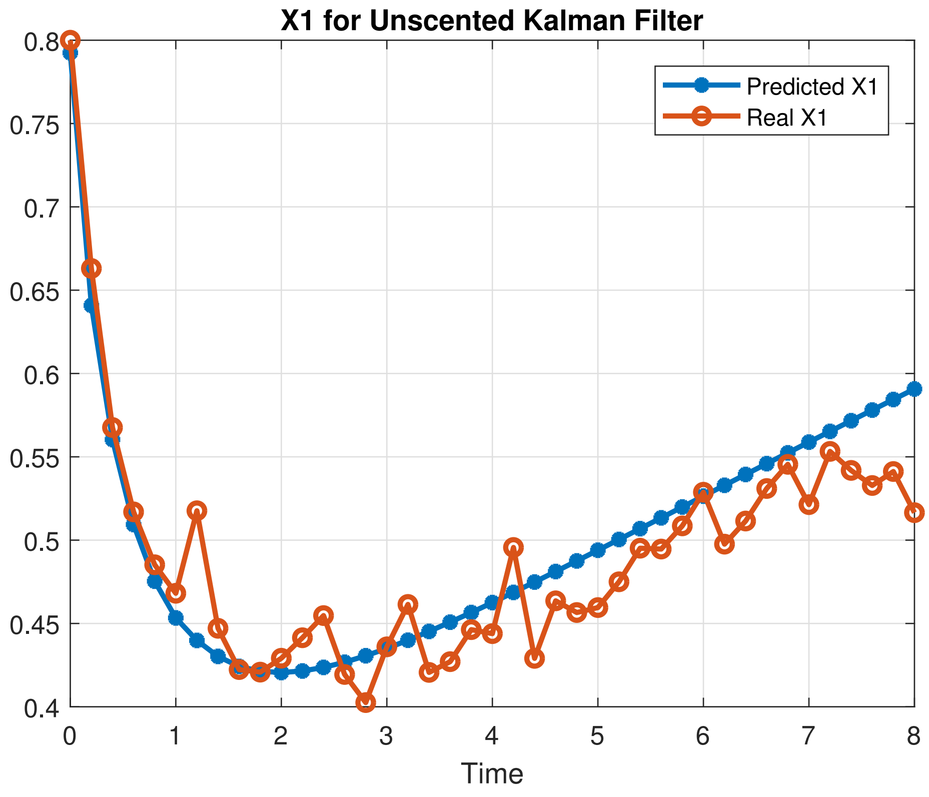

Figure 2.4: Unscented Kalman Filter for X1 in Equation (2.5) and (2.6). Predicted X1 matches well with the real X1 especially during the first half of the time.

We also calculate the mean squared error(MSE) of X1 and X2 for unscented Kalman filter following Equation (2.4) and present the values of MSE in Figure 2.6.

From Figure 2.4 and figure 2.5, we see that after time 3, predicted X1 is a little bit higher than real X1 but predicted X1 matches well with the actual X1 before time 3, which means the application of unscented Kalman filter for predicting X1 are pretty successful. Predicted X2 pairs well with the actual X2 and predicted X2 fluctuates less than real X2 because X2 has relatively low MSE from Figure 2.6, which means the majority of the

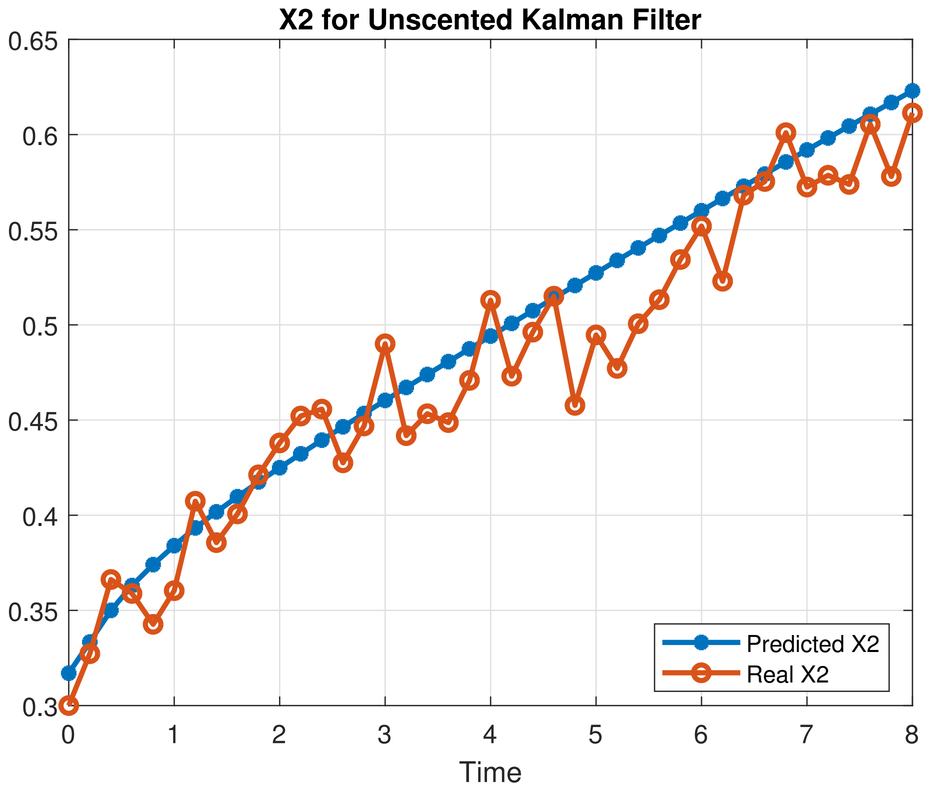

Figure 2.5: Unscented Kalman Filter for X1 in Equation (2.5) and (2.6). Predicted X2 matches well with the real X2 and predicted X2 fluctuate less than real X2.

sigma points of X2 concentrate on the mean value. From Figure 2.6, we can find that MSE has a sharp decreasing around time 1, which means the sigma points are concentrate around mean and we update around mean for the most of the time, that explains why predicted X2 value crosses the center of real X2.

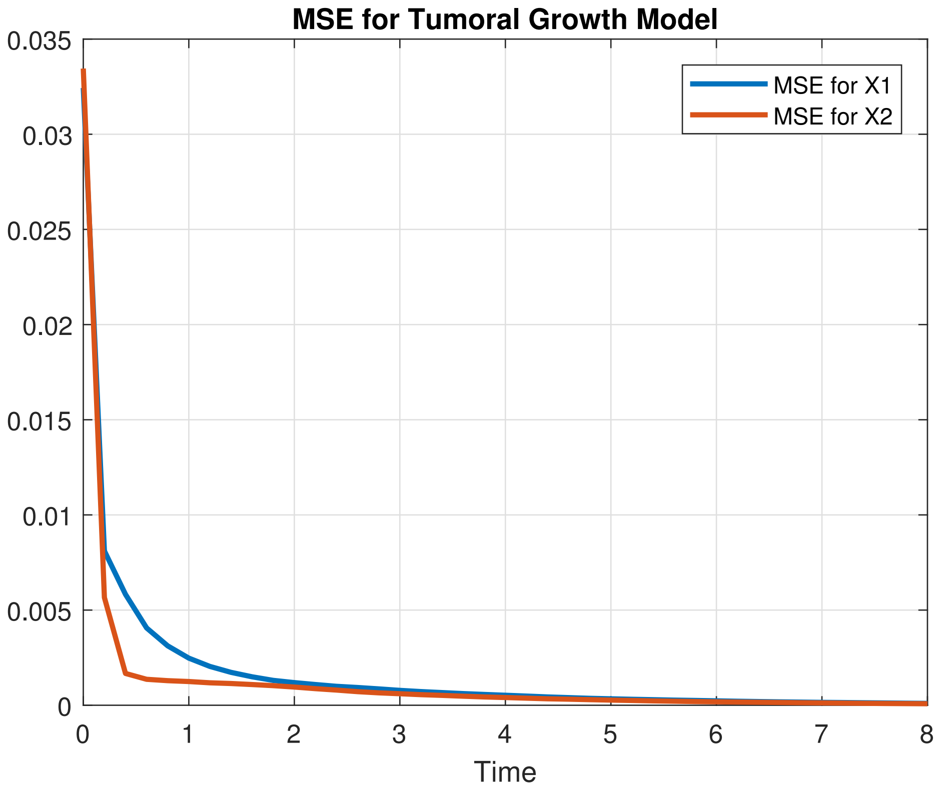

Figure 2.6: MSE for X1 and X2 in Equation (2.5) and (2.6). The MSE has a sharp decreasing after the first time step.

Chapter 3

Particle Flow Filter

When dealing with non-linear discrete dynamic models, Particle filters are often used to get predictions with Bayesian statistical inference. Particle Filters have the problem of particle degeneracy caused by the Bayesian Rule, especially in dealing with high dimensional state vectors.

Particle flow filter is derived from improving the estimation accuracy in high-dimensional space by solving a transport map of particles and it significantly avoids the problem of degeneracy. We set each particle in d-dimensional space as a function of λ denoting as $x(\lambda)$, in which lambda is continuously changing just and λ starts from 0 and ends up with 1 which represents the location of particle filters at the next time step.

In the next section, we present a particle flow filter for the two models in Chapter 2.

3.1 Generalized Method for Stochastic Particle Flow Filters

We start by constructing the stochastic differential equation for the flow of particles [1].

$$dx = f(x, \lambda)d\lambda + Q(x)^{\frac{1}{2}}dW_{\lambda}. \tag{3.1}$$

Here $f(x, \lambda)$ is the particle flow function and $Q(x)$ is the covariance matrix of the diffusion W_λ. W_λ is the measurement noise generated according to λ.

In order to get the solution of $f(x, \lambda)$ and $Q(x)$, probability density function $P(x, \lambda)$ is essential to be introduced. We set

$$\log\ P(x, \lambda) = \log\ g(x) + \lambda \log\ h(x) - \log\ K(\lambda). \tag{3.2}$$

The generalized probability density function has the form of

$$p(x, \lambda) = \frac{g(x)h(x)^\lambda}{\int_{\mathbb{R}^d} g(x)h(x)^\lambda\, dx} = \frac{g(x)h(x)^\lambda}{K(\lambda)}, \tag{3.3}$$

in which $h(x)$ is the likelihood function, $g(x)$ is the prior density of x and $K(\lambda)$ is the integration of product of $g(x)$ and $h(x)^\lambda$ with respect to x. The purpose of incorporating $K(\lambda)$ is to normalize the conditional probability density.

By Equation (3.2), one can solve $f(x, \lambda)$ by setting specific $Q(x)$ to simplify the partial differentiation equation (PDE) for $f(x, \lambda)$, which is of the following form.

$$\frac{\partial \log\ h}{\partial x} = -f^T \frac{\partial^2 \log\ P}{\partial x^2} - \frac{\partial div(f)}{\partial x} - \frac{\partial \log\ P}{\partial x}\frac{\partial f}{\partial x} + \frac{\partial[\mathrm{div}(Q\frac{\partial P}{\partial x})/2P]}{\partial x}. \tag{3.4}$$

The simplest way is to set one part of the equation equals to zero,which is

$$-\frac{\partial div(f)}{\partial x} - \frac{\partial \log\ P}{\partial x}\frac{\partial f}{\partial x} + \frac{\partial[div(Q\frac{\partial P}{\partial x})/2P]}{\partial x} = 0. \tag{3.5}$$

Then $f(x, \lambda)$ has the form

$$f(x, \lambda) = -[\frac{\partial^2 \log\ P(x, \lambda)}{\partial x^2}]^{-1}(\frac{\partial \log h(x)}{\partial x})^T. \tag{3.6}$$

According to Equation (3.5), the corresponding covariance function $Q(x)$ has the form

$$Q = [P - \lambda PH^T(R + \lambda HPH^T)^{-1}HP]H^TR^{-1}H[P - \lambda PH^T(R + \lambda HPH^T)^{-1}HP], \quad (3.7)$$

where R is the measurement noise covariance matrix, P is the prior covariance matrix, and H is the sensitive matrix in the measurement model. To make sure the solution of $Q(x)$ from equation (3.7) is a symmetric matrix, one should implement the following method to symmetry Q, which is

$$\hat{Q} = \frac{Q + Q^T}{2}. \quad (3.8)$$

We always use Q to substitute $\hat{Q}$ in this work.

Algorithm 3.1. *(Algorithm for implementing particle flow Filter with diffusion)*

- a. Use Monte Carlo method randomly choose N particles around observation, and generate particle density function $g(x)$ as prior density function.
- b. Choose a suitable $h(x)$ as likelihood function.
- c. Compute $p(x, \lambda)$ according to (3.3), that is, $p(x, \lambda) = \frac{g(x)h(x)^\lambda}{K(\lambda)}$, where $K(\lambda) = \int_{\mathbb{R}^d} g(x)h(x)^\lambda \, dx$.
- d. Solve function $f(x, \lambda)$ and measurement covariance matrix $Q(x)$ according to (3.6) and (3.7), which is

$$\begin{aligned} f(x, \lambda) &= -[\frac{\partial^2 \log \ P(x, \lambda)}{\partial x^2}]^{-1}(\frac{\partial \log h(x)}{\partial x})^T, \\ Q(x) &= [P - \lambda PH^T(R + \lambda HPH^T)^{-1}HP]H^TR^{-1}H[P - \lambda PH^T(R + \lambda HPH^T)^{-1}HP]. \end{aligned}$$

- e. Plugging the value of $f(x, \lambda)$ and $Q(x)$, one can derive x by solving the PDE:

$dx = f(x,\lambda)d\lambda + LdW_\lambda$, where $L = chol(Q)$. Use forward Euler scheme

$$x^{(n+1)} = x^{(n)} + f(x^{(n)}, \lambda_n)\Delta\lambda + L\Delta W_\lambda,$$

or implicit Euler scheme

$$x^{(n+1)} = x^{(n)} + f(x^{(n+1)}, \lambda_{n+1})\Delta\lambda + L\Delta W_\lambda.$$

- f. For updating each point, repeat steps from a to e.

Remark 3.2. *Here $h(x)$ can be any distribution, but we consider normal distribution with estimated mean and covariance.*

Remark 3.3. *The use of either explicit or implicit Euler method depends on the shape of $f(x,\lambda)$.*

3.2 Implementations of Particle Flow Filter

Here we consider the particle flow filter for two nonlinear models. One is the S&P model from [4] and the other is a perturbed tumor growth model from [2]. Both of the examples are non-linear discrete models, with non-Gaussian distributed noise term. In the next two sections, we present how we implement the algorithms and numerical results.

3.2.1 Implementation and Results for the S&P model

In our previous dividend yield and S&P real return model, we have the measurement models in the form of

$$Z_n = \begin{pmatrix} X_n \\ \delta R_n \end{pmatrix} = \begin{pmatrix} \frac{1}{1+k}X_{n-1} + \frac{k\theta}{1+k} + \frac{\sigma}{1+k}\sqrt{X_{n-1}}\Delta W_{1,n} \\ \mu X_n + a\sqrt{X_{n-1}}\left(\rho\Delta W_{1,n} + \sqrt{1-\rho^2}\Delta W_{2,n}\right) \end{pmatrix},$$

and measurement model in the form of

$$Y_n = \begin{pmatrix} Y_{1,n} \\ Y_{2,n} \end{pmatrix} = \begin{pmatrix} X_n + Q_1 B_{1,n} \\ \delta R_n + Q_2 B_{2,n} \end{pmatrix}.$$

The prior density function which is

$$g(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} e^{-\frac{(x-\mu)^T \Sigma_1^{-1}(x-\mu)}{2}}, \tag{3.9}$$

where μ is sample mean, and Σ_1 is sample covariance with

$$\Sigma_1 = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}.$$

We assume a normally distributed likelihood function as

$$h(x_1, x_2) = \frac{1}{2\pi\sqrt{|\Sigma_2|}} e^{-\frac{(x-m)^T \Sigma_2^{-1}(x-m)}{2}}, \tag{3.10}$$

where m is probability mean and Σ_2 is probability covariance. Therefore, the conditional probability density function $P(x, \lambda)$ in our S&P model has the following equation, which is

$$p(x, \lambda) = \frac{g(x)h(x)^\lambda}{K(\lambda)} = \frac{1}{K(\lambda)} e^{-\frac{(x-\mu)^T \Sigma_1^{-1}(x-\mu) + \lambda(x-m)^T \Sigma_2^{-1}(x-m)}{2}}, \tag{3.11}$$

where

$$K(\lambda) = \int_{\mathbb{R}} g(x)h(x)^\lambda \, dx = \frac{1}{(2\pi)^2\sqrt{|\Sigma_1|}\sqrt{|\Sigma_2|}} \int_{\mathbb{R}} e^{-\frac{(x-\mu)^T \Sigma_1^{-1}(x-\mu) + \lambda(x-m)^T \Sigma_2^{-1}(x-m)}{2}} \, dx.$$

By implementing log function on both side of $P(x,\lambda)$ in (3.11), we get

$$\log P(x,\lambda) = -\frac{(x-\mu)^T\Sigma_1^{-1}(x-\mu) + \lambda(x-m)^T\Sigma_2^{-1}(x-m)}{2} - \log(K(\lambda)). \qquad (3.12)$$

By implementing *log* function on both side of $h(x)$ in (3.21), we get

$$\log\ h(x) = \log(\frac{1}{2\pi(\Sigma_2)^{1/2}}) - \frac{(x-m)^T\Sigma_2^{-1}(x-m)}{2}. \qquad (3.13)$$

Then one can get the particle flow function $f(x)$ with the following form

$$f(x,\lambda) = -[\frac{\partial^2 \log\ P(x,\lambda)}{\partial^2 x}]^{-1}(\frac{\partial \log\ h(x)}{\partial x}) = -[-\Sigma_1^{-1} - \lambda\Sigma_2^{-1}]^{-1}[-\Sigma_2^{-1}(x-m)], \quad (3.14)$$

where $\frac{\partial^2(logP(x,\lambda))}{\partial x^2} = -\Sigma_1^{-1} - \lambda\Sigma_2^{-1}$, $\quad \frac{\partial(log\ h(x))}{\partial x} = -\Sigma_2^{-1}(x-m)$.

According to equation (3.7), the corresponding $Q(x)$, in this case, has the following form

$$Q(x) = [P - \lambda P(V+\lambda P)^{-1}P]V^{-1}[P - \lambda P(V+\lambda P)^{-1}P], \qquad (3.15)$$

where P is the prior covariance, which has the form from Kalman filter (1.11)

$$P_{n(-)} = \Phi P_{n-1(+)}\Phi^T + MZ^{(n-1)}CC^T.$$

Then we can update x with respect to λ by using Backward Euler

$$x^{(n+1)} = x^{(n)} + f(x^{(n+1)},\lambda_{n+1})\Delta\lambda + L(x^{(n)})\Delta W_\lambda, \qquad (3.16)$$

where $L(x) = \sqrt{Q(x)}$.

Subtracting m from each side of the equation (3.24), one can get

$$x^{(n+1)} - m = (x^{(n)} - m) - [\Sigma_1^{-1} + \lambda_{n+1}\Sigma_2^{-1}]^{-1}\Sigma_2^{-1}(x^{(n+1)} - m)\Delta\lambda + L(x^{(n)})\Delta W_\lambda.$$

Set $y^{(n+1)} = x^{(n+1)} - m, y^{(n)} = x^{(n)} - m$, and the equation becomes

$$y^{(n+1)} = y^{(n)} - [\Sigma_1^{-1} + \lambda\Sigma_2^{-1}]^{-1}\Sigma_2^{-1}y^{(n+1)}\Delta\lambda + L(x)\Delta W_\lambda.$$

Finally we can calculate $x_{(n+1)}$ as

$$x^{(n+1)} = (I + \Delta\lambda[\Sigma_1^{-1} + \lambda\Sigma_2^{-1}]^{-1}\Sigma_2^{-1})^{-1}(x^{(n)} - m + L(x)\Delta W_\lambda) + m. \tag{3.17}$$

We set the total points to 100, and time discretization to 0.02 and the initial covariance for x_0 to 0. When we are not able to factorize of covariance, we set L as the square root of x. By plugging observed yield and return from the year 1945 to the year 2010 to Y_n, one can repeat the algorithm for 65 times. Each time we update mean and covariance of $g(x)$, mean and covariance of $h(x)$ and covariance matrix $Q(x)$, and record $x^{(n+1)}$ each step as the predicted value. The specific Matlab code for implementing particle flow filter on the S&P model can be found in Appendix E. Figure 3.1 and 3.2 show the comparison between real value and estimated value with the same time discretization.

We present the results of particle flow particle method (3.17) in the next section.

Results of S&P model

We also calculate the mean squared error(MSE) of yield and return for particle flow filter and follow the equation (2.4) and present the values of MSE in Figure 3.3.

From Figure 3.1 and Figure 3.2, one can see that the predicted yield matches perfectly with the real yield, and prediction for return has excellent performance at the years with significant fluctuation but less well at the year with lower variation. The reason is that the MSE for predicted return is much larger than the MSE for predicted yield, which means the range for sigma points of return is much broader. Therefore, the prediction has more substantial covariance. In summary, by involving the function of $f(x, \lambda)$, the accuracy of predictions have been highly increased, and it perform better than the unscented filter.

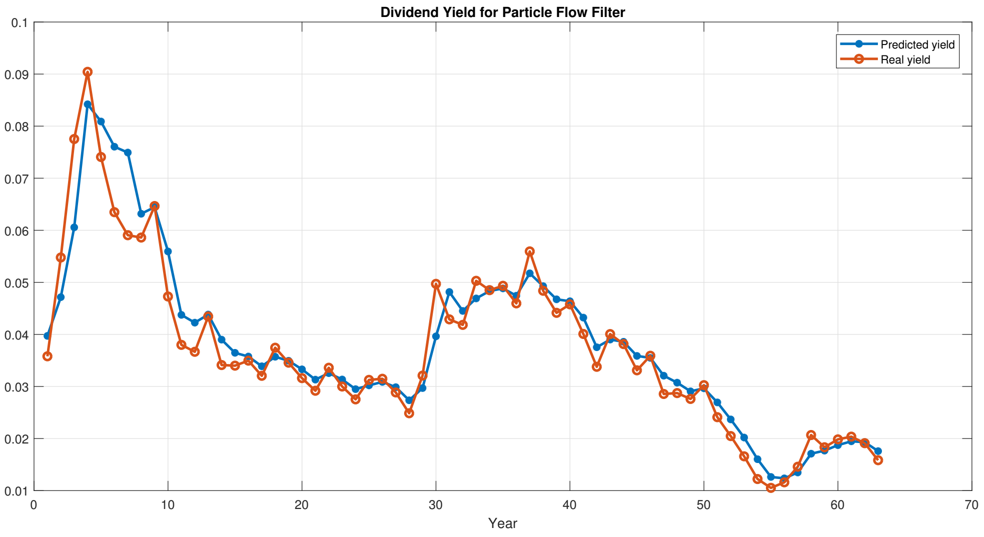

Figure 3.1: Particle Flow Filter for Yield in Equation (1.8) and (1.9). Prediction of yield captures real yield well with few mismatches.

3.2.2 Implementation and Results for Perturbed Tumor Growth Model

In our previous perturbed tumoral growth model, we have the measurement models in the form of

$$d\mathbf{X}_t = F(\mathbf{X}_t)dt + \sigma dW_t, \tag{3.18}$$

where f_1 and f_2 are defined by

$$f_1(\mathbf{X}_t) = \alpha_1 X_t^1 ln(\frac{X_t^2}{X_t^1}),$$
$$f_2(\mathbf{X}_t) = \alpha_2 X_t^1 - \alpha_3 X_t^2 (X_t^1)^{\frac{2}{3}},$$

and measurement model has the form of

$$Y_k = (X_k^1, X_k^2)^T + R\mathbf{v}_k. \tag{3.19}$$

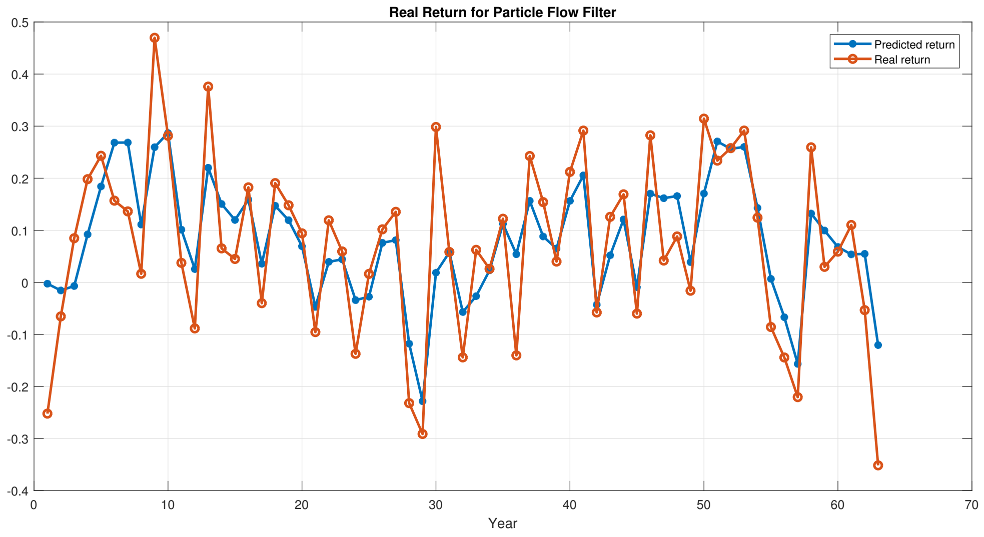

Figure 3.2: Particle Flow Filter Results for Yield in Equation (1.8) and (1.9). Predicted return captures the real return well with larger fluctuations.

In the particle flow particle method, we have

$$g(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} e^{-\frac{(x-\mu)^T\Sigma_1^{-1}(x-\mu)}{2}}, \tag{3.20}$$

and we set $h(x)$ as

$$h(x_1, x_2) = \frac{1}{2\pi\sqrt{|\Sigma_2|}} e^{-\frac{(x-m)^T\Sigma_2^{-1}(x-m)}{2}}, \tag{3.21}$$

Then we can derive $f(x, \lambda)$ as follows

$$f(x,\lambda) = -[\frac{\partial^2 \log\ P(x,\lambda)}{\partial^2 x}]^{-1}(\frac{\partial \log\ h(x)}{\partial x}) = -[-\Sigma_1^{-1} - \lambda\Sigma_2^{-1}]^{-1}[-\Sigma_2^{-1}(x-m)], \tag{3.22}$$

and $Q(x)$

$$Q(x) = [P - \lambda P(V + \lambda P)^{-1}P]V^{-1}[P - \lambda P(V + \lambda P)^{-1}P]. \tag{3.23}$$

Thus, the updated equation for x is

$$x^{(n+1)} = x^{(n)} + f(x^{(n+1)}, \lambda_{n+1})\Delta\lambda + L(x^{(n)})\Delta W_\lambda. \tag{3.24}$$

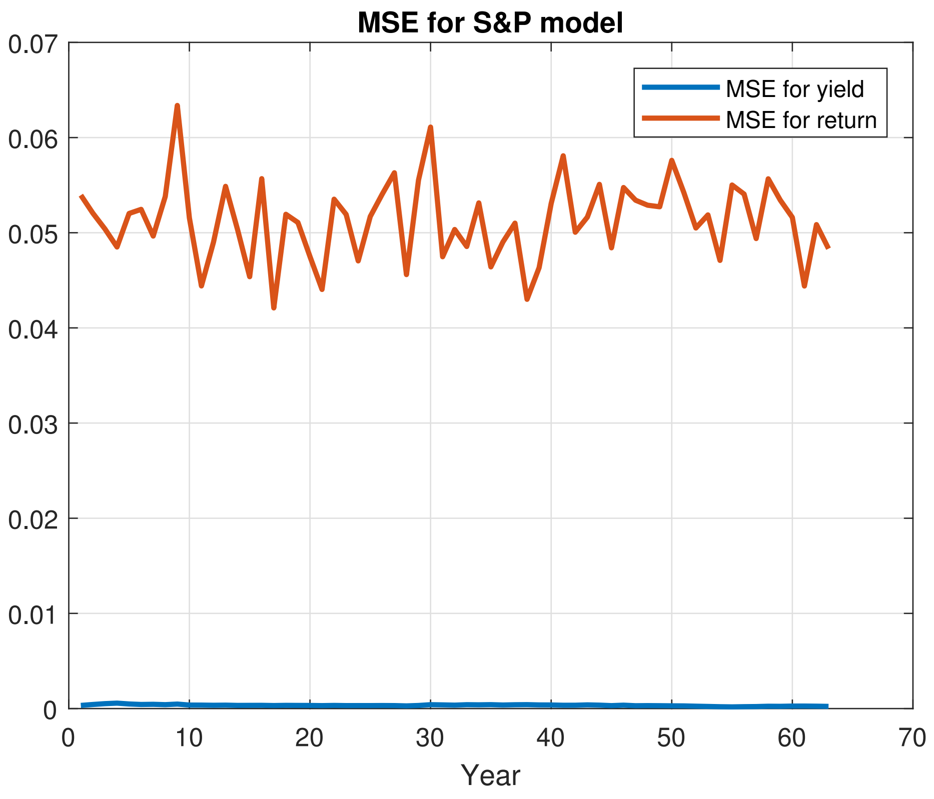

Figure 3.3: MSE for Yield and Return in Equation (1.8) and (1.9) with Particle Flow filter. The MSE for return is much higher than the MSE for yield.

where $L(x) = \sqrt{Q(x)}$.

We set the total points to be 1500, and time discretization to 0.04 and the initial covariance for x_0 to 0. When we are not able to factorize of covariance, we set L as the square root of x. By plugging observed X1 and X2 from time 0 to 8 to Y_n, one can repeat the algorithm for 40 times. Each time we update mean and covariance of $g(x)$, mean and covariance of $h(x)$ and covariance matrix $Q(x)$, and record $x^{(n+1)}$ each step as the predicted value. The specific Matlab code for implementing particle flow filter on the S&P model can be found in Appendix F. Figure 3.4 and 3.5 show the comparison between real value and estimated value with the same time discretization.

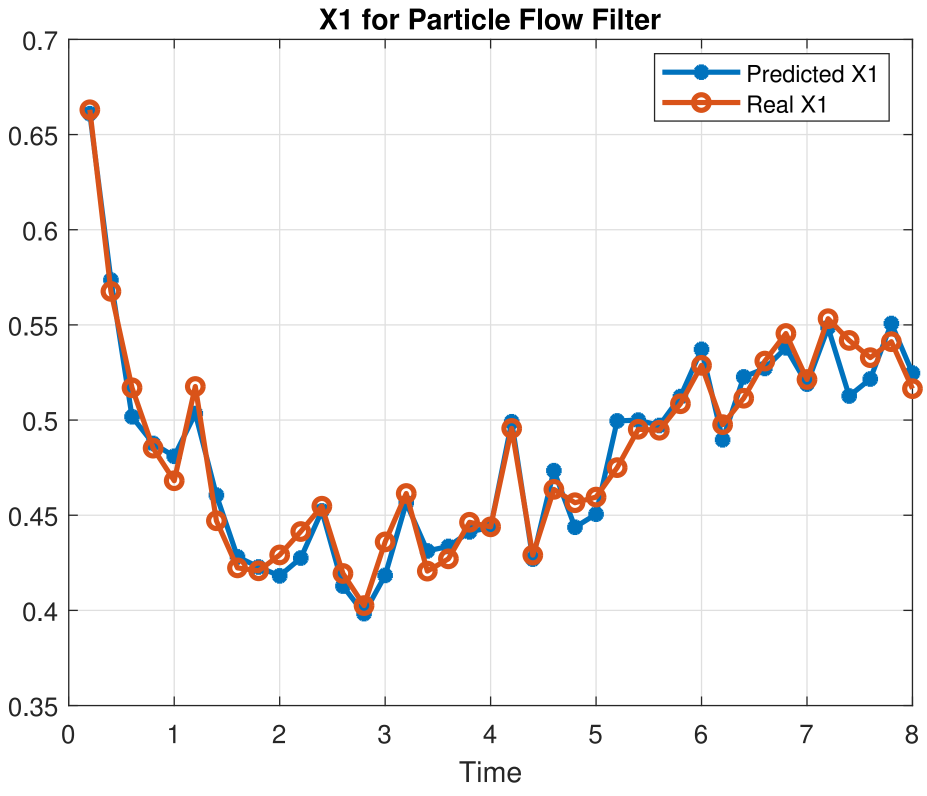

Figure 3.4: Particle Flow Filter for X1 in Equation (2.5) and (2.6). Predicted X1 captures real X1 well with few mismatches.

Results for perturbed tumor growth model

We also calculate the mean squared error(MSE) of X1 and X2 by using particle flow filter and we follow the equation (2.4). The result is given in Figure 3.6.

From Figure 3.4 and Figure 3.5, one can see that the predicted X1 and X2 match perfectly with the real X1 and X2. The MSE of X1 and X2 are reliably acceptable. The MSE for both X1 and X2 with particle flow filter are much smaller than the MSE for X1 and X2 with unscented Kalman filter, which means the particles in particle flow filter are more concentrated. The choice of the function $f(x, \lambda)$ increases the accuracy of predictions and suggests the validity of the particle flow particle method.

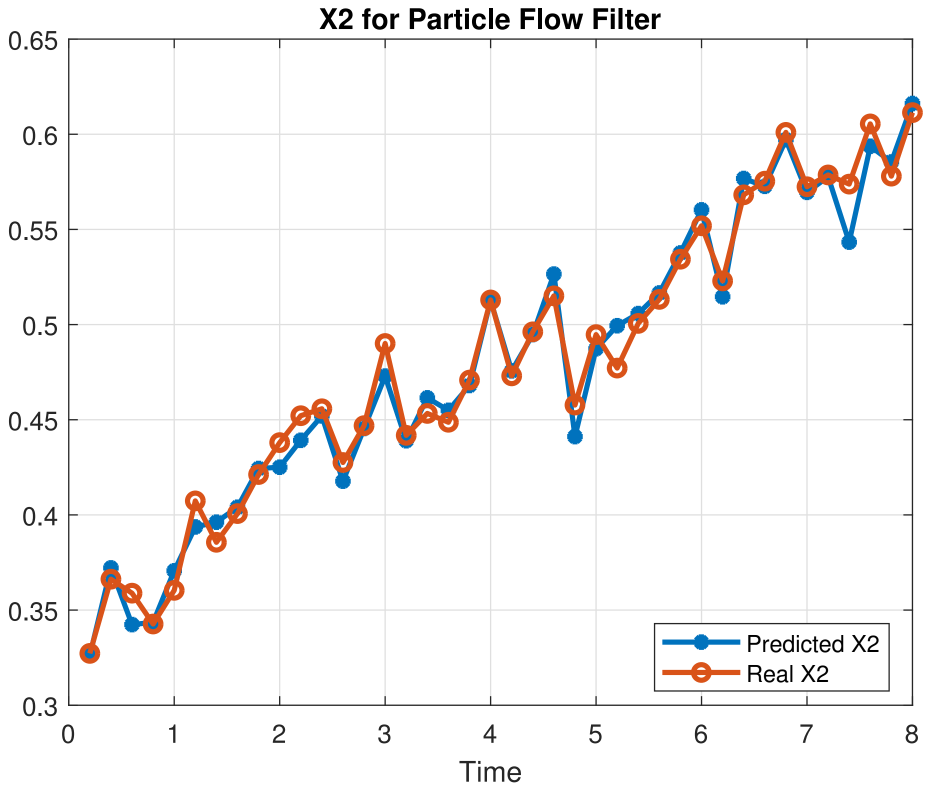

Figure 3.5: Particle Flow Filter for X2 in Equation (2.5) and (2.6). Predicted X2 captures real X2 well with few mismatches.

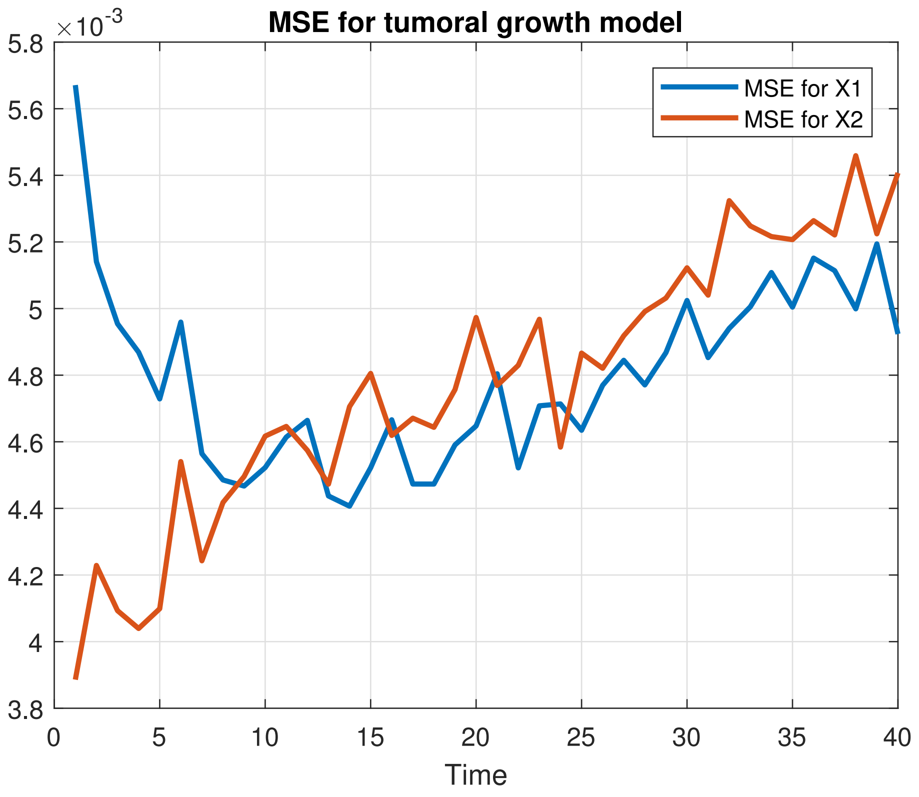

Figure 3.6: MSE for X1 and X2 in Equation (2.5) and (2.6) with Particle Flow filter. The MSE is acceptably small.

Chapter 4

Conclusion

In this paper, we mainly discuss three types of filters: Kalman filter, unscented Kalman filter, and particle flow filters. We implement them on the two dynamically non-linear models and make comparison among the three methods.

In Chapter 1, we implement Kalman filter on the S&P model to predict dividend yield and real return from the year 1945 to the year 2010. We applied the Kalman filter but Kalman filter performs well only in the first steps with capturing the pattern of change for our S&P model because the observation model is a non-linear model, i.e., a non-Gaussian one.

In Chapter 2 and 3, we use non-linear filters to improve the prediction, which are unscented Kalman filter and particle flow filter respectively. We perform the two models for each of the method: the S&P model and the perturbed tumor growth model. For unscented Kalman filter, the key is to generate sigma points around the priori mean, and then update predicted mean and covariance at each iteration. The result of implementing S&P model with unscented Kalman filter is better than those by the Kalman filter. Predicted yield capture most of the trend for whole process with single points mismatched with the real values. But the predicted return does not match the real return so well. For two-dimensional tumor growth model, unscented Kalman filter outperforms the Kalman filter.

In Chapter 3, we perform the same two models with particle flow filter as in Chapter 2. Particle flow filter is moving the particles by particle flow function which is generated properly from partial differentiation equations with properly chosen likelihood functions. By solving the partial differentiation equations, we can construct the explicit dynamic model on how to move particles. For the the S&P model, the particle flow filter outperforms Kalman filter and unscented Kalman filter. For the two-dimensional tumor growth model, particle flow filter performs better than Kalman filter. The predicted X1 and X2 match the real values well with stable MSE.

In conclusion, Kalman filter does not works so well in our S&P model as the model is non-linear, which matches our expectation and is long known. It seems that particle flow filter works better than unscented filter for our two models. It is a bit surprising that the particle flow filter works so well in our S&P model even though we chose a Gaussian likelihood function with mean and variance the same as the prior. The reason why it works so well is not clear to us. Also, the effects of different likelihood function $h(x)$ is unknown. We would like to investigate such issues in the future and apply the particle flow filters to more complicated non-linear models in high dimension.

Bibliography

[1] Fred Daum, Jim Huang, and Arjang Noushin. Generalized gromov method for stochastic particle flow filters. In *Signal Processing, Sensor/Information Fusion, and Target Recognition XXVI*, volume 10200, page 102000I. International Society for Optics and Photonics, 2017.

[2] Clayton G. Webster Guannan Zhang Feng Bao, Yanzhao Cao. An efficient mesh-free implicit filter for nonlinear filtering problems. *International Journal for Uncertainty Quantification*, 6(1):19–33, 2016.

[3] Mohinder S Grewal and Angus P Andrews. Kalman filtering: theory and practice using MATLAB. 2001.

[4] Paolo Guasoni and Gu Wang. Consumption in incomplete markets. *Available at SSRN: https://ssrn.com/abstract=2411236 or http://dx.doi.org/10.2139/ssrn.2411236.*

[5] Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[6] Mu Niu Qingyun Ren. Parameters prediction on dividend yield and s&p real return model.

Appendix A

Steps to Solve S&P Model with Kalman Filter

In this section, we present the details of deriving the Kalman filter for the the S&P Model described in Chapter 1.

A.1 Step 1. Solving for Kalman Gain

Kalman gain can be expressed as

$$\bar{K}_n = P_{n(-)} H_n^T (H_n P_{n(-)} H_n^T + V^2)^{-1}.$$

Below we present how we derive Kalman Gain. The optimal updated estimate $\hat{Z}_{n(+)}$ is a linear function of a priori estimate $\hat{Z}_{n(-)}$ and measurement Y_k, that is,

$$\hat{Z}_{n(+)} = K_n^1 \hat{Z}_{n(-)} + \bar{K}_n Y_n,$$

where K_n^1 and $\bar{K}_n$ are unknown yet. We seek values of K_n^1 and $\bar{K}_n$ such that the estimate $\hat{Z}_{n(+)}$ satisfies the orthogonality principle:

$$E\langle [Z_n - \hat{Z}_{n(+)}] Y_i^T \rangle = 0, \quad i = 1, 2, ...n-1.$$

If one expands Z_n from (1.2) and $Z_{n(+)}$ from (1.1) into (A.1), then one will observe

$$E\langle [\Phi Z_{n-1} + D + M\sqrt{Z_{n-1}} C W_n - K_n^1 \hat{Z}_{n(-)} - \bar{K}_n Y_n] Y_i^T \rangle = 0, \quad \text{for } i = 1, 2, ...n-1.$$

Since W_n and V_n are uncorrelated, it follows that $E\langle W_n Y_i^T \rangle = 0 \; for \; 1 \leq i \leq n-1.$

Using this result, one can obtain the following,

$$E\langle[\Phi Z_{n-1} + D - K_n^1 \hat{Z}_{n(-)} - \bar{K}_n Y_n] Y_i^T\rangle = 0, \quad for\ i = 1, 2, ...n-1.$$

By substituting Y_n using (1.9), one can get

$$E\langle[\Phi Z_{n-1} + D - K_n^1 \hat{Z}_{n(-)} - \bar{K}_n H Z_n - \bar{K}_n V B_n] Y_i^T\rangle = 0, \quad i = 1, 2, ...n-1.$$

Expanding the expectation, we will have

$$\Phi E\langle Z_{n-1} Y_i^T\rangle + D E\langle Y_i^T\rangle - K_n^1 E\langle \hat{Z}_{n(-)} Y_i^T\rangle - \bar{K}_n H E\langle Z_n Y_i^T\rangle - \bar{K}_n V E\langle B_n Y_i^T\rangle = 0. \tag{A.1}$$

We also know that

$$E\langle B_n Y_i^T\rangle = 0, \quad \text{for } i = 1, 2, ...n-1.$$

Thus, (A.1) can be rewrite as:

$$\Phi_{n-1} E\langle Z_{n-1} Y_i^T\rangle + D E\langle Y_i^T\rangle - K_n^1 E\langle \hat{Z}_{n(-)} Y_i^T\rangle - \bar{K}_n H_n E\langle Z_n Y_i^T\rangle = 0.$$

And can be simplified as

$$E\langle[I - K_n^1 - \bar{K}_n H_n]\rangle E\langle Z_n Y_i^T\rangle = 0.$$

(A.1) can be satisfied for any given Z_n if

$$K_n^1 = I - \bar{K}_n H,$$

Thus, K_n^1 in (A.1) satisfied with (A.1).

We define estimation errors as

$$\begin{aligned} \tilde{Z}_{n(+)} &\triangleq \hat{Z}_{n(+)} - Z_n, \\ \tilde{Z}_{n(-)} &\triangleq \hat{Z}_{n(-)} - Z_n, \\ \tilde{Y}_n &\triangleq \hat{Y}_{n(-)} - Y_n = H_n Z_{n(-)} + Y_n. \end{aligned}$$

Since $\tilde{Y}_{n(-)}$ depends linearly on Y_n, from (A.2), we have

$$E\langle[Z_n - \hat{Z}_{n(+)}]\tilde{Y}_n^T\rangle = 0.$$

Substitute Z_n, $\hat{Z}_{n(+)}$, and $\tilde{Y}_n$ from (1.8), (A.1), and (A.2) respectively. Then

$$E\langle\Phi_{n-1} Z_{n-1} + D + M\sqrt{Z_{n-1}} C W_n - K_n^1 \hat{Z}_{n(-)} - \bar{K}_n Y_n][H_n \hat{Z}_{n(-)} - Y_n]^T\rangle = 0.$$

By the orthogonality of

$$E\langle W_n Y_n^T\rangle = E\langle W_n X_{n(-)}^T\rangle = 0,$$

we obtain

$$E\langle\Phi_{n-1} Z_{n-1} + D - K_n^1 \hat{Z}_{n(-)} - \bar{K}_n Y_n][H_n \hat{Z}_{n(-)} - Y_n]^T\rangle = 0.$$

Substituting for K_n^1, Y_n and using (A.1)

$$E\langle \Phi_{n-1}Z_{n-1} + D - \hat{Z}_{n(-)} + \bar{K}H_n\hat{Z}_{n(-)} - \bar{K}_nZ_n - \bar{K}_nVB_n][H_n\hat{Z}_{n(-)} - H_nZ_n - VB_n]^T\rangle = 0.$$

Then combine terms of $Z_n - \hat{Z}_{n(-)}$, we get

$$E\langle[-\tilde{Z}_{n(-)} + \bar{K}_nH_n\tilde{Z}_{n(-)} - \bar{K}_nVB_n][H_n\tilde{Z}_{n(-)} - VB_n]^T\rangle = 0.$$

Using the fact that $E\langle\tilde{Z}_{n(-)}B_n^T\rangle = E\langle B_n^T\tilde{Z}_{n(-)}^T\rangle = 0$, the last result will be as following:

$$(-I + \bar{K}_nH_n)E\langle\tilde{Z}_{n(-)}\tilde{Z}_{n(-)}^T\rangle H_n^T + \bar{K}_nVE\langle B_nB_n^T\rangle V^T = 0. \tag{A.2}$$

For the second term of (A.2), $\bar{K}_nVE\langle B_nB_n^T\rangle V^T$ have the matrix form:

$$\bar{K}_n\begin{pmatrix}Q_1 & 0\\ 0 & Q_2\end{pmatrix}E\begin{pmatrix}B_{1n}^2 & B_{2n}B_{1n}\\ B_{1n}B_{2n} & B_{2n}^2\end{pmatrix}\begin{pmatrix}Q_1 & 0\\ 0 & Q_2\end{pmatrix} = \bar{K}_n\begin{pmatrix}Q_1 & 0\\ 0 & Q_2\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}Q_1 & 0\\ 0 & Q_2\end{pmatrix} = \bar{K}_nV^2.$$

Plugging the value of (A.1) to (A.2):

$$(-I + \bar{K}_nH_n)E\langle\tilde{Z}_{n(-)}\tilde{Z}_{n(-)}^T\rangle H_n^T + \bar{K}_nV^2 = 0.$$

By definition, the error covariance matrix is $P_{n(-)} = E\langle\tilde{Z}_{n(-)}\tilde{Z}_{n(-)}^T\rangle$, it satisfies the equation:

$$(-I + \bar{K}_nH_n)P_{n(-)}H_n^T + \bar{K}_nV^2 = 0,$$

further we have

$$\bar{K}_n(H_nP_{n(-)}H_n^T + V^2) = P_{n(-)}H_n^T.$$

Therefore, Kalman gain can be expressed as

$$\bar{K}_n = P_{n(-)}H_n^T(H_nP_{n(-)}H_n^T + V^2)^{-1}. \tag{A.3}$$

which is the solution we want to seek as a function of priori covariance before update.

A.2 Step 2. Solving for Priori and Posterior Estimation

By definition, the priori estimation

$$\hat{Z}_{n(-)} = \Phi_{n-1}\hat{Z}_{n(+)} + D.$$

By substituting (A.1) into (A.1), one can obtain the equations

$$\hat{Z}_{n(+)} = (I - \bar{K}_nH_n)\hat{Z}_{n(-)} + \bar{K}_nY_n.$$

Therefore, the posterior estimation we want to seek is a function of priori estimation and Kalman gain.

$$\hat{Z}_{n(+)} = \hat{Z}_{n(-)} + \bar{K}_n(-H_n\hat{Z}_{n(-)} + Y_n). \tag{A.4}$$

A.3 Step 3. Solving for Priori and Posterior Covariance

The priori and posterior covariance have the form of

$$P_{n(-)} = \Phi_{n-1}P_{n-1(+)}\Phi_{n-1}^T + X_{n-1}CC^T,$$

and

$$P_{n(+)} = (I - \bar{K}_nH_n)P_{n(-)}.$$

Below is the process of how we derive these two equations.

One can derive a formula for posterior covariance, which is

$$P_{n(+)} = E\langle\tilde{Z}_{n(+)}\tilde{Z}_{n(+)}^T\rangle.$$

By plugging equation (A.4) to (A.1), one can obtain the equations

$$\begin{aligned}
\tilde{Z}_{n(+)} &= \hat{Z}_{n(+)} - Z_n = \hat{Z}_{n(-)} - \bar{K}_nH_n\hat{Z}_{n(-)} + \bar{K}_nY_n - Z_n \\
&= \hat{Z}_{n(-)} - \bar{K}_nH_n\hat{Z}_{n(-)} + \bar{K}_nH_nZ_n - \bar{K}_nVB_n - Z_n \\
&= (\hat{Z}_{n(-)}Z_n) - \bar{K}_nH_n(\hat{Z}_{n(-)} - Z_n) + \bar{K}_nVB_n \\
&= (I - \bar{K}_nH_n)\tilde{Z}_{n(-)} + \bar{K}_nVB_n.
\end{aligned}$$

By substituting (A.5) into (A.3) and noting that $E\langle\tilde{Z}_{n(-)}B_n^T\rangle = 0$, one obtains

$$\begin{aligned}
P_{n(+)} &= E\langle[(I - \bar{K}_nH_n)\tilde{Z}_{n(-)} + \bar{K}_nVB_n][(I - \bar{K}_nH_n)\tilde{Z}_{n(-)} + \bar{K}_nVB_n]^T\rangle \\
&= E\langle(I - \bar{K}_nH_n)\tilde{Z}_{n(-)}\tilde{Z}_{n(-)}^T(I - \bar{K}_nH_n)^T + \bar{K}_nVB_nB_n^TV^T\bar{K}_n^T\rangle \\
&= (I - \bar{K}_nH_n)P_{n(-)}(I - \bar{K}_nH_n)^T + \bar{K}_nV^2\bar{K}_n^T \\
&= P_{n(-)} - \bar{K}_nH_nP_{n(-)} - P_{n(-)}H_n^T\bar{K}^T + \bar{K}_nH_nP_{n(-)}H_n^T\bar{K}_n^T + \bar{K}_nV^2\bar{K}_n^T \\
&= (I - \bar{K}_nH_n)P_{n(-)} - P_{n(-)}H_n^T\bar{K}_n^T + \bar{K}_n(H_nP_{n(-)}H_n^T + v^2)\bar{K}^T \\
&= (I - \bar{K}_nH_n)P_{n(-)} - P_{n(-)}H_n^T\bar{K}_n^T + P_{n(-)}H_n^T\bar{K}_n^T \\
&= (I - \bar{K}_nH_n)P_{n(-)}.
\end{aligned}$$

This is the final form of posterior covariance, which shows the effects of Kalman gains on priori covariance. Respectively, the definition of prior covariance

$$P_{n(-)} = E\langle\tilde{Z}_{n(-)}\tilde{Z}_{n(-)}^T\rangle.$$

By plugging (1.8) into (A.2), one can obtain the equations

$$\begin{aligned}\tilde{Z}_{n(-)} &= \Phi_{n-1}\hat{Z}_{n-1(+)} + D - Z_n \\ &= \Phi_{n-1}\hat{Z}_{n-1(+)} + D - Z_n - \Phi_{n-1}Z_{n-1} - D - M\sqrt{Z_{n-1}}CW_n \\ &= \Phi_{n-1}\tilde{Z}_{n-1(+)} - M\sqrt{Z_{n-1}}CW_n.\end{aligned}$$

Using the fact that $E\langle\tilde{Z}_{n-1}W_{n-1}^T\rangle$ to obtain the results

$$\begin{aligned}P_{n(-)} &= E\langle[\Phi_{n-1}\tilde{Z}_{n-1(+)} - M\sqrt{Zn-1}CW_n][\Phi_{n-1}\tilde{Z}_{n-1(+)} - M\sqrt{Zn-1}CW_n]^T\rangle \\ &= \Phi_{n-1}E\langle\tilde{Z}_{n-1(+)}\tilde{Z}_{n-1(+)}^T\rangle\Phi_{n-1}^T + \sqrt{X_{n-1}}CE\langle W_nW_n^T\rangle C^T\sqrt{X_{n-1}} \\ &= \Phi_{n-1}P_{n-1(+)}\Phi_{n-1}^T + X_{n-1}CC^T,\end{aligned}$$

which gives a prior value of the covariance matrix as a function of the previous posterior covariance.

Appendix B

Matlab Code for the S&P Model with Kalman Filter

```
% Miniproject for Kalman Filter
clear; close all; clc;
rng(50,'twister')
%% Initials
%parameters
Q1 = 2.0714; Q2 = 2.0451;
k = 0.3003; theta = 0.1907;
sigma = 0.9197; mu = 1.6309;
alpha = 0.0310; rho = −0.8857;

%matrices :
%Zn = Phi*Zn−1 + D + sqrt(Xn−1)*C*W
Phi = [1/(1+k) 0; mu/(1+k) 0];
D = [k*theta/(1+k) ; mu*k*theta/(1+k)];
C = [sigma/(1+k) 0; mu*sigma/(1+k)+alpha*rho alpha*sqrt(1−rho^2)];

%Yn = Hn*Zn + V*B
%Hn = eye(2,2);
V = [Q1 0; 0 Q2];
%observation data
Y_obs(1, :) = xlsread('Shiller data.xlsx','Sheet4','G3:G67'); %yield
Y_obs(2, :) = xlsread('Shiller data.xlsx','Sheet4','E3:E67'); %real return
%Initial Z0
Z = [Y_obs(1,1)−Q1*randn; Y_obs(2,1)−Q2*randn];

%% Kalman filter
Z_post = zeros(2,66); %Set Z_posterior vector to store results
Z_post(:,1) = Z;
P_post = zeros(2,2); %Initial P posterior
```

```
for i = 2:66
    %Step1: Compute variance P prior from year 1946 to 2010
    %Pn(-) = Phi*Pn-1(+)*Phi' + (Xn-1)*C*C'
     P_prior = Phi * P_post * Phi' + Z_post(1, i-1).* C*C';

    %Step2: Compute kalman gain K
    %K = Pn(-)*Hn'*[Hn*Pn(-)*Hn' + V^2]^(-1) where Hn is an identity matrix
    K = P_prior * inv(P_prior + V^2);

    %Step3: Compute variance P posterior
    %Pn(+)=(I-K*Hn)*Pn(-) where Hn is an identity matrix
     I = eye(2,2);
     P_post = (I-K)*P_prior;

    %Step4: Compute Z prior
    %Zn(-) = Phi(n-1)*Zn-1(+) +D
     Z_prior = Phi*Z_post(:,i-1) + D;

    %Step5: Compute Z posterior
    %Zn(+)=Zn(-)+K*(Yn-Hn*Zn(-)) where Hn is an identity
     Z_post(:,i) = Z_prior + K*(Y_obs(:,i-1) - Z_prior);
end

figure(1);
plot(Z_post(1,2:66),'*-', 'linewidth',2);
hold on;
title('Dividend yield for Kalman Filter');
plot(Y_obs(1,:), '*-', 'linewidth',2);
legend('Estimated yield', 'Real yield', 'location','best');
grid on;
grid minor;
hold off;
xlabel('Year');
%saveas(gcf,'Yield','epsc');

figure(2);
plot(Z_post(2,2:66),'*-','linewidth', 2);
hold on;
title('Real return for Kalman Filter');
plot(Y_obs(2,:), 'o-','linewidth', 2);
legend('Estimated return', 'Real return');
grid on;
grid minor;
hold off;
```

```
 xlabel('Year');
%saveas(gcf,'Return','epsc');
```

Appendix C

Matlab Code for the S&P model with Unscented Kalman Filter

```
%Unscented Kalman filter for S&P model
close all; clear; clc;
rng(50,'twister')
%% Initials
%parameters:
Q1 = 0.0002; Q2 = 0.5;
k = 0.088; theta = 0.035;
sigma = 0.0005; mu=1.5;
a = 0.5; rho= -0.16;

%matrices:
%Zn = Phi*Zn-1 + D + sqrt(Xn-1)*C*W
Phi = [1/(1+k) 0; mu/(1+k) 0];
D = [k*theta/(1+k) ; mu*k*theta/(1+k)];
C = [sigma/(1+k) 0; mu*sigma/(1+k)+a*rho a*sqrt(1-rho^2)];

%observation data:
Y(:,1) = xlsread('Shiller data.xlsx','Sheet4','G2:G65'); % dividend yield
Y(:,2) = xlsread('Shiller data.xlsx','Sheet4','E2:E65'); %real return

%Set weights
omega = 1/3; N = 400; w = (1-omega)/N;
weights = [omega, ones(1,N)*w];

%% unscented filter
%Initial mean and covariance
mu_X = [Y(1,1) - Q1*randn(1); Y(2,1) - Q2 *randn(1)];
var_X=zeros(2,2);
W_1 = randn(1,length(Y));
```

```
W_2 = randn(1,length(Y));
X_n = zeros(2,length(Y));
MSE = zeros(2,length(Y));

for i = 1:length(Y)

    %step 1-- generate sigma points
    L= cholcov(var_X); %facotrization of covariance
    if size (L,1) <2
        L=sqrt(abs(var_X));
    end

    Particle_X_full = [mu_X, repmat(mu_X,1,N) + 4*L*(rand(2,N)-0.5)];

    % step 2 -- prediction
    x_hat = fx(k,theta,mu,sigma,a,rho,W_1(i),W_2(i),Particle_X_full);
    x_hat(:,1) = abs(x_hat(:,1));

    %step 3 -- predicted mean
    mu_pred = sum(weights.*x_hat,2);

    %step 4 -- predicted covariance
    x_hat_rem = (x_hat - repmat(mu_pred,1,N+1));
    cov_pred = zeros(2,2);
    for j=1:N+1
        cov_pred= cov_pred + weights(j).*x_hat_rem(:,j)*x_hat_rem(:,j)';
    end

    % step 5 -- get prediction points
    points_Y = x_hat + diag([Q1 Q2])*randn(2,N+1);

    %step 6 -- observation mean
    est_Y(i,:) = sum(weights.*points_Y,2);

    %step 7 -- observation covariance
    points_y_rem =points_Y-[est_Y(i,1)*ones(1,N+1);est_Y(i,2)*ones(1,N+1)];
    cov_neu =  zeros(2,2);
    for j=1:N+1
        cov_neu= cov_neu + weights(j).*points_y_rem(:,j)*points_y_rem(:,j)';
    end

    %step 8 -- update
    neu = Y(i,:) - est_Y(i,:);
    W_n = cov_pred/cov_neu.';
    mu_X= mu_pred + W_n * neu.';
    var_X = cov_pred - W_n*cov_neu*W_n.';

    X_n(:,i)= mu_X;

    %extra step: calculating MSE
    sum_square = 0;
    for j = 1:N
        sum_square = sum_square + abs(Particle_X_full(:,j).^2 - mean(Particle
    end
    MSE(:,i) = sqrt(1/(N*(N-1)) * sum_square);
end

%% Plots
figure(1);
plot(X_n(1,:),'-*','linewidth',2);
hold on;
grid on;
plot(Y(1:64,1),'-o', 'linewidth',2);
hold off;
xlabel('Year');
legend('Predicted yield', 'Real yield');
title('Dividend Yield for Unscented Kalman Filter');
%saveas(gcf,'UYield','epsc');

figure(2);
plot(X_n(2,:), '-*','linewidth',2);
hold on;
grid on;
plot(Y(1:64,2),'-o', 'linewidth',2);
hold off ;
xlabel('Year');
legend('Predicted return','Real return');
title('Real Return for Unscented Kalman Filter');
%saveas(gcf,'Ureturn','epsc');

figure(3)
plot(MSE(1,:), 'linewidth', 2);
grid on;
hold on;
plot(MSE(2,:), 'linewidth', 2);
xlabel('Year');
legend('MSE for yield', 'MSE for return');
title('MSE for S&P model');
hold off;
%saveas(gcf,'U_MSE1','epsc');
```

Appendix D

Matlab Code for the Tumor Growth Model with Unscented Kalman Filter

```
%Unscented Kalman filter for tumoral growth model
close all; clear; clc;
rng(50, 'twister')

%% Simulation of samples
% Parameters
sigma = 0.01; R = 0.1;
alpha1 = 1; alpha2 = 0.2; alpha3 = 0.2;
T_N = 8; delta_t = 0.04; n= T_N/delta_t;

X = zeros(n+1, 2);
X(1,:) = [0.8 0.3];
Y = zeros(n+1, 2);
Y(1,:) = X(1,:);

for i = 2:n+1
    F_x = F(alpha1, alpha2, alpha3, X(i-1,1), X(i-1,2));
   X(i,:) = X(i-1,:) + delta_t * F_x' + sigma *sqrt(delta_t) * randn(1,2);
   Y(i,:) = X(i,:) + R * sqrt(delta_t) * randn(1,2);
end
Y = Y(1:5:n+1,:);


%% Unscented Kalman Filter
%Set weights
omega = 1/3; N = 1500; w = (1-omega)/N;
weights = [omega, ones(1,N)*w];
```

```
%Initial mean and covariance
mu_X = Y(1,:)' - R * sqrt(0.2)*randn(2,1);
var_X=rand(2,2);
X_n = zeros(2,length(Y));

count=0;
for i = 1:length(Y)

    %step 1-- generate sigma points
    var_X = (var_X+var_X.')/2;
    L= cholcov(var_X); %facotrization of covariance
    if size (L,1) <2
        L=sqrt(abs(var_X));
        count =count+1;
        disp('i= ');
        disp (i);
    end

    Particle_X_full = [mu_X, repmat(mu_X,1,N) + 4*L*(rand(2,N)-0.5)];

    % step 2 -- prediction
    x_hat = Particle_X_full + F(alpha1, alpha2, alpha3, Particle_X_full(1,:),
    x_hat = abs(x_hat);

    %step 3 -- predicted mean
    mu_pred = sum(weights.*x_hat,2);

    %step 4 -- predicted covariance
    x_hat_rem = (x_hat - repmat(mu_pred,1,N+1));
    cov_pred = zeros(2);
    for j=1:N+1
        cov_pred= cov_pred + weights(j).*x_hat_rem(:,j)*x_hat_rem(:,j)';
    end

    % step 5 -- get prediction points
    points_Y = x_hat + R*sqrt(0.2) * randn(2,N+1);

    %step 6 -- observation mean
    est_Y(i,:) = sum(weights.*points_Y,2);

    %step 7 -- observation covariance
    points_y_rem =points_Y-[est_Y(i,1)*ones(1,N+1);est_Y(i,2)*ones(1,N+1)];
    cov_neu =  zeros(2,2);
    for j=1:N+1
```

```
            cov_neu= cov_neu + weights(j).*points_y_rem(:,j)*points_y_rem(:,j)';
        end

        %step 8 -- update
        neu = Y(i,:) - est_Y(i,:);
        W_n = cov_pred/cov_neu.';
        mu_X= mu_pred + W_n * neu.';
        var_X = cov_pred - W_n*cov_neu*W_n.';

        X_n(:,i)= mu_X;

        %extra step: calculating MSE
        sum_square = 0;
        for j = 1:N
            sum_square = sum_square + abs(Particle_X_full(:,j).^2 - mean(Particle
        end
        MSE(:,i) = sqrt(1/(N*(N-1)) * sum_square);
    end

    %% Plots
    x = [0:0.2:8]';
    figure(1);
    plot(x, X_n(1,:),'-*','linewidth',2);
    hold on;
    grid on;
    plot(x, Y(1:41,1),'-o', 'linewidth',2);
    hold off;
    xlabel('Time');
    legend('Predicted X1', 'Real X1');
    title('X1 for Unscented Kalman Filter');
    %saveas(gcf,'UX1','epsc');

    figure(2);
    plot(x, X_n(2,:), '-*','linewidth',2);
    hold on;
    grid on;
    plot(x, Y(1:41,2),'-o', 'linewidth',2);
    hold off ;
    xlabel('Time');
    legend('Predicted X2','Real X2', 'Location', 'best');
    title('X2 for Unscented Kalman Filter');
    %saveas(gcf,'UX2','epsc');

    figure(3)
    plot(x, MSE(1,:), 'linewidth', 2);
```

```
grid on;
hold on;
plot(x, MSE(2,:), 'linewidth', 2);
xlabel('Time');
legend('MSE for X1', 'MSE for X2');
title('MSE for Tumoral Growth Model');
hold off;
%saveas(gcf,'U_MSE2','epsc');


function [x_hat] = F(alpha1, alpha2, alpha3, x1, x2)
x_hat(1,:) = alpha1 .* x1 .* log(x2./x1);
x_hat(2,:) = alpha2 .* x1 - alpha3 .* x2 .* x1.^(2/3);

end
```

Appendix E

Matlab Code for the S&P Model with Particle Flow Filter

```
%Particle Flow Filter
close all; clear; clc;
rng(50, 'twister')

%% Simulation of samples
%Parameters
sigma = 0.01; R = 0.1;
alpha1 = 1; alpha2 = 0.2; alpha3 = 0.2;
N = 8; delta_t = 0.04; n= N/delta_t;

X = zeros(n+1, 2);
X(1,:) = [0.8 0.3];
Y = zeros(n+1, 2);
Y(1,:) = X(1,:);

for i = 2:n+1
    [f1, f2] = F(alpha1, alpha2, alpha3, X(i-1,1), X(i-1,2));
    X(i,:) = X(i-1,:) + delta_t * [f1 f2] + sigma *sqrt(delta_t) * randn(1,2)
    Y(i,:) = X(i,:) + R * sqrt(delta_t) * randn(1,2);
end
Y = Y(1:5:n+1,:);

 %% Particle Flow Filter
recording_X = zeros(length(Y), 2);
recording_X(1,:) = Y(1,:) - R*sqrt(0.2)*randn(1,2);
updated_X = zeros(2,n+1);
MSE = zeros(length(Y)-1,2);

 N_particle = 1500;
```

```
delta_lambda = 0.04;
syms lambda

for i = 2:length(Y)

    % Step 1: Calculate covariance matrix for prior density
    % Generate particles
    Particle_X = repmat(Y(i,:), N_particle,1) - R * sqrt(0.2) *randn(N_partic

    % Generate particle probability density functions
    u = Particle_X(:,1);
    v = Particle_X(:,2);
    u = ksdensity(u,u,'function','cdf');
    v = ksdensity(v,v,'function','cdf');
    [Rho,nu] = copulafit('t',[u v],'Method','ApproximateML');

    % Step 2: Set mean and covariance for likelihood function h(x)
    % Set mean
    m = mean(Particle_X)';

    % Set covariance
    Sigma = Rho;

    % Step 3: Generate covariance matrix of the diffusion
    P = [sigma 0 ; 0 sigma];
    R_matrix = [R 0; 0 R];
    Q(lambda)= (P-lambda*P*(R_matrix +lambda*P)^(-1)*P)*R_matrix^(-1)*(P-lamb
    new_Q(lambda)= 1/2*(Q(lambda)+Q(lambda)');

    % Step 4: Implicit iteration
    updated_X(:,1) = recording_X(i-1,:);
    for j = 1:n
       updated_X(:,j+1) = inv(eye(2) + delta_lambda * inv(inv(Rho) + (j/n)*in
    end

    recording_X(i,:) = updated_X(:,n+1);

    %extra step: calculating MSE
    sum_square = 0;
    for j = 1:N_particle
        sum_square = sum_square + abs(Particle_X(j,:).^2 - m'.^2);
    end
    MSE(i-1,:) = sqrt(1/(N_particle*(N_particle-1)) * sum_square);
end
```

```
%% Plots
x = [0.2:0.2:8]';
figure(1);
plot(x, recording_X(2:41,1),'-*','linewidth',2);
xlabel('Time');
hold on;
grid on;
plot(x, Y(2:41,1),'-o', 'linewidth',2);
hold off;
legend('Predicted X1', 'Real X1', 'Location', 'Best');
title('X1 for Particle Flow Filter');
%saveas(gcf,'PX1','epsc');

figure(2);
plot(x, recording_X(2:41,2), '-*','linewidth',2);
xlabel('Time');
hold on;
grid on;
plot(x, Y(2:41,2),'-o', 'linewidth',2);
hold off ;
legend('Predicted X2','Real X2', 'Location', 'Best');
title('X2 for Particle Flow Filter');
%saveas(gcf,'PX2','epsc');

figure(3)
plot(MSE(:,1), 'linewidth', 2);
grid on;
hold on;
plot(MSE(:,2), 'linewidth', 2);
xlabel('Time');
legend('MSE for X1', 'MSE for X2');
title('MSE for tumoral growth model');
hold off;
%saveas(gcf,'P_MSE2','epsc');

function [f1,f2] = F(alpha1, alpha2, alpha3, x1, x2)
f1 = alpha1 * x1 * log(x2/x1);
f2 = alpha2 * x1 - alpha3 * x2 * x1^(2/3);

end
```

Appendix F

Matlab Code for the Tumor Growth Model with Particle flow Filter

```
%Particle Flow Filter
close all; clear; clc;
rng(50, 'twister')

%% Simulation of samples
%Parameters
sigma = 0.01; R = 0.1;
alpha1 = 1; alpha2 = 0.2; alpha3 = 0.2;
N = 8; delta_t = 0.04; n= N/delta_t;

X = zeros(n+1, 2);
X(1,:) = [0.8 0.3];
Y = zeros(n+1, 2);
Y(1,:) = X(1,:);

for i = 2:n+1
    [f1, f2] = F(alpha1, alpha2, alpha3, X(i-1,1), X(i-1,2));
    X(i,:) = X(i-1,:) + delta_t * [f1 f2] + sigma *sqrt(delta_t) * randn(1,2)
    Y(i,:) = X(i,:) + R * sqrt(delta_t) * randn(1,2);
end
Y = Y(1:5:n+1,:);

 %% Particle Flow Filter
recording_X = zeros(length(Y), 2);
recording_X(1,:) = Y(1,:) - R*sqrt(0.2)*randn(1,2);
updated_X = zeros(2,n+1);
MSE = zeros(length(Y)-1,2);

 N_particle = 1500;
```

```
delta_lambda = 0.04;
syms lambda

for i = 2:length(Y)

    % Step 1: Calculate covariance matrix for prior density
    % Generate particles
    Particle_X = repmat(Y(i,:), N_particle,1) - R * sqrt(0.2) *randn(N_partic

    % Generate particle probability density functions
    u = Particle_X(:,1);
    v = Particle_X(:,2);
    u = ksdensity(u,u,'function','cdf');
    v = ksdensity(v,v,'function','cdf');
    [Rho,nu] = copulafit('t',[u v],'Method','ApproximateML');

    % Step 2: Set mean and covariance for likelihood function h(x)
    % Set mean
    m = mean(Particle_X)';

    % Set covariance
    Sigma = Rho;

    % Step 3: Generate covariance matrix of the diffusion
    P = [sigma 0 ; 0 sigma];
    R_matrix = [R 0; 0 R];
    Q(lambda)= (P-lambda*P*(R_matrix +lambda*P)^(-1)*P)*R_matrix^(-1)*(P-lamb
    new_Q(lambda)= 1/2*(Q(lambda)+Q(lambda)');

    % Step 4: Implicit iteration
    updated_X(:,1) = recording_X(i-1,:);
    for j = 1:n
       updated_X(:,j+1) = inv(eye(2) + delta_lambda * inv(inv(Rho) + (j/n)*in
    end

    recording_X(i,:) = updated_X(:,n+1);

    %extra step: calculating MSE
    sum_square = 0;
    for j = 1:N_particle
        sum_square = sum_square + abs(Particle_X(j,:).^2 - m'.^2);
    end
    MSE(i-1,:) = sqrt(1/(N_particle*(N_particle-1)) * sum_square);
end
```

```
%% Plots
x = [0.2:0.2:8]';
figure(1);
plot(x, recording_X(2:41,1),'-*','linewidth',2);
xlabel('Time');
hold on;
grid on;
plot(x, Y(2:41,1),'-o', 'linewidth',2);
hold off;
legend('Predicted X1', 'Real X1', 'Location', 'Best');
title('X1 for Particle Flow Filter');
%saveas(gcf,'PX1','epsc');

figure(2);
plot(x, recording_X(2:41,2), '-*','linewidth',2);
xlabel('Time');
hold on;
grid on;
plot(x, Y(2:41,2),'-o', 'linewidth',2);
hold off ;
legend('Predicted X2','Real X2', 'Location', 'Best');
title('X2 for Particle Flow Filter');
%saveas(gcf,'PX2','epsc');

figure(3)
plot(MSE(:,1), 'linewidth', 2);
grid on;
hold on;
plot(MSE(:,2), 'linewidth', 2);
xlabel('Time');
legend('MSE for X1', 'MSE for X2');
title('MSE for tumoral growth model');
hold off;
%saveas(gcf,'P_MSE2','epsc');

function [f1,f2] = F(alpha1, alpha2, alpha3, x1, x2)
f1 = alpha1 * x1 * log(x2/x1);
f2 = alpha2 * x1 - alpha3 * x2 * x1^(2/3);

end
```