Master's Thesis 2014

Candidate: Chao Xi

Title: Evaluation and comparison of Kalman Filter algorithms

Telemark University College

Faculty of Technology
Kjølnes
3914 Porsgrunn
Norway

Telemark University College
Faculty of Technology
M.Sc. Programme

MASTER'S THESIS, COURSE CODE FMH606

Student:	**Chao Xi**
Thesis title:	**Evaluation and comparison of Kalman filter algorithms**
Signature:	. . . . .
Number of pages:	71
Keywords:	**Nonlinear Kalman filter, Anaerobic digestion, Particle filter**
	Unscented Kalman filter, Extended Kalman filter

Supervisor:	**Finn Haugen**	sign.:	. . . . .
2nd Supervisor:		sign.:	. . . . .
Censor:		sign.:	. . . . .
External partner:		sign.:	. . . . .
Availability:	**Open**		

Archive approval (supervisor signature)**:** sign.: . . . . . **Date :** . . . . . .

Abstract:

In practical applications, biogas flow sensors are used to measure the output methane gas from the reactor. But none of the states are measured. State estimation methods can be used to solve this problem. In this thesis the problem of optimal state estimation regarding to an anaerobic digestion (AD) reactor is considered. For the nonlinear system, different approaches: extended Kalman filter (EKF), unscented Kalman filter (UKF) and particle filter (PF), are used to estimate the unmeasured states.

The main aim is to evaluate and compare the three alternative Kalman filter algorithms with a proper state augmentation, both in terms of estimation accuracy and computational efforts, by applying them to a real and a simulated AD reactor.

Both the simulation and test with real data show that UKF has a similar performance with EKF, but more susceptible to the initial condition. PF has the best state estimate performance with the highest computational requirement, about 30 times more CPU time than the EKF. Overall we conclude that the EKF, with Jacobian calculations every time step, is the best choice for the unmeasured state estimation in this case.

Contents

Preface

This thesis is submitted in partial fulfillment of the requirements for a Master's Degree in Systems and Control Engineering. It contains work done from February to June 2014. The master thesis work is done based on Foss Biolab which is a joint research laboratory between Telemark University College and Foss farm in Skien, Norway. The implementation and results were carried out using MATLAB.

It is expected that the reader have some knowledge on modeling and simulation, and preferably basic knowledge on probability theory. Understanding of MATLAB code would be beneficial, but it's not necessary.

At last, I would like to thank my parents for their constant loving care and financial support. Special thanks to my supervisor, Finn Haugen, for his elaborate guidance, valuable comments and sincere encouragement. I would like to extend my gratitude to Evelyn and Alf Oden Hansen, for their precious friendship. Thanks are also due to a larger number of students who offered advices and suggestions. To you all, I hereby present a heart of flowers and sincere blessings!

Porsgrunn, June 2nd, 2014

Chao Xi

Nomenclature

List of symbols

Symbol	Meaning/Explanation
S_{vfa}	Concentration of VFA in reactor [g VFA/L].
S_{bvs}	Concentration of BVS in reactor [g BVS/L].
X_{acid}	Concentration of acidogens[g acidogens/L].
X_{meth}	Concentration of methanogens[g methanogens/L].
T_{reac}	Reactor temperature [°C].
F_{feed}	Influent or feed flow or load rate, assumed equal to effluent flow (constant volume) [L/d].
F_{meth}	Methane gas flow [L CH4/d].
A_c	State matrix of state-space model in continuous time domain.
B_c	Input-to-state matrix of state-space model in continuous time domain.
C_c	State-to-output matrix of state-space model in continuous time domain.
D_c	Direct feed-through matrix of state-space model in continuous time domain.
I	Identity matrix.
$x(t_k)$	Time continuous representation of variable x at time t_k.
$\dot{x}(t_k)$	Time derivative of $x(t_k)$.
$E(x)$	Expected value of x.
$cov(x)$	Covariance of x.
$Q(k)$	Covariance squared matrix of process noise, in discrete time domain.
$R(k)$	Observation (sensor) noise covariance, in discrete time domain.
$w(k)$	Process noise or system noise; vector in discrete time domain.
$v(k)$	Observation or sensor noise; vector in discrete time domain.
i, j, k	Subscripts of a sum or subscripts of a matrix elements, or index denoting iteration.

List of abbreviations

Abbreviation	Meaning/Explanation
AD	Anaerobic digestion
BVS	Biodegradable volatile solid
PWM	Pulse width modulation
VFA	Volatile fatty acids
EKF	Extended Kalman filter
UKF	Unscented Kalman filter
PF	Particle filter
RMSE	Root mean square error
pdf	Probability distribution function
PID	Proportional Integral Derivative
P&ID	Piping &Instrumentation diagram

Overview of figures

Overview of tables

1 Introduction

1.1Backgrond and previous work

Nowadays, state estimation of dynamic systems plays an important role in operating the industry processes safely and economically, since it is of central importance in monitoring and control of system. Furthermore, a variety of system identification problems can be addressed by state estimation. As it was indicated by Simon (2006a), *the states of a system are those variables that provide a complete representation of the internal conditions or status of the system at a given instant of time.* The problem of determining the state of a system from noisy measurements is called estimation or filtering(Jazwinski, 1970). State estimation algorithm can estimate and predict desired state variables of a dynamic system from available noisy measurements, and estimation of the state variables is one of the fundamental and significant problems in control and signal processing areas. State estimation can be utilized for supervision in a physical process where states cannot be directly measured or disturbances that can be of vital importance. Likewise, it is critical to make a more reliable or advanced controller for the processes, because it can help to overcome measurement uncertainties due to sensor failure or noise.

One of the most popular state estimation tools is Kalman filter, which has been applied in various engineering and scientific areas such as communications, machine learning, neuroscience, economics, finance, political science, and many others. The Kalman filter(Kalman, 1960) is a popular and effective tool which can estimate the variables of a great variety of processes. It provides a way to estimate the state vector using an optimal observer gain to minimize the expected mean square error of the estimate.

The standard Kalman filter can be easily applied to linear system. Nonlinear systems, however, require approximations in order to handle the nonlinearity in the state dynamic equations and/or the output equations of the system. There are many variations on the Kalman filter for nonlinear systems, three of the most common ones are the extended Kalman filter (EKF), the unscented Kalman filter (UKF) and the particle filter (PF). These three filters use different approaches to handle the nonlinearity. The EKF relies on linearization of both the transition and measurement equations of the nonlinear system which involving calculation of Jacobian matrices. Unfortunately, the EKF has two important potential drawbacks. First, the derivation of the Jacobian matrices can be complex causing implementation difficulties. Second, the linearization can lead to filter instability if the time step intervals are not sufficiently small (Julier et al., 1995, Simon, 2006c). To handle with these limitations, Julier et al. (1995) suggested an alternative as UKF. The UKF uses a statistical approach called the Unscented Transformation (UT) (Julier and Uhlmann, 1997) for propagation of means and covariances. The PF, introduced by Gordon et al.

(1993b), estimate the posterior density of the state-space by directly implementing the Bayesian recursion equations. On comparing with the last two approximation methods, the particle method do not rely on a linear approximation scheme or any crude functional approximation and give a better result with sufficient samples. The price is that more computational effort needs to be paid for this flexibility.

Recently, Kalman filters have been applied in different kinds of anaerobic digestion (AD) reactors. In a study of catalytic reactor, Kupilik and Vincent (2011) have successfully used the EKF method in estimating an unknown inlet composition from only measurement of the reactor temperature. In a study based on real data, Haugen et al. (2014) estimate four states and an unknown influent concentration for a real AD pilot reactor fed with dairy manure. Modified Hill model and UKF method are deployed. The state estimator uses only methane gas flow measurement to update its states.

From different comparison studies, UKF is a superior alternative to EKF for a variety of estimation and control problems. Based on both theoretical and empirical analyses of a particular application, the EKF was found to be approximately 10 times more computationally efficient than the UKF(Gross et al., 2012). A test conducted by Chai et al. (2007) in wastewater treatment area found similar performance between them. According to Myers et al. (2012), it was found that the EKF and PF methods presented similar results when using the one-way ultrasonic pulse time of flight measurement model. However, an empirical study(LaViola Jr, 2003) shows that EKF is a better choice for estimating quaternion motion in virtual reality. By comparing different filtering approaches, Simon (2008) concluded that EKF is the best choice for aircraft engine health estimation. In an explicit study on comparison of EKF, UKF and PF applying to a tracking problem, has (Ristic et al., 2004)found that the EKF is fast but unreliable, UKF shows similar error with PF while needs more CPU time. Regarding the state estimation of AD reactor, a pioneering work by (Haugen et al., 2012a) illustrated that UKF has the similar performance as EKF, but more adaptable when model changes.

Many estimation algorithms employ a dynamic model of the system under consideration. It's effective to use augmented models to extend such algorithms to fit practical problems. Augmentative states describe appending additional dynamics to an existing model of the system. These augmented dynamics may represent system disturbances or model parameters. The advantage of augmented models is that they don't conceptually change the underlying algorithm, and still retain most or all of its properties(Maeder, 2010). In the domain of state estimation or system identification, augmented models have shown to be useful. According to Ray (1997) and Gustafsson (1997), tire/road friction coefficients for vehicles were inferred from the behavior of the vehicles. However, in simulation study of a simple nonlinear case, Fuming et al. (2009) found that UKF is scenario-dependent, that means, non-augmented UKF has better performance than the augmented UKF in the presence of a bigger process noise.

1.2 Statement of the Problem

In the AD reactor there are several parameters and states which are not measured, but are crucial for the performance. Such kind of parameter like the concentration of volatile fatty acids (S_{vfa}), should cause process failure in case of abnormal situation. Because high concentration of volatile fatty acids (VFA) is inhibitory to methane generating microbes. In addition, advanced controllers may be developed based on the estimation of system states. So it is necessary to estimate the internal condition of the system using a state estimation algorithm. It is interesting to identify which method is most appropriate for the AD reactor.

The purpose of this thesis is to evaluate and compare different Kalman filter algorithms for state estimation of the AD reactor. Therefore, we will focus on study the differences of the three filters when applying to the biogas reactor.

- Comparison of Estimation Accuracy of the filtering techniques.
- Comparison of Computational efficiency.
- Comparison of augmented and non-augmented Kalman filters

1.3 Thesis outline

In Chapter 1, an introduction of the state estimation application is given, including literature review and a short statement of the purpose of this thesis. A popular mathematical model is briefly explained in Chapter 2, where the deduction of the model to the Kalman filter algorithms is also proposed. In Chapter 3, three estimation methods are described and analysis of difference among the estimation algorithms is focused. In particular, the Bayesian approach for nonlinear dynamic systems is discussed. Chapter 4 explains the implementation of both model and Kalman filters in MATLAB, and simulation results. In chapter 5, Real time series data from Foss Biolab is applied to the filters, all the states are estimated by the different filters. Chapter 6 concludes the work and suggests some future works.

2 Mathematical modeling

Several models have been proposed for biogas reactor. Some popular models such as Hill's model(Hill, 1983), which states a kinetic model that describes the fermentation of animal wastes with upgrade accuracy and predictive ability; Husain's model(Husain, 1998) developed a steady state solution to Hill's simplified dynamic model. In the next context, a simply modified Hill's model proposed by (Haugen et al., 2013)will be used. The biogas plant will be described in Section 2.1 and followed with a model presentation in Section 2.2. Section 2.3 shows a state space model and Section 2.4 carries out a model linearization.

2.1 System description

Foss Biolab is a pilot biological plant for nutrient and energy recovery, situated at Foss farm, Skien, Norway. The plant contains AD reactor. The feedstock to the reactor is dairy manure diluted with 25% water and filtered with a sieve. Output from the AD reactor is biogas consisting of approximately 70% methane and liquid fertilizer, which is fed to a nitrification reactor at the plant. The AD reactor temperature is kept fixed at its set point with an automatic temperature control system. The plant is monitored and controlled by a PC-based automation system that implemented in LabVIEW. Figure 2.1 shows a Piping &Instrumentation Diagram (P&ID) of the biological plant.

The plant is mainly consists of four different parts as shown in Figure 2.1. According to Haugen et al. (2012b), "the first is a reservoir for raw dairy manure containing approximately 25% added water. The second is a separator for separating the manure into two fractions of similar total solid mass: >70% of the volume is the wetter fraction and <30% is the dryer fraction. The third part is a 220L high rate AD reactor fed with filtered cow manure as substrate for production of energy-rich biogas that contains mainly methane. The last part is a 200L nitrification reactor fed with AD reactor effluent to produce high quality liquid fertilizer and pellets fertilizer from formed foam."

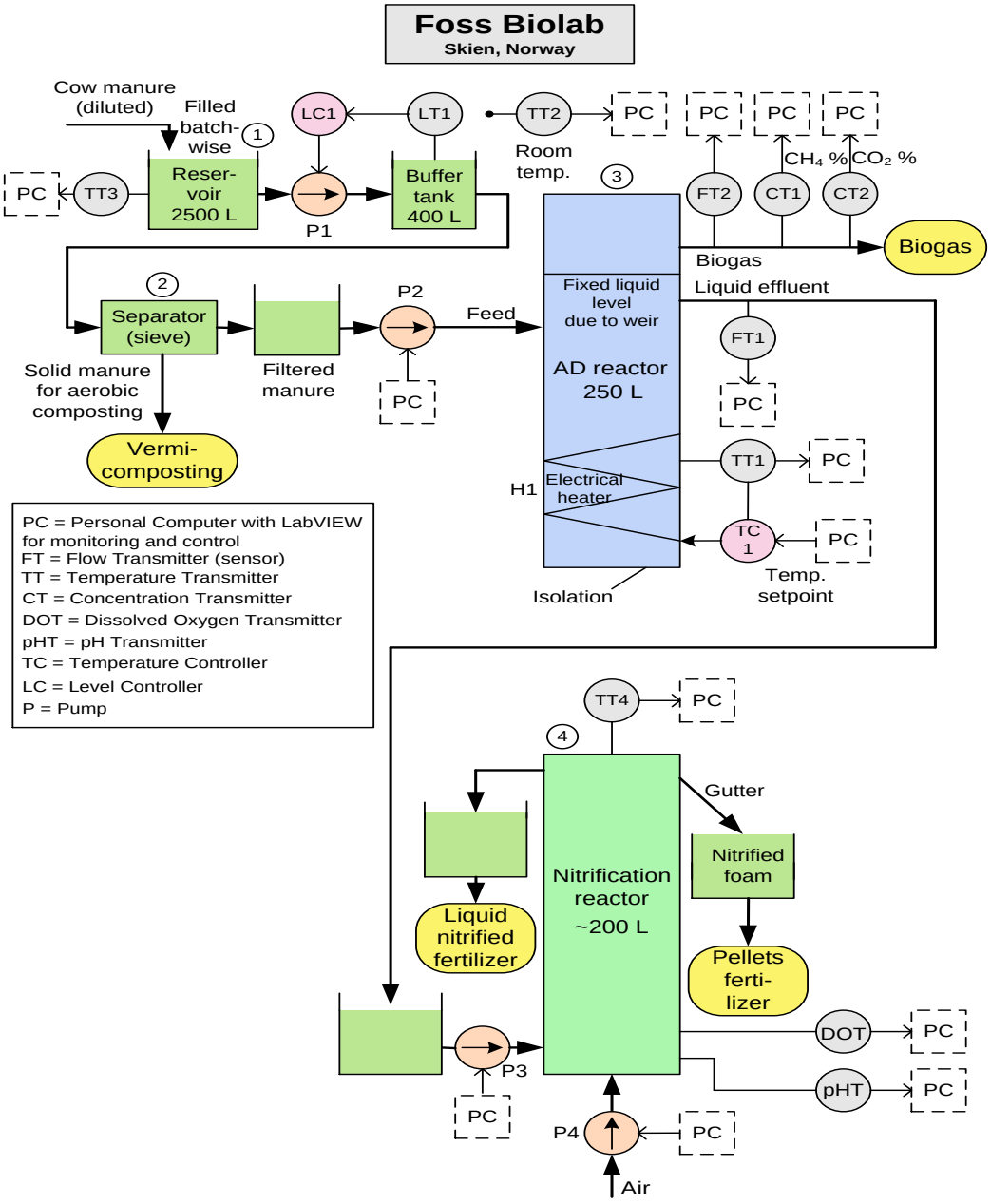

Figure 2.1: Piping & Instrumentation Diagram (P&ID) of the biological process line of the pilot plant at Foss Biolab(Haugen et al., 2013).

2.2 Modified Hill's model

This thesis work will directly use the modified Hill's model, which has been deliberately depicted and approved sufficiently accurate as a basis for both state estimation and advanced control strategies design(Haugen et al., 2013). The dynamic model consists of four states.

- S_{bvs} - Concentration of biodegradable volatile solids (BVS)

- S_{vfa} - Concentration of total volatile fatty acid (VFA)
- X_{acid} - Concentration of acid-forming micro-organisms
- X_{meth} - Concentration of methane-forming micro-organisms

Model equations in the modified Hill's model can be expressed as follows. Defining that portion of the raw waste which can serve as biodegradable material (substrate):

$$S_{bvs_{in}} = B_0 S_{vs_{in}} \tag{2.1}$$

Defining that portion of the biodegradable material which is initially in acid form:

$$S_{vfa_{in}} = A_f S_{bvs_{in}} \tag{2.2}$$

Mass balance of BVS:

$$\dot{S}_{bvs} = (S_{bvs_{in}} - S_{bvs})\frac{F_{feed}}{V} - \mu k_1 X_{acid} \equiv f_1\,(\cdot) \tag{2.3}$$

Mass balance of total VFA:

$$\dot{S}_{vfa} = (S_{vfa_{in}} - S_{vfa})\frac{F_{feed}}{V} + \mu k_2 X_{acid} - \mu_c k_3 X_{meth} \equiv f_2\,(\cdot) \tag{2.4}$$

Mass balance of acidogens:

$$\dot{X}_{acid} = \left(\mu - K_d - \frac{F_{feed}}{bV}\right) X_{acid} \equiv f_3\,(\cdot) \tag{2.5}$$

Mass balance of methanogens:

$$\dot{X}_{meth} = \left(\mu_c - K_{dc} - \frac{F_{feed}}{bV}\right) X_{meth} \equiv f_4\,(\cdot) \tag{2.6}$$

Methane gas flow rate (gas production):

$$F_{meth} = V\mu_c k_5 X_{meth} \equiv \mathrm{h}\,(\cdot) \tag{2.7}$$

Where, the reaction rates, with Monod kinetics, are as follows:

$$\mu = \mu_m \frac{S_{bvs}}{k_s + S_{bvs}} \tag{2.8}$$

$$\mu_c = \mu_{mc} \frac{S_{vfa}}{k_s + S_{vfa}} \tag{2.9}$$

Where, the maximum reaction rates are functions of the reactor temperature as follows:

$$\mu_m(T_{reac}) = \mu_{mc}(T_{reac}) = 0.013T_{reac} - 0.129 \tag{2.10}$$

$$(20^oC < T_{reac} < 60^oC)$$

Known parameters are provided as illustrated in Table 2.1.

Table 2.1: Parameters in Modified Hill's model adapted to AD reactor at Foss Biolab(Haugen et al., 2014).

Parameter	Value	Unit	Comment
A_f	0.69	(g VFA/L)/ (g BVS/L)	Acidity constant
b	2.90	d/d	Retention time ratio
B_0	0.25	(g BVS/L)/ (g VS/L)	Biodegradability constant
k_1	3.89	g BVS/(g acidogens/L)	Yield constant
k_2	1.76	g VFA/(g acidogens/L)	Yield constant
k_3	31.7	g VFA/(g methanogens/L)	Yield constant
k_5	26.3	L/g methanogens	Yield constant
K_d	0.02	d^{-1}	Specific death rate of acidogens
K_{dc}	0.02	d^{-1}	Specific death rate of methanogens
K_s	15.5	g BVS/L	Monod half-velocity constant for acidogens
K_{sc}	3	g VFA/L	Monod half-velocity constant for Methanogens
V	250	L	Effective reactor volume

2.3 State space model

A state space model is a mathematical model of a process, where the process state x is represented by a numerical vector. State space model consists of a process model which describes how the state propagates in time based on external influences such as input and noise, and a measurement model which describes how the measurement y is obtained from the process.

The model provided for the AD reactor is a continuous time model and for implementation of state estimation algorithm the model has to be discretized. The continuous model was discretized using an Euler forward approximation. There are other types of discretization methods, however Euler forward was chosen as it is straightforward and simple to use.

Continuous-time model:

Letting $[S_{bvs} \quad S_{vfa} \quad X_{acid} \quad X_{meth}]^T = x$, $F_{feed} = u$, $F_{meth} = y$, the continuous-time model can be represented in a more condense form as:

$$\dot{x} = f(x, u) \tag{2.11}$$

$$y = h(x, u) \tag{2.12}$$

Where f is the system function composed of four differential equations as (2.3 – 2.6), h is the measurement or process output function as (2.7), $x \in \mathbb{R}^4$ is the state vector, u is the input. Applying the forward Euler approximation on the Continuous model

$$\dot{x} = \frac{x_{t+1} - x_t}{\Delta T} \Rightarrow x_{t+1} = x_t + \Delta T f(x, u) \quad (2.13)$$

Which leads to the discrete-time model as

$$x_{t+1} = x_t + T_s f(x_t, u_t) \quad (2.14)$$

$$y_t = h(x_t, u_t) \quad (2.15)$$

Where, ΔT is the integration interval. However the sampling time Ts correlated to the integration time as $T_s = m\Delta T$ typically with m=1. In numerical theory, as ΔT becomes infinitesimally small, the finite difference relation in (2.13) becomes a better approximation of the differential equation in (2.11). Here ΔT depends on the experimental data. In the real process, the produced methane gas is sampled every 15 minutes and filtered every 0.1 day (8640 seconds). For a better prediction and given that a slow and steady process, we choose 0.1 day (8640 seconds) as our sampling time.

2.4 Linearization and evaluation

The purpose for this section is to build a linearized model for the state estimation algorithms.

2.4.1 Steady-state operating point

The idea of linearization is to find a linear system whose states represent the deviations from a nominal trajectory of the nonlinear system(Simon, 2006c). To linearize the nonlinear system, we first need to find a nominal state, also known as steady-state operating point or equilibrium point. The desired behavior of a system is when it maintains a predefined output and state stability. This means that the system can be linearized around this equilibrium point (x_0, u_0). With equilibrium point means that the first derivative of the states is equal to zero,

$$\dot{x} = f(\mathrm{x}_0, \mathrm{u}_0) \equiv 0 \quad (2.16)$$

The equilibrium point includes state variables that do not change with time.by analysis of the AD reactor dynamics and stability, Haugen et al. (2013) have given such a point in a systematic way. Values of inputs and states in the steady-state operation point are showed in Table 2.2.

Table 2.2: Values of inputs and states in the steady-state operation point.

F_{feed} = 45L/d
T_{reac} = 35 ℃
S_{bvs} = 5.2155g/L
S_{vfa} = 1.0094g/L
X_{acid} = 1.3128g/L
X_{meth} = 0.3635g/L
S_{vsin} = 30.2g/L
F_{meth} = 196.1560 L CH4/d

2.4.2 Linearization

The result from the modeling of the reactor was nonlinear functions which depend on the state-space variable x and the input signal F_{feed}. By linearizing these nonlinear equations, as Eqs. (2.3-2.6), will result in the following description of the dynamics

$$\Delta\dot{x} = A_c\Delta x + B_c\Delta F_{feed} \tag{2.17}$$

$$\Delta F_{meth} = C_c\Delta x + D_c\Delta F_{feed} \tag{2.18}$$

Where *x* is the state vector

$$x = \left[S_{bvs}\, S_{vfa}\, X_{acid}\, X_{meth}\right]^T = [x_1\, x_2\, x_3\, x_4]^T \tag{2.19}$$

$$\Delta x = x - x_0 \tag{2.20}$$

$$\Delta F_{feed} = F_{feed} - F_{feed_0} \tag{2.21}$$

$$\Delta F_{meth} = F_{meth} - F_{meth_0} \tag{2.22}$$

The linearization of the dynamic model will result in A_c , B_c , C_c and D_c matrices, where the elements in these matrices are given by $a_{i,j}$, $b_{i,j}$, $c_{i,j}$, respectively,

$$a_{i,j} = \frac{\partial f_i(x_0, F_{feed_0})}{\partial x_j},\quad b_{i,j} = \frac{\partial f_i(x_0, F_{feed_0})}{\partial F_{feed_j}},\quad c_{i,j} = \frac{\partial h_i(x_0, F_{feed_0})}{\partial x_j},\ d_{i,j} = \frac{\partial h_i(x_0, F_{feed_0})}{\partial F_{feed_j}} \tag{2.23}$$

Where i denotes the row and j denotes the column, i, j = 1…4.

$$A_c = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{F_{feed}}{V} - \frac{\mu_m k_1 K_s X_{acid}}{(K_s + S_{bvs})^2} & 0 & -\frac{k_1 \mu_m S_{bvs}}{(K_s + S_{bvs})} & 0 \\ \frac{\mu_m k_2 K_s X_{acid}}{(K_s + S_{bvs})^2} & -\frac{F_{feed}}{V} - \frac{\mu_{mc} k_3 K_{sc} X_{meth}}{(K_{sc} + S_{vfa})^2} & \frac{k_2 \mu_m S_{bvs}}{(K_s + S_{bvs})} & -\frac{k_3 \mu_{mc} S_{vfa}}{(K_{sc} + S_{vfa})} \\ \frac{\mu_m K_s X_{acid}}{(K_s + S_{bvs})^2} & 0 & -\frac{F_{feed}}{bV} + \frac{\mu_m S_{bvs}}{(K_s + S_{bvs})} - K_d & 0 \\ 0 & \frac{\mu_{mc} K_{sc} X_{meth}}{(K_{sc} + S_{vfa})^2} & 0 & -\frac{F_{feed}}{bV} + \frac{\mu_{mc} S_{vfa}}{(K_s + S_{vfa})} - K_{dc} \end{bmatrix} \tag{2.24}$$

$$B_c = \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{bmatrix} = \begin{bmatrix} \frac{B_0 S_{vs_{in}} - S_{bvs}}{V} \\ \frac{A_f B_0 S_{vs_{in}} - S_{vfa}}{V} \\ -\frac{X_{acid}}{bV} \\ -\frac{X_{meth}}{bV} \end{bmatrix} \tag{2.25}$$

$$C_c = [c_{11} \quad c_{12} \quad c_{13} \quad c_{14}] = \begin{bmatrix} 0 & \frac{\mu_{mc} V k_5 K_{sc} X_{meth}}{(K_{sc} + S_{vfa})^2} & 0 & \frac{V k_5 \mu_{mc} S_{vfa}}{(K_{sc} + S_{vfa})} \end{bmatrix} \tag{2.26}$$

$$D_c = 0 \tag{2.27}$$

The linearized discrete-time model can be derived from Eq.(2.17) and (2.18)

$$\Delta \dot{x}(t_{k+1}) = A\Delta x + B\Delta F_{feed} \tag{2.28}$$

$$\Delta F_{meth}(t_k) = C\Delta x(t_k) + D\Delta F_{feed}(t_k) \tag{2.29}$$

Where A, B, C, and D can be calculated from A_c , B_c , C_c and D_c matrices by command 'c2d' in MATLAB. As discussed in Section 2.2, discretization time is selected as 0.1 day. Corresponding program can be seen as Appendix B and the results as below.

$$A = \begin{bmatrix} 0.9762 & 0 & -0.0315 & 0 \\ 0.0026 & 0.9154 & 0.0138 & -0.2490 \\ 0.0015 & 0 & 1.0000 & 0 \\ 0 & 0.0021 & 0 & 0.9997 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.0024 \\ 0.0026 \\ -0.0002 \\ -0.0000 \end{bmatrix}$$

$$C = [0 \quad 145.4053 \quad 0 \quad 539.6315]$$

$$D = 0$$

2.4.3 Linearized model evaluation

For this work we assumed a perfect model being used, so we are not discussing the dynamics of the Modified Hill's model. The topic here is to compare the response from both the nonlinear model and the discrete-time model given the identical condition. Figure 2.2 displays the simulation result when same step responses are given to both nonlinear and linear model. Corresponding code can be seen as Appendix C.

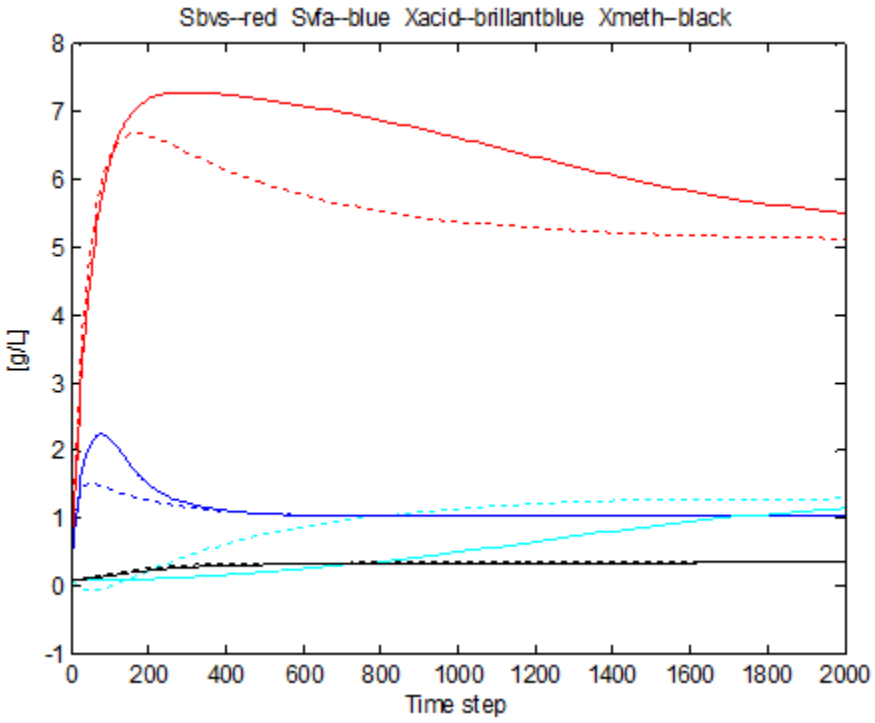

Figure 2.2: Simulation result when same step responses are given to both nonlinear and linear model. Initial value as [0; 0; 0; 0] for both nonlinear model and linearized model.

The simulation show that the discrete-time model gives a similar response as the nonlinear model does. Hence we can conclude that the discrete model can describe the most dominant dynamics of the AD reactor.

3 State estimation with Kalman filter

Measuring all of the states through sensors is usually not possible, because required sensors can be expensive, and some states cannot be measured. The commonly used method to get around this problem is to construct an estimator for the state based on the output. Specifically, the output measures some of the state variables, which are affected by the states that we do not measure. By examining how the measured states change with time, we can potentially determine the values of the unmeasured states as well. We will do this by constructing a state estimator as Kalman filter.

In this chapter, three nonlinear extensions of the Kalman filter will be discussed. In Section 3.1, we will explain briefly how a Kalman filter works. In Section 3.2 we will develop an augmented model for different nonlinear filters. Then Section 3.3 will put emphasis on the EKF algorithm, which is the most widely used estimation technique the past few decades. In Section 3.4, we will introduce the unscented transformation and after that the UKF, an extension of Kalman filter that reduces the linearization error of the EKF. And in the last section, we will discuss the PF method, which is a probability-based estimator.

3.1 Introduction to Kalman filter

The Kalman filter is an algorithm used to estimates the values of state variables of a dynamic system which is exited by stochastic disturbances and stochastic measurement noise. The Kalman filter will produce an optimal estimate in the sense that the mean value of the sum of the estimation error gets a minimal value. A Kalman filter usually works as shown in Figure 3.1.

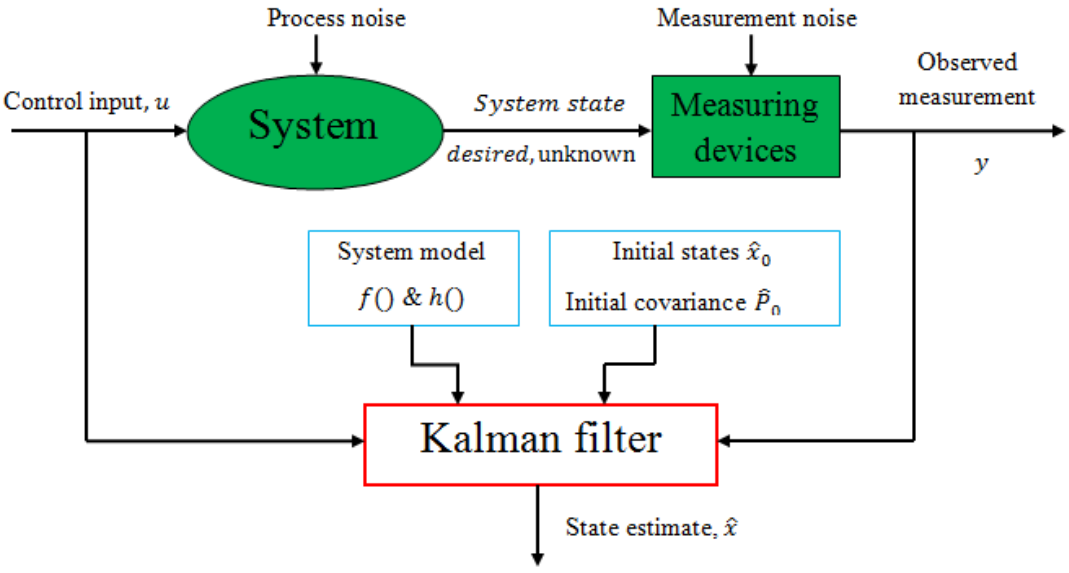

Figure 3.1: Block diagram of principle of Kalman filter.

The Kalman filter runs parallel with the physical process, which contains the system and measuring devices as shown in Figure 3.1. Thus the input-output data can be processed recursively as they become available. Consider the dynamic system described by linear stochastic difference equation f and measuring devices by h, the problem now is to estimate the (desired) unknown system state vector from noisy (observed) measurement y. The system model should also consists of simulated process noise and measurement noise, which are usually assumed to be mutually independent, white, Gaussian noises with known covariance matrices respectively. The initial state vector $\hat{x}_0$ is also assumed to be independent of the noises. The Kalman filter estimates the system states with two set of equations: time update equations (apriori estimates) and measurement update equations (aposteriori estimates). To obtain the apriori estimates for the next time step the time update equations project forward (in time) the current state and error covariance estimates. The measurement update equations get the feedback to obtain an improved aposteriori estimate incorporating a new measurement into the apriori estimate.

3.2 Augmented model for nonlinear filtering

Augmented models can be used for online estimation of system parameters. The parameters which define the dynamics of the system are assumed to be unknown or varying. Since the parameters usually enter the system dynamics nonlinearly, nonlinear estimation schemes such as EKF or UKF are applied. In this project, we will introduce an augmentative state variable - an assumed unknown influent concentration$S_{vs,in}$. We choose this parameter because it has a close relationship with the methane that was produced by the digester(Fischer et al., 1984).

The corresponding differential equation is

$$\dot{S}_{vsin} = 0 \tag{3.1}$$

And discrete form

$$S_{vsin}(k+1) = S_{vsin}(k) + w_{vsin}(k) \tag{3.2}$$

Where w_{vsin} is white process noise with assumed auto-covarianceQ_{vsin}.

Our system now evolves to a five states system. The state vector is

$$\begin{aligned} x = & \ [S_{bvs} \quad S_{vfa} \quad X_{acid} \quad X_{meth} \quad S_{vsin}]^T \\ = & \ [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5]^T \end{aligned} \tag{3.3}$$

Nothing changed when we try to implement a Kalman filter for the augmented model. The Kalman filter estimates both the original states and the augmentative states. Then the discrete nonlinear model for our state estimator can be developed from the continuous time model as Eqs.(2.14) and (2.15).

$$x(t_{k+1}) = x(t_k) + T_s f_c[x(t_k, u_k)] + w(t_k) \tag{3.4}$$

$$y(t_k) = h[x(t_k)] + v(t_k) \tag{3.5}$$

w is the random process noise with mean 0 and covariance Q

$$w(t_k) \sim [0, Q(t_k)]$$

v is the random process noise with mean 0 and covariance R

$$v(t_k) \sim [0, R(t_k)]$$

The linearized discrete-time model can be derived from Eqs.(2.28) and (2.29)

$$\Delta\dot{x}(t_{k+1}) = A\Delta x + B\Delta F_{feed} + \Delta w(t_k) \tag{3.6}$$

$$\Delta F_{meth}(t_k) = C\Delta x(t_k) + D\Delta F_{feed}(t_k) + \Delta v(t_k) \tag{3.7}$$

Where *A*, *B*, *C*, and *D* can be calculated by command 'c2d' in MATLAB.

3.3 Extended Kalman Filter

The system and measurement equations are (3.4) and (3.5).

The filter algorithm can be summarized as follows(Haugen et al., 2012b).

Initial step ($k = 0$):

Aposteriori state estimate:

$$x_{apost}(t_0) = E(x_0) \tag{3.8}$$

Covariance of estimation error:

$$P_{apost}(t_0) = E\left\{\begin{matrix}[x(t_0) - x_{apost}(t_0)] \\ [x(t_0) - x_{apost}(t_0)]^T\end{matrix}\right\} \tag{3.9}$$

For time steps $k = 1,2,3\ldots$

1. Partial derivative of system function (Jacobian matrix):

$$F(t_{k-1}) = \frac{\partial f}{\partial x}\Big|_{x_{apost}(t_{k-1}),u(t_{k-1})} = I + T_s \frac{\partial f_c}{\partial x}\Big|_{x_{apost}(t_{k-1}),u(t_{k-1})} \tag{3.10}$$

2. Time-updates:

(a) Apriori estimate (predicted estimate):

$$x_{apri}(t_k) = f\left[x_{apost}(t_{k-1}), u(t_{k-1})\right]$$
$$= x_{apost}(t_{k-1}) + T_s f_c\left[x_{apost}(t_{k-1}), u(t_{k-1})\right] \quad (3.11)$$

(b) Covariance of estimation-error:

$$P_{apri}(t_k) = F(t_{k-1})P_{apost}(t_{k-1})F(t_{k-1})^T + Q \quad (3.12)$$

3. Partial derivative of output function (Jacobian matrix):

$$H(t_k) = \frac{\partial h}{\partial x}\Big|_{x_{apri}(t_k), u(t_k)} \quad (3.13)$$

4. Measurement updates:

(a) Kalman filter gain:

$$K(t_k) = P_{apri}(t_k)H(t_k)\left[H(t_k)P_{apri}(t_k)H(t_k)^T + R\right]^{-1} \quad (3.14)$$

(b) Measurement estimate:

$$y_{apri}(t_k) = h(x_{apri}(t_k)) \quad (3.15)$$

(c) State estimate (Aposteriori estimate, or corrected estimate), which is used as the applied estimate, x_e:

$$x_e(t_k) = x_{apost}(t_k) = x_{apri}(t_k) + K(t_k)(y(t_k) - y_{apri}(t_k)) \quad (3.16)$$

(d) Covariance of estimation error:

$$P_{apost}(t_k) = [I - K(t_k)H(t_k)]P_{apri}(t_k) \quad (3.17)$$

Equations (3.8) - (3.17) are the complete algorithm of EKF. The algorithm is implemented in MATLAB and attached in Appendix D.

According to the algorithm, EKF takes the estimate $x_{apost}(t_k)$ from last loop to the initial step of the current loop. The estimation result is then used as a nominal state, where states of the linear system represent the deviations from a nominal trajectory of the nonlinear system(Simon, 2006b). Therefore, the accuracy of estimation in a loop is not only relied to the linearization in the loop, but also the estimated value from last loop. That means this algorithm can be divergent if the consecutive linearization is not good approximation of the nonlinear model. Normally, attention needs to be paid in these two points when using EKF:

- The linearization method based on Taylor expansion is susceptible to the nominal state, where EKF recursively take up with the current estimate. The Kalman gain $K(t_k)$ depends on the current state estimate. So if there a big difference between the current

estimate and real values, further linearization error and imprecise Kalman filter updates will be caused.

- Because of calculation of two Jacobian matrices, the continuity of the state transition model and observation model should be paid attention when using EKF.

The above points illustrate the basic premises of EKF: Small deviation of initial conditions and system of weak nonlinearity; enough smoothness and continuity of the model to make sure the existence of Jacobian matrix $F(t_k)$ and $H(t_k)$.

3.4 Unscented Kalman Filter

As discussed earlier, EKF always use first-order linearization to propagate the mean and covariance of the state, thus the posterior mean and covariance could be corrupted. Unlike EKF, the UKF overcome this problem by using a deterministic sampling approach(Wan and van der Merwe, 2002). The state distribution is represented using a minimal set of carefully chosen sample points, called sigma points. The points are then propagated through the non-linear transformation, obtaining a cloud of points in the transformed space, and their mean and covariance are computed. This method permits to avoid the linearization and takes the name of UT.

3.4.1 Unscented Transformation

The UT calculates the statistics of a random variable by a nonlinear transformation. And it is based on the principle, that it is easier to approximate a Gaussian distribution than to approximate an arbitrary nonlinear function or transformation(Julier and Uhlmann, 1997). The UT method can be summarized as follows(Simon, 2006a).

Consider propagating a n-element vector x through an arbitrary nonlinear function $y = h(x)$ to generate the mean and covariance of y, denoted as $\bar{y}_{ukf}$ and P_{ukf}. The ensemble mean and covariance of the transformed vectors approximate the true mean and covariance up to the third order(Simon, 2006a). Assume x has mean $\bar{x}$ and covariance P.

1. Form 2n sigma point vectors $x^{(i)}$ as follows:

$$x^{(i)} = \bar{x} + \tilde{x}^{(i)} \quad i = 1, \dots, 2n$$

$$\tilde{x}^{(i)} = (\sqrt{nP})_i^T \quad i = 1, \dots, n$$

$$\tilde{x}^{(n+i)} = -(\sqrt{nP})_i^T \quad i = 1, \dots, n \tag{3.18}$$

   Where $\sqrt{nP}$ is the matrix square root of nP such that $(\sqrt{nP})^T\sqrt{nP} = nP$, and $(\sqrt{nP})_i$ is the ith row of $\sqrt{nP}$.

2. Transform the sigma points as follows:

$$y^{(i)} = h\left(x^{(i)}\right) \quad i = 1, \dots, 2n \tag{3.19}$$

3. Approximate the mean and covariance of y as follows:

$$\bar{y}_{ukf} = \frac{1}{2n}\sum_{i=1}^{2n} y^{(i)}$$

$$P_{ukf} = \frac{1}{2n}\sum_{i=1}^{2n}\left(y^{(i)} - y_u\right)\left(y^{(i)} - y_u\right)^T \quad (3.20)$$

Figure 3.2 shows an example of 2-dimentional system to illustrate the idea of EKF and UKF. The left plot shows the true mean and covariance propagation using Monte-Carlo sampling; the middle plot shows the result when using linearization approach as in EKF; the right plot shows the performance of the UT (note only 5 sigma points are required). The superior performance of the UT is clear.

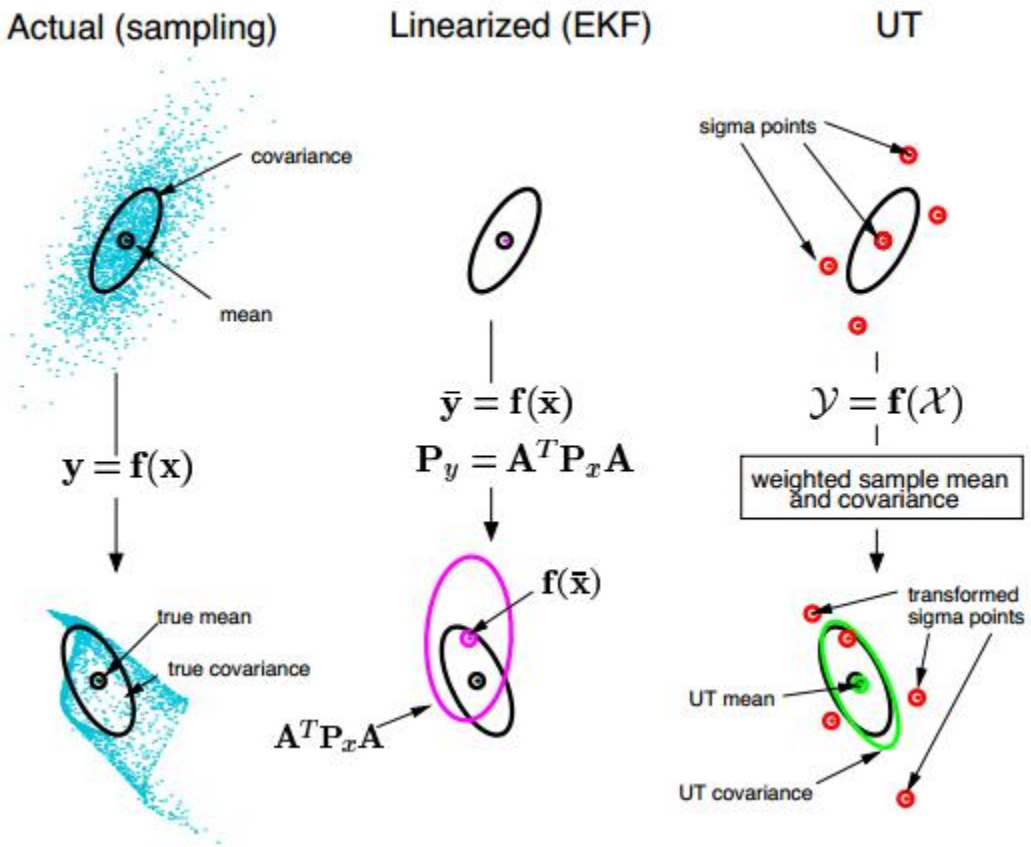

Figure 3.2: Comparison of different linearization methods for mean and covariance propagation. a) actual, b) first-order linearization (EKF), c) UT.(Wan and Van der Merwe, 2000).

3.4.2 Unscented Kalman Filter

Considering the nonlinear state and measurement equations as (3.4)-(3.5), the UKF algorithm can be states as follows(Haugen et al., 2012b).

Initial step ($k = 0$):

-Aposteriori state estimate:

$$x_{apost}(t_0) = E(x_0) \tag{3.21}$$

-Covariance of estimation error:

$$P_{apost}(t_0) = E\begin{Bmatrix}[x(t_0) - x_{apost}(t_0)] \\ [x(t_0) - x_{apost}(t_0)]^T\end{Bmatrix} \tag{3.22}$$

For time steps $k = 1, 2, 3, ...$

1. Time updates:
   (a) Calculate $2n$ sigma points(n is the number of states) based on the available aposteri estimate:

$$x_\sigma^{(i)}(t_{k-1}) = x_{apost}(t_{k-1}) + x^{(i)T}, \;\; i = 1, \dots, 2n \tag{3.23}$$

$$x^{(i)} = \left(\sqrt{nP_{apost}(t_{k-1})}\right)_i^T, i = 1, \dots, n \tag{3.24}$$

$$x^{(i+n)} = -\left(\sqrt{nP_{apost}(t_{k-1})}\right)_i^T, i = 1, \dots, n \tag{3.25}$$

   Where $\sqrt{}$ means matrix square root, and sub index i means $i'th$ row.

   (b) Propagate the sigma points using the (discrete-time) system function f(below, f_c is the continuous-time system function):

$$x_\sigma^{(i)}(t_k) = f\left[x_\sigma^{(i)}(t_{k-1}), u(t_{k-1})\right]$$

$$= x_\sigma^{(i)}(t_{k-1}) + T_s f_c\left[x_\sigma^{(i)}(t_{k-1}), u(t_{k-1})\right] \text{(Explicit Euler)} \tag{3.26}$$

   (c) Calculate the apriori state estimate as the mean values of the propagated, transformed sigma points:

$$x_{apri}(t_k) = \frac{1}{2n}\sum_{i=1}^{2n} x_\sigma^{(i)}(t_k) \tag{3.27}$$

   (d) Calculate the apriori state error covariance:

$$P_{apri}(t_k) = \frac{1}{2n}\sum_{i=1}^{2n}\left[x_\sigma^{(i)}(t_k) - x_{apri}(t_k)\right]\left[x_\sigma^{(i)}(t_k) - x_{apri}(t_k)\right]^T + Q(t_k) \tag{3.28}$$

2. Measurement updates:
   (a) Calculate $2n$ sigma points(n is the number of states) based on the available apriori estimate:

$$x_\sigma^{(i)}(t_k) = x_{apri}(t_k) + x^{(i)T}, \;\; i = 1, \dots, 2n \tag{3.29}$$

$$x^{(i)} = \left(\sqrt{nP_{apri}(t_k)}\right)_i^T, i = 1, \dots, n \quad (3.30)$$

$$x^{(i+n)} = -\left(\sqrt{nP_{apri}(t_k)}\right)_i^T, i = 1, \dots, n \quad (3.31)$$

Where √ means matrix square root, and sub index i means $i'th$ row.

(b) Transform the state estimate sigma points to corresponding measurement 'sigma points' using the measurement function h:

$$y_\sigma^{(i)}(t_k) = h\left[x_\sigma^{(i)}(t_k)\right] \quad (3.32)$$

(c) Calculate the predicted measurement as the mean values of the transformed measurement 'sigma points':

$$y_{pred}(t_k) = \frac{1}{2n}\sum_{i=1}^{2n} y_\sigma^{(i)}(t_k) \quad (3.33)$$

(d) Estimate the covariance of predicted measurement and take the measurement noise into account:

$$P_y(t_k) = \frac{1}{2n}\sum_{i=1}^{2n}\left[y_\sigma^{(i)}(t_k) - y_{pred}(t_k)\right]\left[y_\sigma^{(i)}(t_k) - y_{pred}(t_k)\right]^T + R(t_k) \quad (3.34)$$

(e) Estimate the cross variance between apriori state estimate and predicted measurement:

$$P_{xy} = \frac{1}{2n}\sum_{i=1}^{2n}\left[x_\sigma^{(i)}(t_k) - x_{apri}(t_k)\right]\left[y_\sigma^{(i)}(t_k) - y_{pred}(t_k)\right]^T \quad (3.35)$$

(f) Calculate Kalman filter gain:

$$K(t_k) = P_{xy}P_y^{-1} \quad (3.36)$$

(g) Calculate the innovation process:

$$e(t_k) = y(t_k) - y_{pred}(t_k) \quad (3.37)$$

(h) Calculate the state estimate(aposteriori estimate, or corrected estimate), which is used as the applied estimate:

$$x_{apost}(t_k) = x_{apri}(t_k) + K(t_k)e(t_k) \quad (3.38)$$

(i) Estimate the covariance of aposteriori state estimate:

$$P_{apost}(t_k) = P_{apri}(t_k) - K(t_k)P_yK(t_k)^T \quad (3.39)$$

The algorithm above is based on the assumption that the process and measurement function are linear with respect to the noise as shown in Equations (3.4) and (3.5). Compared to EKF, UKF doesn't need to calculate Jacobian matrices, and it can solve state estimation problem in case of arbitrary Gaussian high nonlinearity.

3.5 Particle filter

This section presents the PF, which is a probability-based estimator. In Section 3.5.1, a Bayesian approach to state estimate will be briefly discussed. And based on the approach, we will derive the PF algorithm.

3.5.1 Bayesian state estimation

Bayesian estimation is an optimal estimation method that constructs probability distribution of the target state based on the available information. In a general discrete-time state-space model，the system and measurement equations are given as follows:

$$x(t_{k+1}) = f_k(x(t_k), w(t_k)) \tag{3.40}$$

$$y(t_k) = h_k(x(t_k), v(t_k)) \tag{3.41}$$

The noise sequence $w(t_k)$ and $v(t_k)$ are assumed to be independent and white with known probability density functions (pdf). The goal of a Bayesian estimator is to approximate the conditional pdf of $x(t_k)$based on measurements $y(t_1), y(t_2), \dots, y(t_k)$. The conditional pdf is denoted as

$$p(x(t_k)|Y(t_k)) = pdf\ of\ x(t_k)\ conditioned\ on\ measurements\ y(t_1), y(t_2), \dots, y(t_k) \tag{3.42}$$

According to Gordon et al. (1993a) and Simon (2006a), the recursive equations of the Bayesian state estimation filter can be summarized as follows.

The recursive Bayesian state estimator:

1. Assuming that the pdf of the initial state $p(x(t_0))$ is known, initialize the estimator as follows:

$$p(x(t_0)|Y(t_0)) = p(x(t_0)) \tag{3.43}$$

2. For $k = 1,2,..$, perform the following.
   (a) The apriori pdf $p(x(t_k)|Y(t_{k-1})) =$ is computed from the filtering distribution $p(x(t_{k-1})|Y(t_{k-1}))$ at time $k-1$, which can be deduced as

$$p(x(t_k)|Y(t_{k-1})) = \int p(x(t_k)|x(t_{k-1}))\, p(x(t_{k-1})|Y(t_{k-1}))dx(t_{k-1}) \tag{3.44}$$

   (b) The aposteriori pdf $p(x_k|Y_k)$ can be derived as

$$p(x(t_k)|Y(t_k)) = \frac{p(y(t_k)|x(t_k))p(x(t_k)|Y(t_{k-1}))}{\int p(y(t_k)|x(t_k))p(x(t_k)|Y(t_{k-1}))dx(t_k)} \tag{3.45}$$

Execution of Equations (3.44) and (3.45) in a loop is the basic method of recursive Bayesian estimation.

3.5.2 Particle filter algorithm

Considering the nonlinear state and measurement equations as (3.40)-(3.41), the PF algorithm can be states as follows(Simon, 2006a).

1. Assuming that the pdf of the initial state $p(x_0)$ is known, randomly generate N initial particles on the basis of pdf $p(x_0)$. These particles are denoted as $x_{apost}^{(i)}(t_0),\ i = 1, \dots, N$. The parameter *N* is chosen by the user as a tradeoff between computational effort and estimation accuracy.

2. For $k = 1,2, \dots$, do the following.
   (a) Perform the time propagation step to obtain apriori particles $x_{apri}^{(i)}(t_k)$ using the known process equation and the known pdf of process noise.
   $$x_{apri}^{(i)}(t_k) = f_{k-1}\left(x_{apost}^{(i)}(t_{k-1}), w^{(i)}(t_{k-1})\right), i = 1, \dots, N \quad (3.46)$$
   Where each $w^{(i)}(t_{k-1})$ noise vector is randomly generated on the basis of known pdf of $w(t_{k-1})$.
   (b) Compute the relative likelihood q_i of each particle $x_{apri}^{(i)}(t_k)$, conditioned on the measurement $y(t_k)$. This is done by evaluating the pdf $p\left(y(t_k)\middle|x_{apri}^{(i)}(t_k)\right)$ on the basis of the nonlinear measurement equation and the pdf of the measurement noise.

   (c) Scale the relative likelihoods obtained in the previous step as follows:
   $$q_i = \frac{q_i}{\sum_{j=1}^{N} q_j} \quad (3.47)$$
   Now the sum of all the likelihoods is equal to one.

   (d) Generate a set of aposteriori particles $x_{apost}^{(i)}(t_k)$ on the basis of the relative likelihood q_i. This is called the resampling step. A straightforward resampling method can be formed as follows(Ristic et al., 2004).

   - Generate a random number r that is uniformly distributed on [0,1].
   - Compare the sum of likelihood q_i with r at each step. If
   $$\sum_{m=1}^{j-1} q_m < r, \sum_{m=1}^{j} q_m \geq r \quad (3.48)$$
   Then the new particle $x_{apost}^{(i)}(t_k)$ is set equal to the old particle $x_{apri}^{(j)}(t_k)$,
   $$x_{apost}^{(i)}(t_k) = x_{apri}^{(j)}(t_k)\ with\ probability\ q_j\ , i, j = 1, \dots, N \quad (3.49)$$

   (e) As we have get a set of particles $x_{apost}^{(i)}(t_k)$ that are distributed according to pdf $p(x(t_k)|y(t_k))$, we can compute the desired state mean and covariance of this pdf.

3.6 Implementation issues

In this section, a few implementation issues that often arise in the application of Kalman filters are discussed.

While the calculations of Jacobian matrices for the EKF may seem straightforward, in practice these calculations are prone to errors. Many mistakes in EKF implementations can be happen either when calculation or coding of the Jacobian matrices, which can be time-consuming and complex. The UKF does not require this calculation, and therefore is typically easier and faster to implement than an EKF. However, the EKF usually seems to be more computationally efficient for most applications, as long as the EKF is using analytically determined Jacobian matrices instead of numerical derivatives. Because UKF requires the calculation of the nonlinear functions multiple times (proportional to the number of states), while the EKF requires this calculation only once. A critical phase of the filter implementation is initialization. An initial state estimate and error covariance can be chosen based on mainly intuition. The measurement noise covariance can be obtained by calibration of the real sensors. The process noise which drives the bias estimation can be tuned to improve the convergence. The biases are supposed to be constant or slow varying, so the process noise must have a small covariance.

From the PF algorithm, it's easy to find that given a large number of particles, the resampling process will be time consuming. However, there must be enough particles to guarantee the approximation accuracy of sampling and maintain a small variance. Especially when dealing with a high dimensional state space, insufficient number of particles will lead to the failure of the algorithm. Unfortunately, there is no unified rule on how to choose the number of particles. It relies on the experience and known specification. The three estimators are implemented in MATLAB and can be seen in Appendix D.

4 Simulation–State estimation with simulated reactor

4.1 Introduction

In this chapter, the three state estimators will be applied to a simulated AD reactor.

Evaluation Methods will be used as follows.

(1) To determine how well the EKF, UKF and PF algorithms are performing, we need to compare the estimated data to the 'real' data which is generated by a simulator. With the real data, we can calculate the root mean square error (RMSE) for each test. For real and estimated state, $x(t_k)$ and $x_{apost}(t_k)$, RMSE is defined by

$$RMSE_x = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(x(t_k) - x_{apost}(t_k)\right)^T\left(x(t_k) - x_{apost}(t_k)\right)} \tag{4.1}$$

(2) Computational efficiency. The relative efficiency of each algorithm is determined based on computational time and degree of precision required.

It may be very useful to test an estimator (in code function) with a simulated process before applied to the real (physical) process. If the mathematical model used in the simulator is an accurate representation of the real process, we may even tune the estimator parameters (e.g.Q, R) using the simulator. Besides, analysis of robustness of the estimator is focused on, that is, how these estimators react to model error.

In Section 4.2, we are going to explore how the Kalman filters react to model with or without state augmentation. In the following Sections, we will use the model as equations (3.5) and (3.6) with augmentative state. Where, it is assumed that the noise sequences are pure white, zero-mean, and uncorrelated.

4.2 State augmentation

As we have discussed in Chapter 2, state augmentation consists of defining new states. In this case, we have generated an augment model with state $S_{vs,in}$. Thus, to recognize the value of augmentative state, we will compare the performance of the estimators with and without augmentation.

4.2.1 Simulation setup

The model and estimators are implemented in MATLAB. Sample time is 0.1 day for all the simulation tasks. The ordinary Kalman filters and augmented Kalman filters are equally tuned.

To tune the estimator, it is vital to choose a steady-state operating point as a start point for all the filters. For example, EKF is based on a linearized model around the steady-state point, so fast

rate of convergence to true value and a stronger stability can be achieved by doing this. In this system, $x_{apost}(t_0)$, R can be calculated from the real-time series of data. So $P_{apost}(t_0)$ and Q can be adjusted here. Measurement noise covariance R is usually set to a constant matrix. Here R is one element matrix as there is only one output in our system. As we know, the standard deviation of a representative real time-series of F_{meth} is 1.2 L CH4/d. So

$$R = var(F_{meth}) = [std(F_{meth})]^2 = 1.2^2 = 1.44 \tag{4.2}$$

$$x_{apost}(t_0) = [5.2155 \;\; 1.0094 \;\; 1.3128 \;\; 0.3635 \;\; 30.2]^T \tag{4.3}$$

(1) Initial state estimation error $P_{apost}(t_0)$ is a diagonal matrix which has the same dimension with system matrix. From Eqs. (3.12) and (3.14), we know a small $P_{apost}(t_0)$ will lead to a small Kalman gain and thus the Kalman filter places more importance on model-predicted estimation of states. Conversely, a small measurement noise and a large estimation-error covariance estimate results in the measured value of states dominating the aposteriori state estimate. As initial state $x_{apost}(t_0)$ is quite close to the real value, a properly small $P_{apost}(t_0)$ like zero matrix is more fit here.

$$P_{apost}(t_0) = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0.01 \end{bmatrix} \tag{4.4}$$

(2) Process noise covariance Q is a diagonal matrix with a same dimension with system matrix. A good model means a good estimation of the process, and a small Q. The parameters belonging to Q on the other hand are much more difficult to obtain, and this is why the Kalman filter is known as difficult to tune. Here we are not sure about the quality of model. From experience, the the standard deviation of each of the state disturbances can be guessed as one percent of the representative value of the respective state variable.

$$Q = \left[1\% * diag\left(x_{apost}(t_0)\right)\right]^2 \tag{4.5}$$

$$Q = \begin{bmatrix} 0.002725 & 0 & 0 & 0 & 0 \\ 0 & 0.0001 & 0 & 0 & 0 \\ 0 & 0 & 0.000169 & 0 & 0 \\ 0 & 0 & 0 & 0.000013 & 0 \\ 0 & 0 & 0 & 0 & 0.1024 \end{bmatrix} \tag{4.6}$$

4.2.2 Augmented Kalman filters

A good way to see the value of augmented Kalman filters is to impose a model error. Model error is caused by wrong or inaccurate parameters when building the model. There can be model error regarding model parameters, and model error regarding assumption about initial state, and both these types of model errors are easily to implement in a simulator based test system. An effective way to simulate the model error is to impose an error on the initial state. Another way to reduce the influence of model error is to use primary nonlinear model. A nonlinear model should represent more dynamic characteristics of the real process than a linearized model. That's why we prefer the nonlinear model both for calculation of the states and output inside the algorithm.

Here we simulate a model error with the augmentative state. The value of S_{vsin} from lab analysis at the selected operating point is

$$S_{vsin} = 30.2\ g/L \tag{4.7}$$

To impose a model error, we assume the real process value is

$$S_{vsin}(real) = S_{vsin} + \Delta S_{vsin} = 30.2 + 15 = 45.2\ g/L \tag{4.8}$$

$S_{vsin}(real)$ will be used in the simulated reactor while S_{vsin} will be used as initial state in the state estimator. Thus we can detect how the augmented Kalman filters response to the error model. Figure 4.1-4.3 show the augmented Kalman filters performs well when model error added.

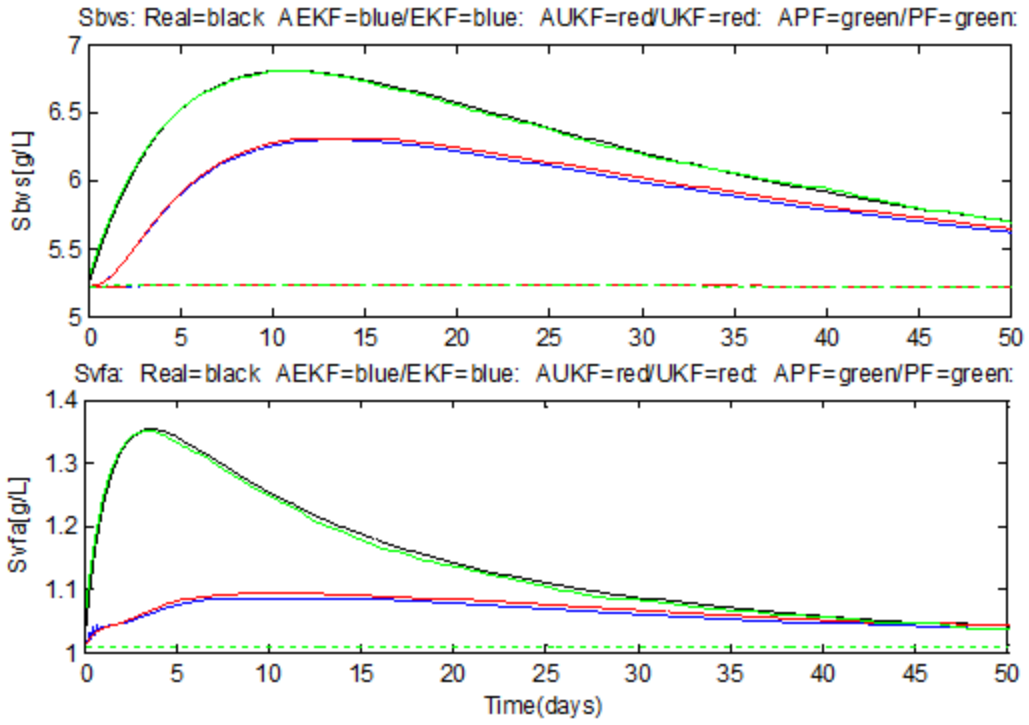

Figure 4.1 : State estimation of simulated reactor, S_bvs and S_vfa.

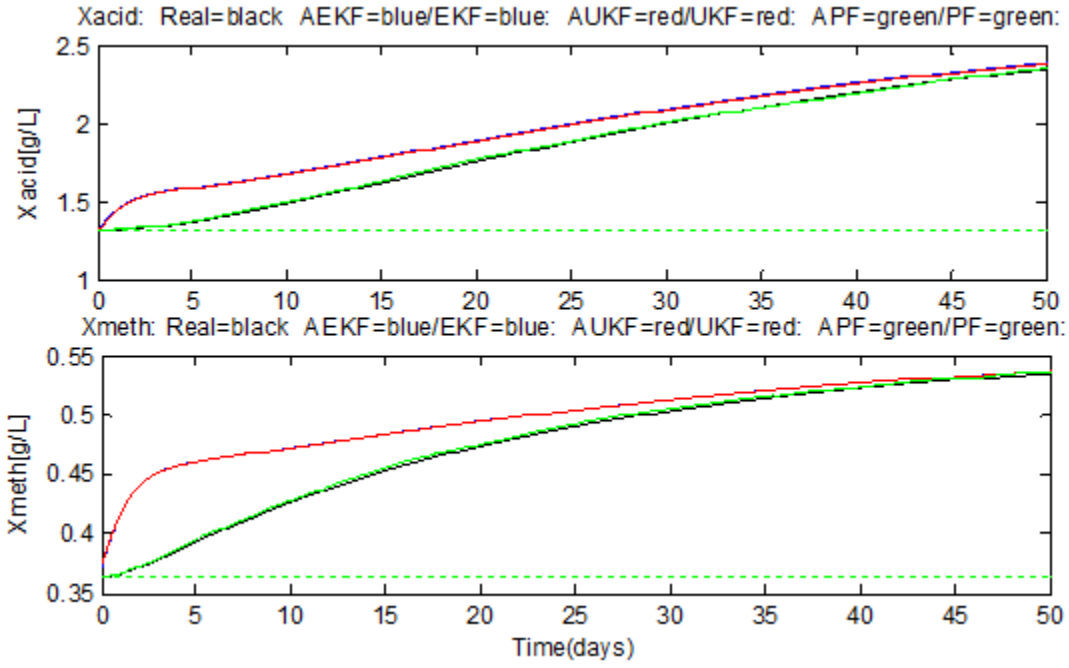

Figure 4.2: State estimation of simulated AD reactor, X_acid and X_meth.

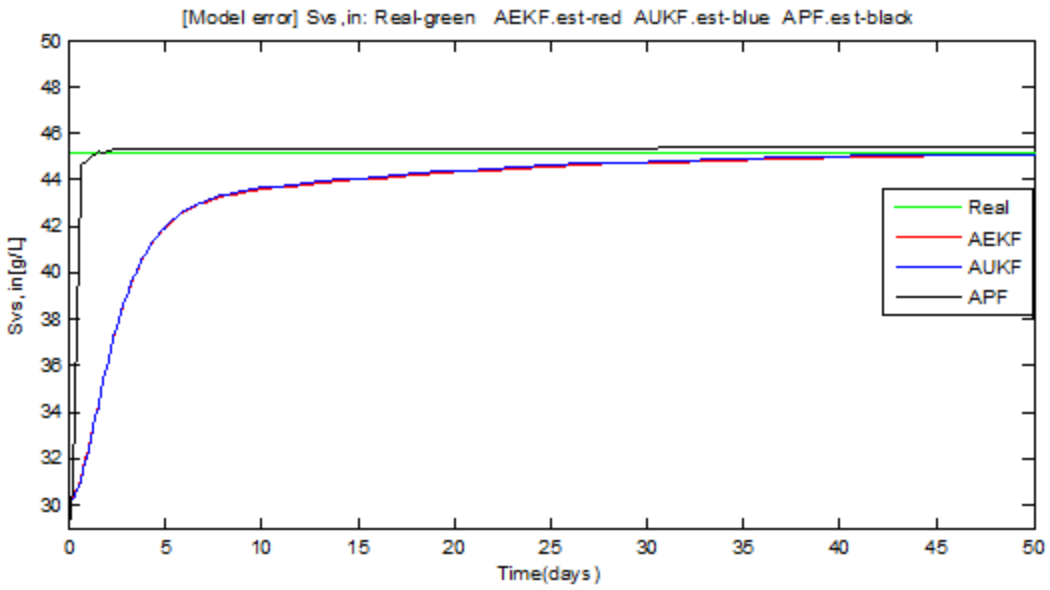

Figure 4.3: State estimation of simulated AD reactor, augmented state S_vsin.

Figure 4.1-4.2 illustrate that augmentation methods have superior performance compared to the ordinary nonlinear filtering versions. Figure 4.3 shows the augmented methods perform very good when model error added.

Figure 4.4 shows that the Kalman gains for each estimator are almost the same. But be noted that the two algorithms have different calculation methods for Kalman gain. EKF use the iterated Jacobian matrices, while UKF calculation is based on the chosen sigma points. So in this case, it proves our model is adaptable for both algorithms.

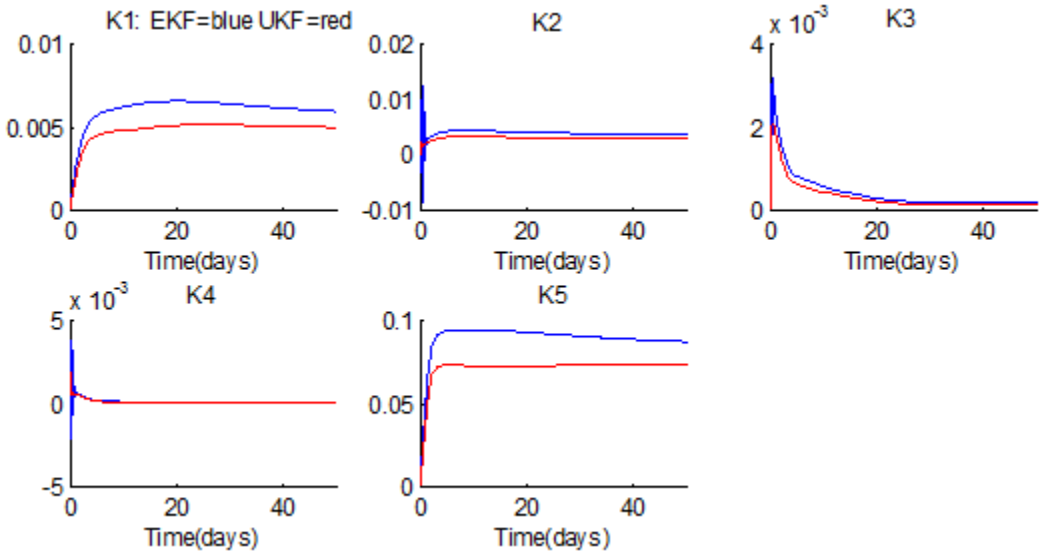

Figure 4.4: Kalman gains for the augmented EKF and augmented UKF.

4.3 Tuning the particle filter

The aim of the section is find the proper setup for particle filter with regard to number of particles. The number of particles N is the most direct and important design parameter in the PF.

Figure 4.5-4.7 clearly show how the particle filter performs when number of particles changes.

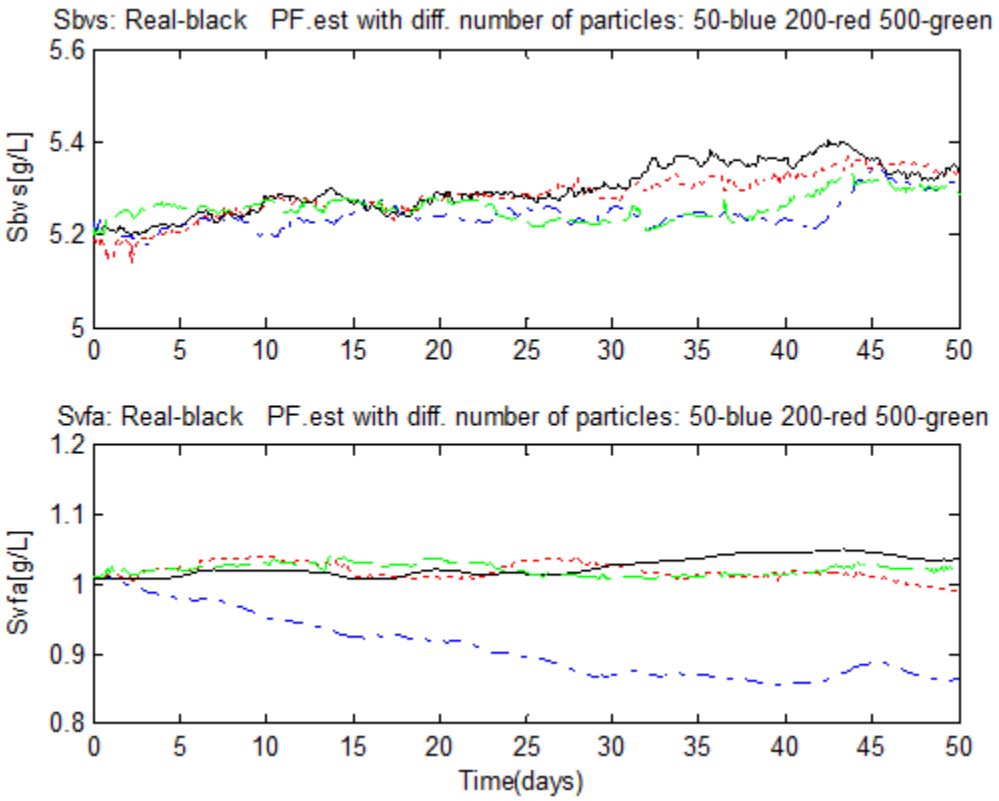

Figure 4.5: Particle filter estimation with different number of particles, S_bvs and S_vfa.

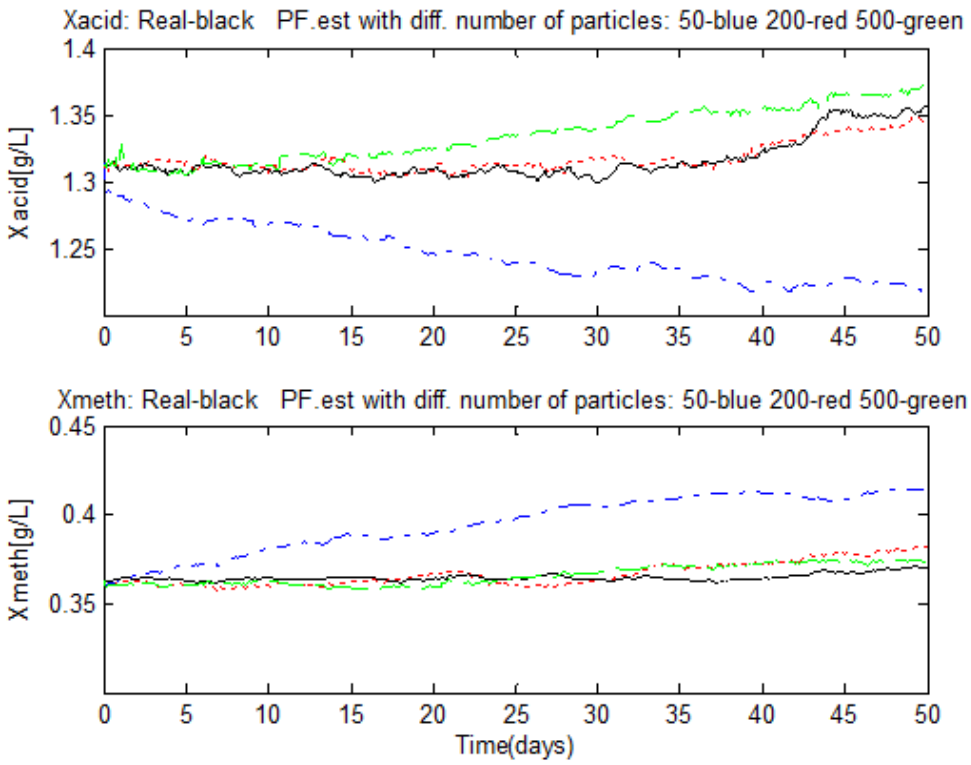

Figure 4.6: Particle filter estimation with different number of particles, X_acid and X_meth.

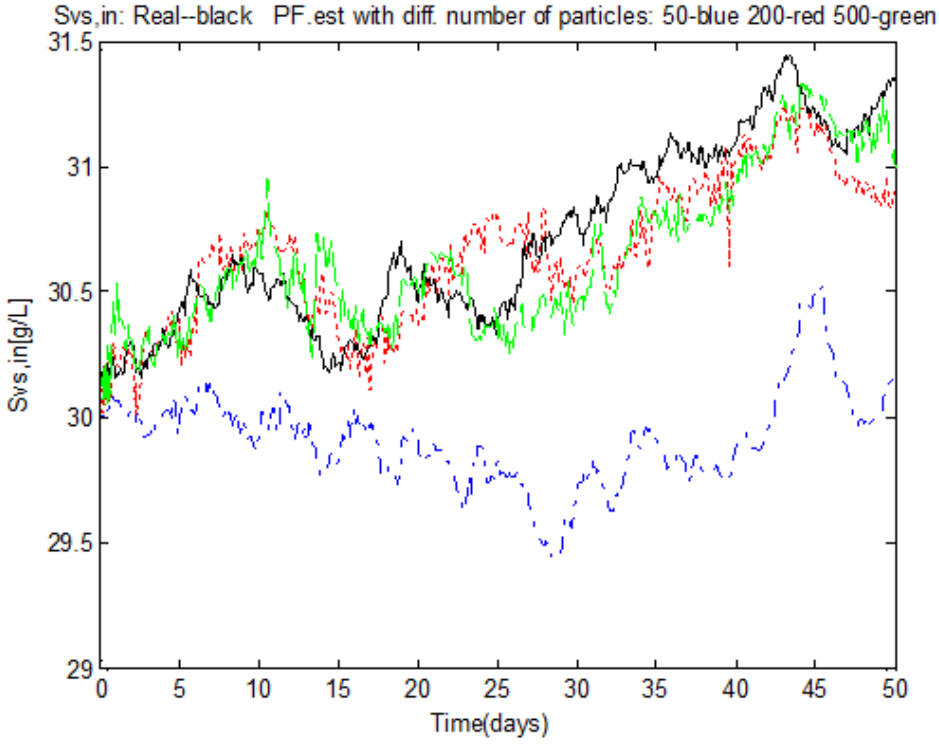

Figure 4.7: Particle filter estimation with different number of particles, Svsin.

From Figure 4.5-4.7, we can see the blue line which represents 50 particles goes far away from the real data and show a trend of divergence. While the others with 200 and 500 particles show a robust performance. A series of tests show that the more particles, the better estimation accuracy. And when the number is above 200, a satisfactory result can be obtained.

Table 4.1 shows the RMSE for each state estimate across the different number of sampling points. Note that when the number of samples increases, the RMSE slightly decreases, but execution time of the algorithm increase a lot.

Table 4.1: RMSE and execution time with different number of particles.

Numbers / **Performance**	**N=50**	**N=200**	**N=500**
Elapsed time(sec)	0.869	5.910	31.648
$\boldsymbol{RMSE_{S_{bvs}}}$	0.005	0.004	0.008
$\boldsymbol{RMSE_{S_{vfa}}}$	0.024	0.010	0.003
$\boldsymbol{RMSE_{X_{acid}}}$	0.017	0.016	0.009
$\boldsymbol{RMSE_{X_{meth}}}$	0.024	0.005	0.008
$\boldsymbol{RMSE_{S_{vs,in}}}$	0.027	0.021	0.024

4.4 Comparison of the filters

In this simulation task, the same initial value will be used as Section 4.2.1.

4.4.1 Performance with respect to different process noise.

In this task, we set three different process noise covariance as 0.01*Q, Q and 100*Q. Figure 4.8-4.9 show the estimation error when various values of Q are used in the Kalman filters.

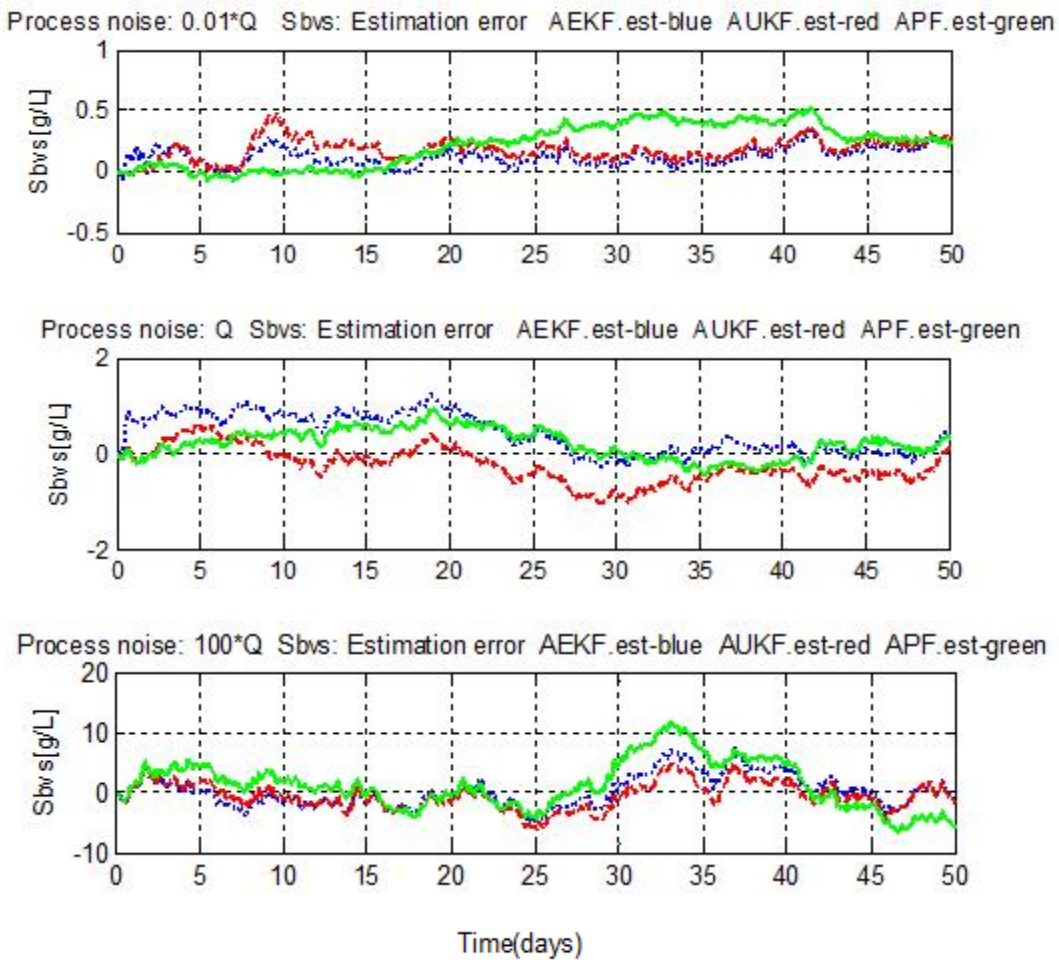

Figure 4.8: Estimation error with respect to various process noise covariance, S_bvs.

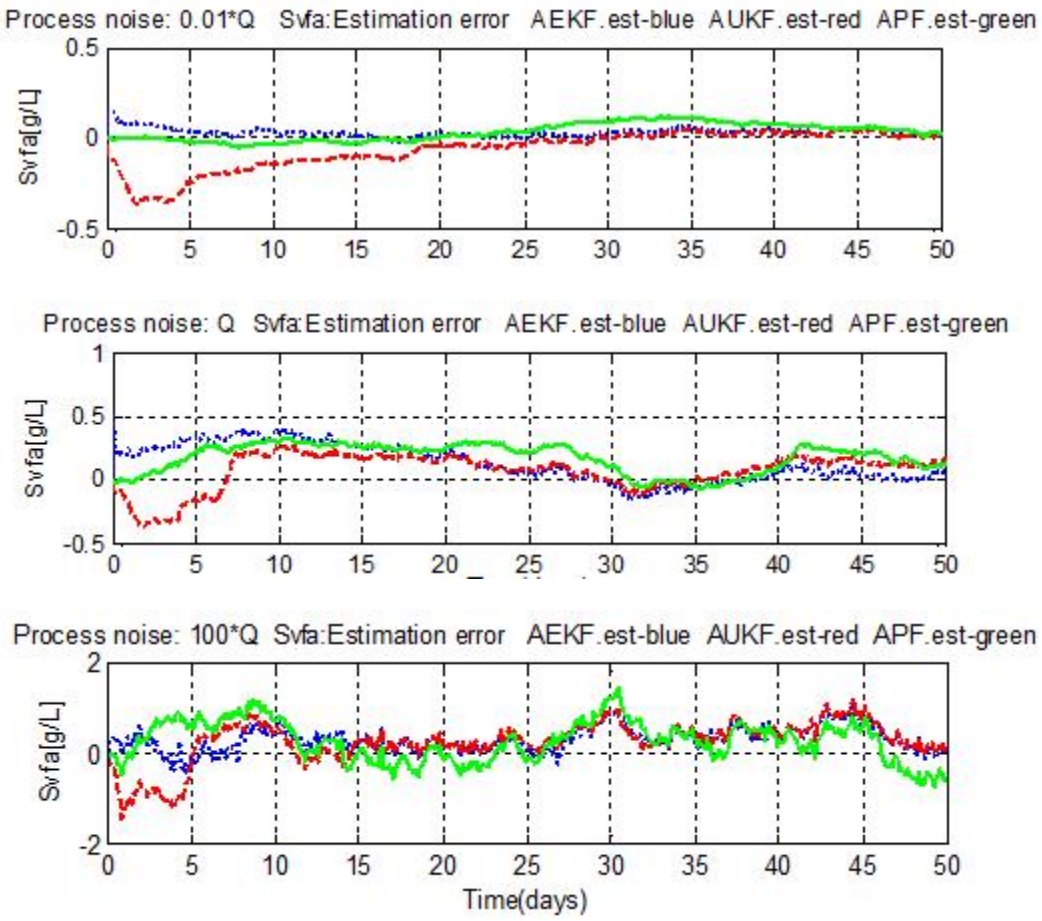

Figure 4.9: Estimation error with respect to various process noise covariance, S_vfa.

As the process noise gets larger, the estimation error becomes larger, as we can see the amplitude of y-axis increases. This means the estimation is becoming more inaccurate and noisy. The reason behind is that the Kalman gain will converge to a larger steady-state value when Q is larger, which cause the filters more dependent on noisy measurements. An appropriate larger Q can be a compensation to model error, because a system with too little noise might be overly susceptible to model error(Simon, 2006a).

An interesting phenomenon is that EKF shows more stable than UKF at the first few days of simulation. And when the process noise covariance increases (i.e. 1000*Q), UKF even becomes to divergent. That means EKF is more robust to the process noise.

4.4.2 Performance with respect to different measurement noise.

It's important to know how the Kalman filters react to measurements. In this task, we set three different measurement noise covariance as 0.0144, 1.44 and 144. The simulation result can be presented as Figure 4.10-4.11.

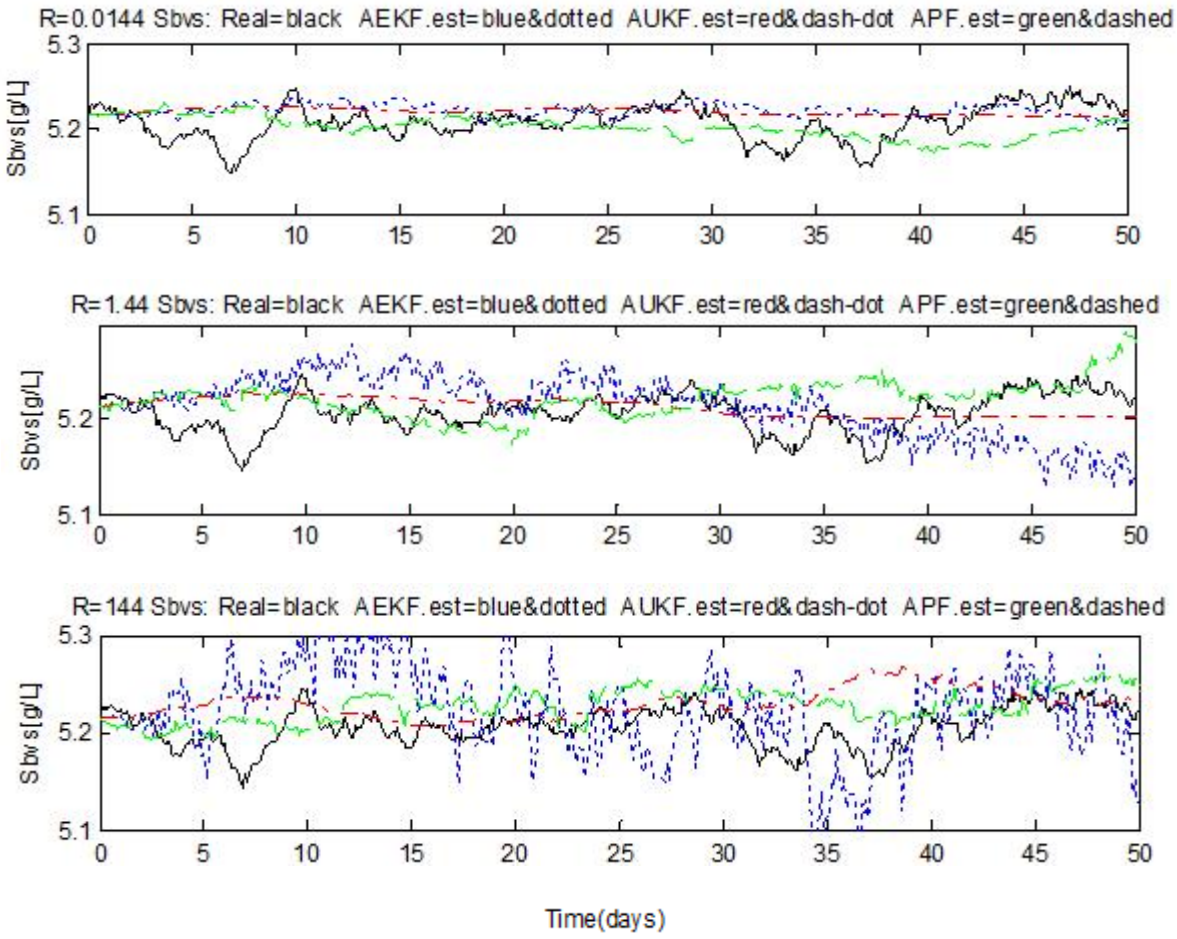

Figure 4.10: State estimation with respect to various measurement noise covariance, S_bvs.

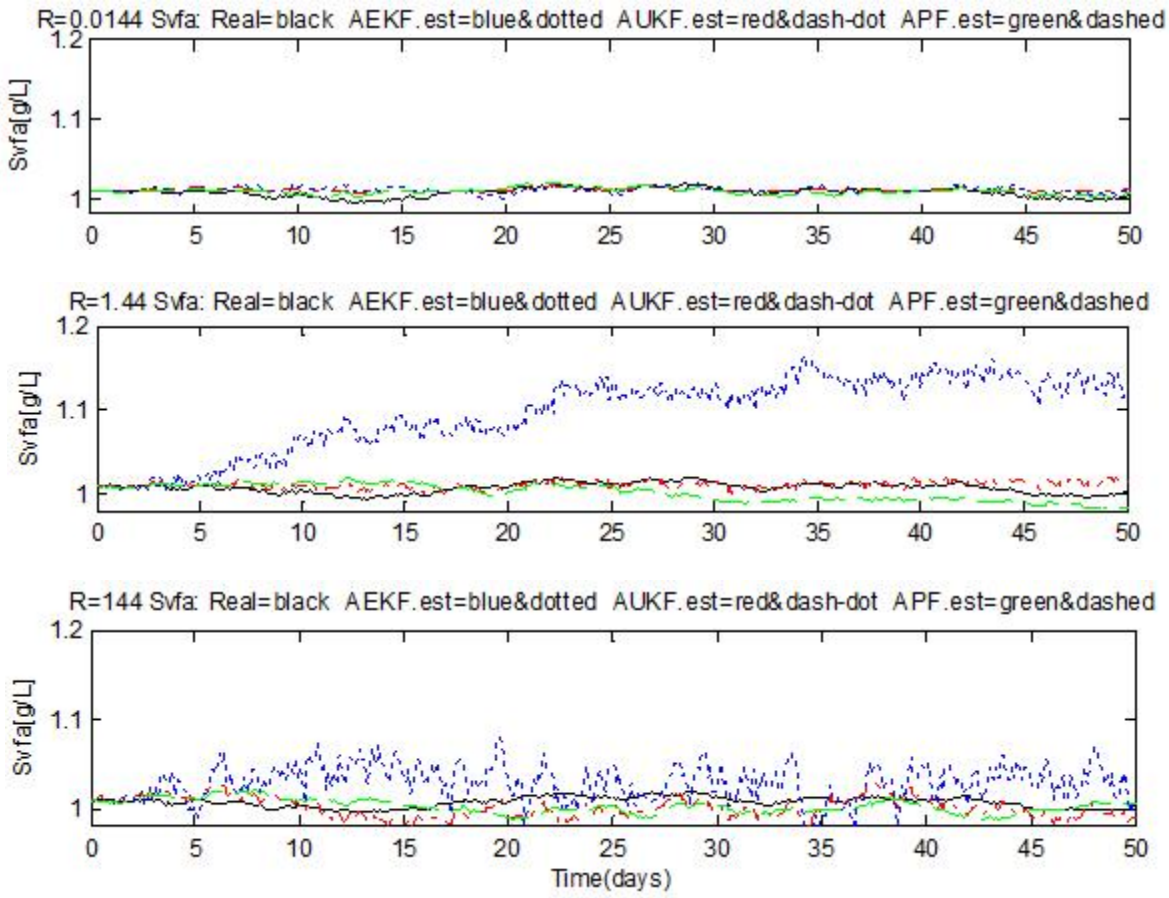

Figure 4.11: State estimation with respect to various measurement noise covariance, S_vfa.

It can be easily found from Figure 4.10-4.11, even the measurement noise increases by a factor of 100, The Kalman filters can give estimation accuracy with a small difference as before. That means all the filters are essentially immune to the measurement noise, as the algorithms put more weights on the process model. Note when $R = 144$, EKF becomes extremely sensitive, while the UKF and PF show a strong robustness regarding to this measurement noise.

4.4.3 Performance with respect to different initial error covariance.

In order to test the performance of the filters with different initial errors, three different initial conditions are applied. For initial state estimation error, we will start with $P_{apost}(t_0)$ and set the second and third test value as $10 * P_{apost}(t_0)$ and $100 * P_{apost}(t_0)$. The results can be shown as Figure 4.12 and 4.13.

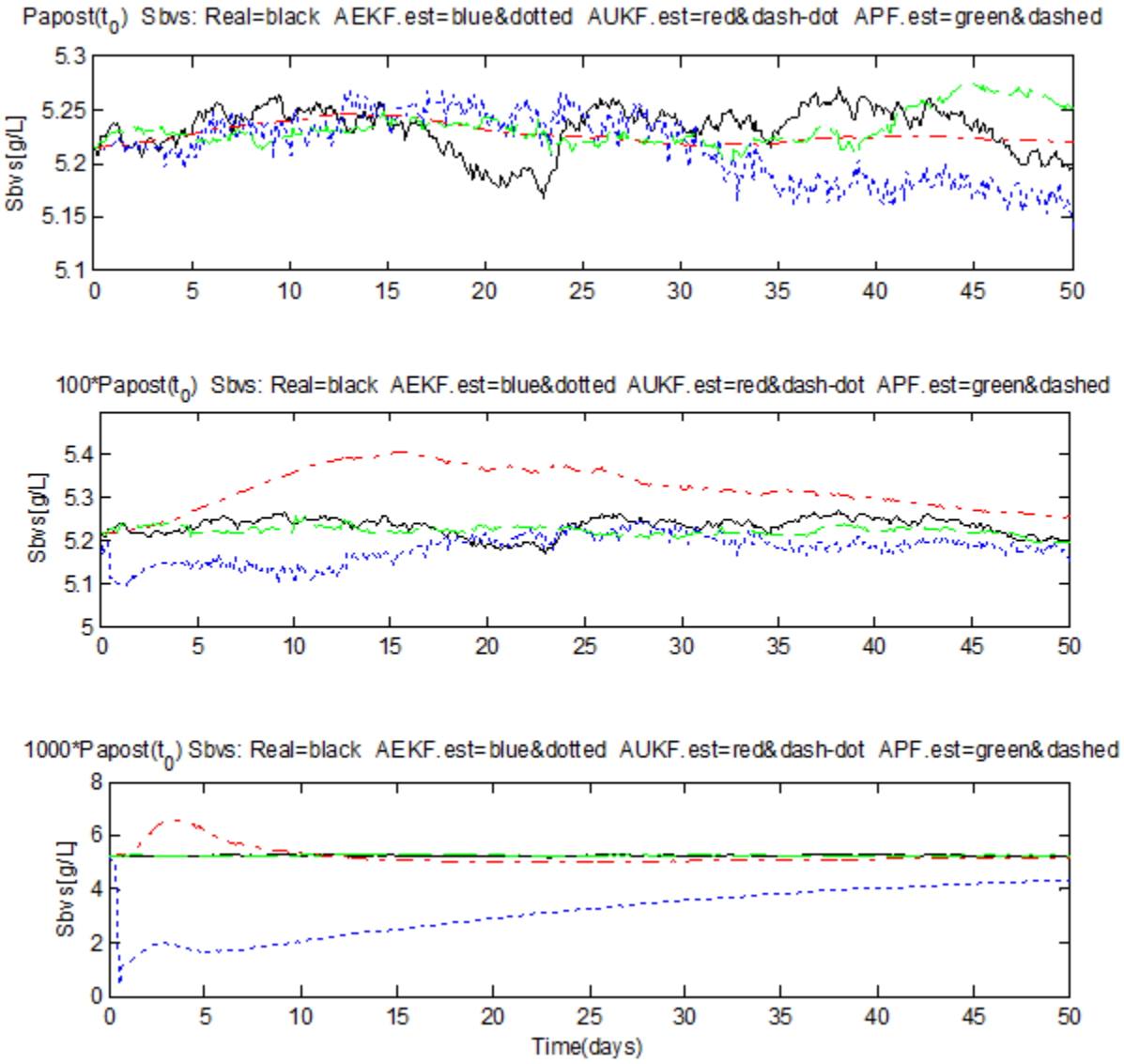

Figure 4.12: State estimation with respect to various initial state covariance, S_bvs.

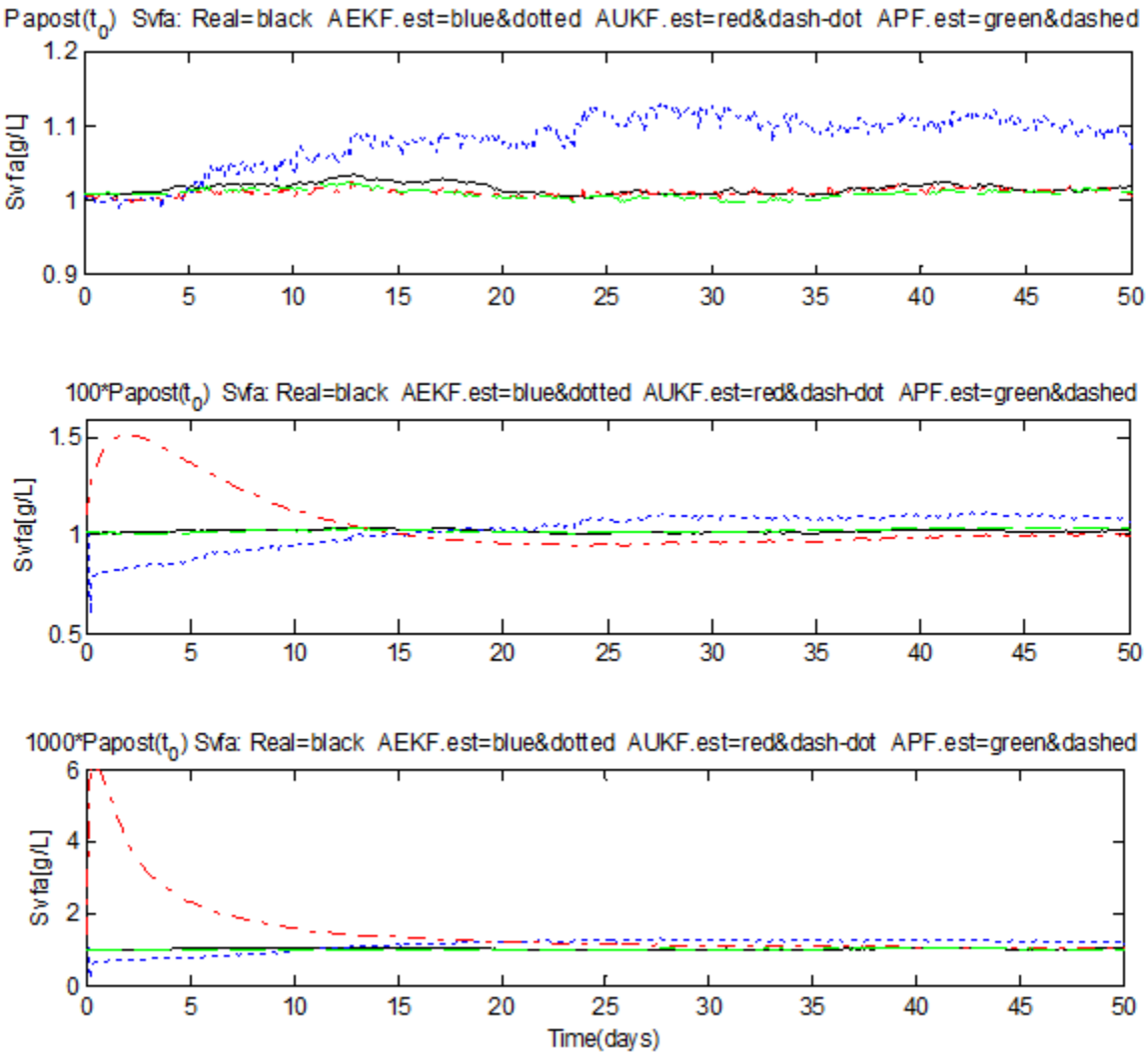

Figure 4.13: State estimation with respect to various measurement noise covariance, S_vfa.

The simulation results show that UKF, EKF and PF, give very similar results while having a small initial estimation error. However, when a large initial error (for example, $1000 * P_{apost}(t_0)$ in this case) is given, the UKF is highly susceptible and shows a slower convergence with respect to the EKF and PF, but after a settling time the performance become identical. Compare to UKF and EKF, The reason behind this may be the sigma points that UKF relies on. PF does a better job. The estimation is more stable with a faster convergence.

4.4.4 Performance comparison under same initial condition

Figure 4.14-4.16 present the simulation results of EKF (dotted), UKF (dash-dot) and PF (dashed) in 80 days. The solid, black line represents the true states.

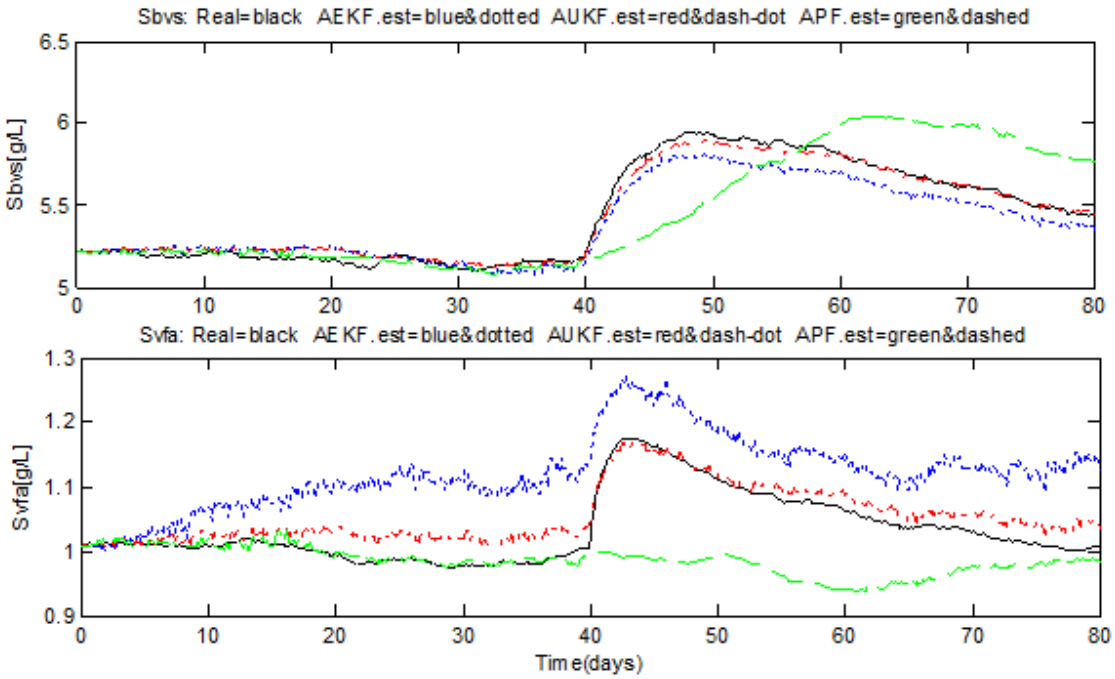

Figure 4.14: Simulation: Estimates of the states S_bvs and S_vfa, disturbance S_vsin=35 at day 40.

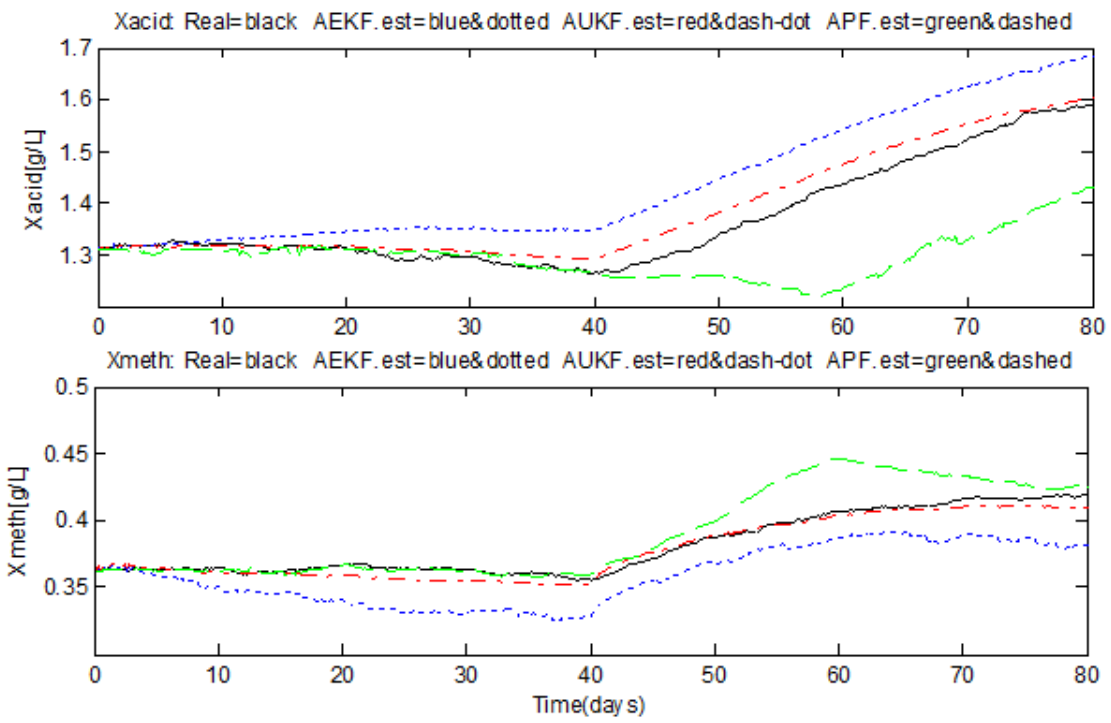

Figure 4.15: Simulation: estimates of the states X_acid and X_meth, disturbance S_vsin=35 at day 40.

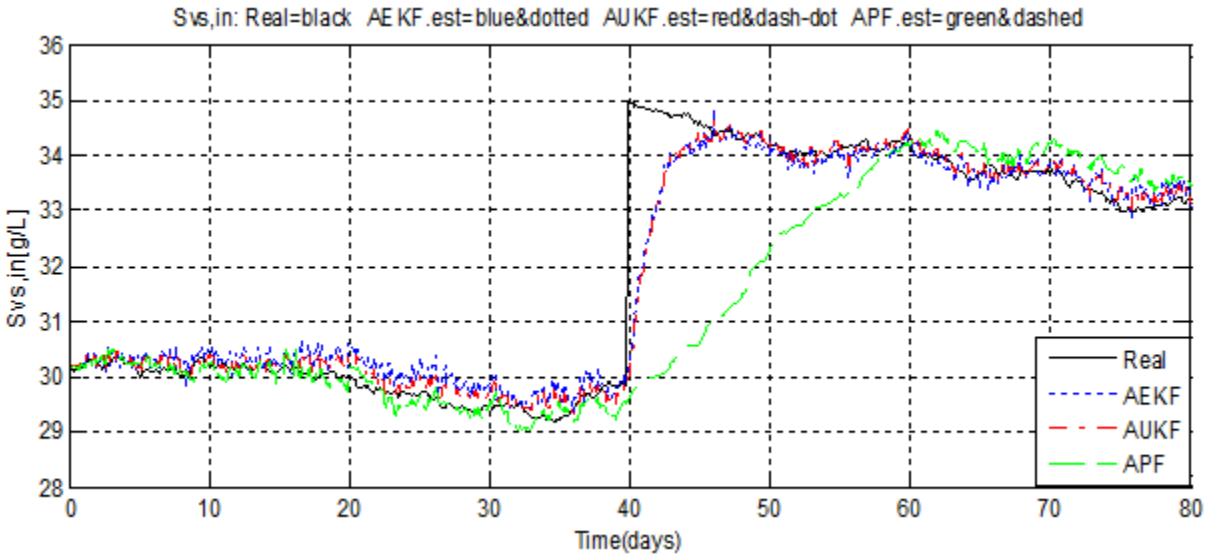

Figure 4.16: Simulation: Estimates of the state S_vsin, disturbance S_vsin=35 at day 40.

From Figure 14-16, it can be easily seen that the UKF and PF give a better estimation performance than EKF from day 0 to 40. When a disturbance happens at day 40, as S_{vsin} suddenly increases to 35, EKF and UKF show a fast response and estimate the states as well as before. In contrast, the PF shows a slow convergence and unstable estimation especially

estimation of S_{bvs} and X_{meth}. From this experiment, we find that EKF and UKF have better robust stability that PF.

To compare the estimation qualities of the EKF and the UKF, Figure 4.17 below plots the absolute value of estimation errors which are relative to the true states. The result shows that the three Kalman filters give the similar absolute estimation error for all the five states. However, for the states S_{vfa} and X_{acid}, the superior performance of the PF and UKF is clear(absolute estimation error is under 0.1 and 0.02 for the whole time axis).

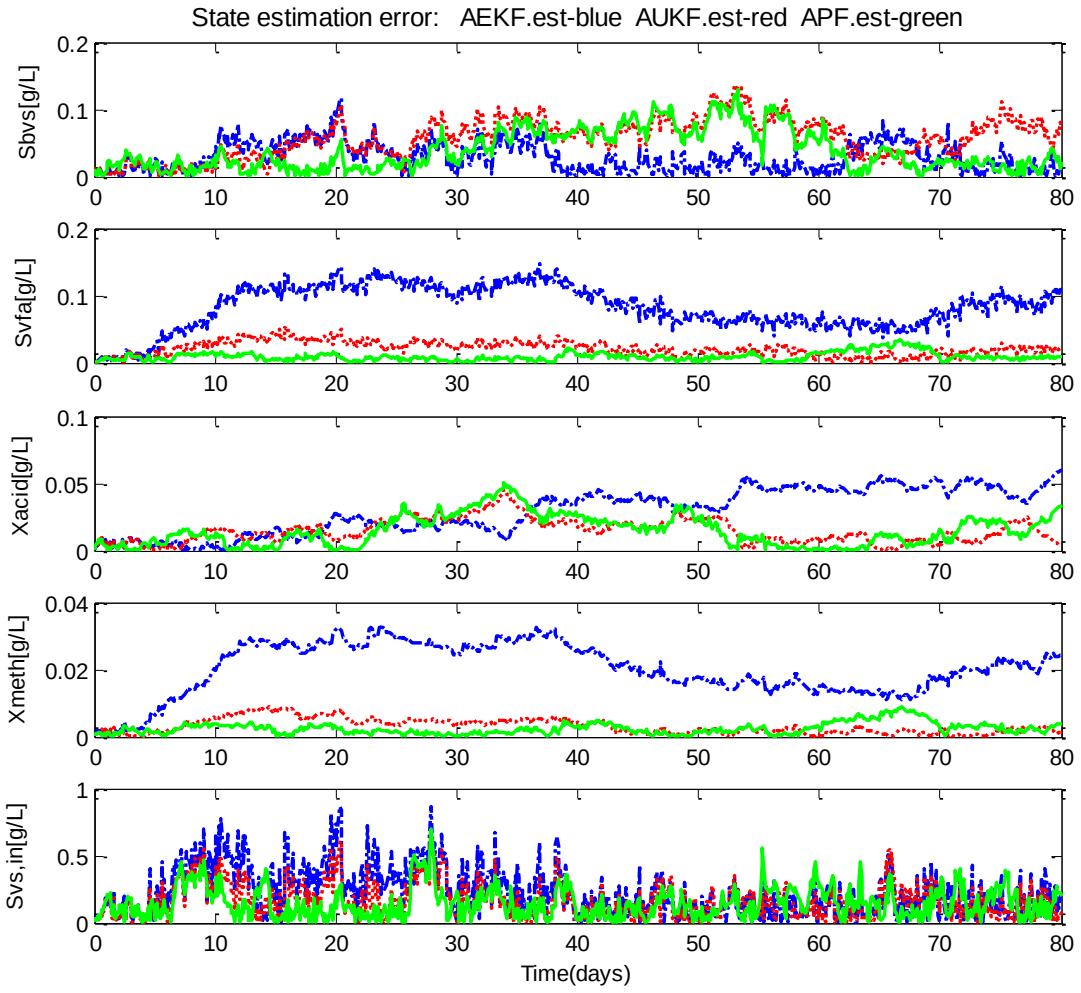

Figure 4.17: Comparison of the absolute estimation errors of EKF (dashed), UKF (dotted) and PF (solid) for each state.

Table 4.2 shows the RMSE of each algorithm. From the number, we can find the performance between EKF and UKF almost the same. PF shows better performance than the others regarding to estimation of X_{acid} and bigger error with S_{bvs}.

Table 4.2: Performance results of the nonlinear estimation filters.

Algorithm / Performance	EKF	UKF	PF
$RMSE_{S_{bvs}}$	0.0001	0.0001	0.0004
$RMSE_{S_{vfa}}$	0.0046	0.0046	0.0040
$RMSE_{X_{acid}}$	0.0037	0.0038	0.0017
$RMSE_{X_{meth}}$	0.0069	0.0067	0.0066
$RMSE_{S_{vs,in}}$	0.0042	0.0040	0.0043

Table 4.3 shows that the discrete-time UKF is the fastest filter with an elapsed time around 0.8 second, compared to 2.5 seconds of EKF and 70.9 seconds of PF. PF algorithm need more computational effort with approximately 90 times and 30 times more time consuming in terms of CPU time usage. For instance, the discrete-time UKF requires about 31% of the computational burden of the EKF, because the calculation of the Jacobian matrix in the EKF is time-consuming.

The algorithms were implemented on a 2.6 GHz machine which had 4 GB RAM. Simulation software used is MATLAB R2012a(64-bits). A sampling time of 0.1sec was considered.

Table 4.3: Elapsed time of filtering algorithms.

Algorithm / Time steps	EKF time(sec)	UKF time(sec)	PF time(sec)
800	2.5	0.8	70.9

The simulation results show that UKF is more sensitive to noise than EKF and PF. UKF is the fastest algorithm based on the same condition.

4.5 Summary

These results show that the EKF and UKF have roughly the same error in all cases. EKF has a satisfactory estimate with higher computational cost as update of Jacobian matrix in every time step. UKF has a similar performance with EKF, but susceptible to the initial condition. UKF can avoid linearization in algorithm implementation. When there is no disturbance, PF has the best performance, which unfortunately based on a high computational requirement. Also, UKF and PF are more complicated than EKF. No one algorithm has absolute advantage over another one. The only way is to consider case by case. For this AD reactor, EKF can be a good choice as a reliable estimator, and simple model used such that easily calculation of Jacobian matrix.

5 Test – State estimation with real reactor

In this part, the real process data will be applied to the state estimators to see how they performance. We are also interested in that how they deal with a sensor failure. We do this by changing the real process data to be zero at some point.

5.1 Data preprocessing

Remove outliers from raw data

Removing outliers can cause data to become more normal but contrary data reality always lies in these suchlike "outliers". There may be many truly valid reasons to remove data-points. These include outliers caused by measurement errors, incorrectly entered data-points or impossible values in real life. In Figure 5.1 below, there are two potential outliers in methane gas flow happened at day 34 and 74. Yet we are not sure they are outliers or not, because temperature and feed flow change during this time. A decrease in temperature can slow down the reaction and thus results in low output. Such decrease can cost time from practical experience. A rough but reliable method is to observe the lasting time in change of the data. Observe from the raw data, temperature change from 35 ℃ to 22℃and to 40℃, and this change actually lasts almost one day around the day 34. So an overshot in methane gas flow can be possible.

The upper part shows the real data given is from t = 0d (19. May 2012) to t = 110d (15. Aug 2013). The time step between each point of sensor data is 15 minutes. The lower part shows the time series of feed flow, reactor temperature and methane gas flow during this period.

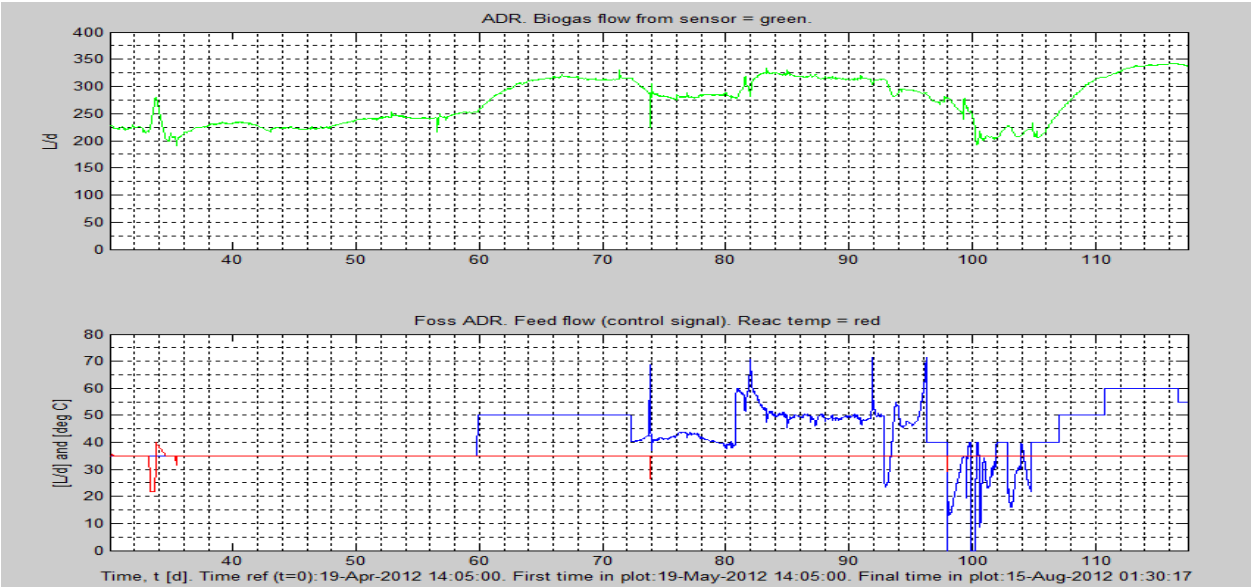

Figure 5.1: Upper: Measured methane gas flow. Lower: Measured feed flow and reactor temperature.

Interpolation

As in our raw dataset, there are miserable data and a big number of NaNs exists inside. So our method is to remove these NaNs and substitute them with interpolation. Linear interpolation will be deployed as our data is logged on a small interval. So in a small time zone, a linear relationship should be holding on these data. An example as Figure 5.2 shows our interpolation for the analysis data inf_vs, which in fact equal to state $S_{vs,in}$. There are four points of NaNs and interpolated here. All the data processed and being prepared to used in our estimator are stored in mat file test.mat as Appendix E.

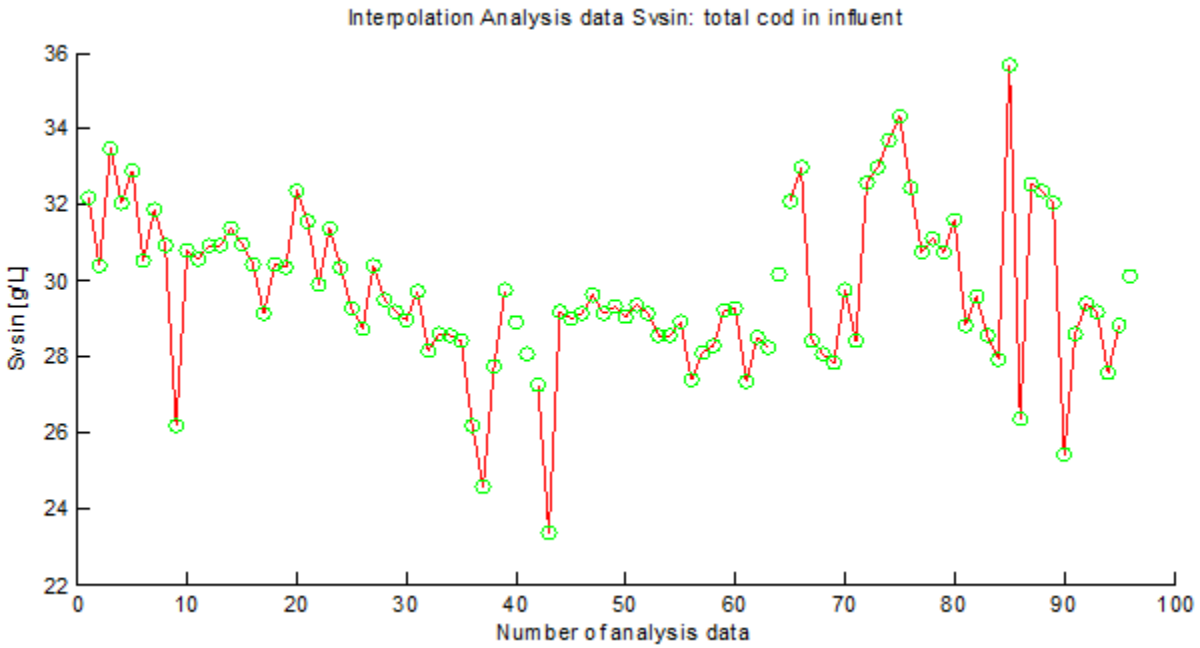

Figure 5.2: Interpolation for the analysis data inf_vs.

5.2 Tuning the state estimators

A tuned Kalman filter will accurately estimate the statistical characteristics of the measurement and process model, such that a minimum variance estimate of the state is produced. Tuning is often considered as an art, but through dynamic simulation and statistical testing, the filter can be systematically adjusted to produce optimal results.

When the initial state of the system is unknown, it is usually set as zero with a high valued initial state covariance matrix. If the best estimation is used as initial value, then a small valued diagonal matrix can be deployed to express the uncertainty of this estimate. Note that for UKF, the initial state covariance is a significant and susceptible parameter. Because the sigma points are chosen so that their mean and covariance to be exactly the initial state and the initial state covariance. And the estimation ability of UKF highly depends on the set of sigma points. UKF

uses a weighted set of deterministically sampled points called sigma-points, which are passed through the nonlinearity and are used to approximate the statistics of the distribution.

The Kalman filter should be tuned to be robust enough to compensate for initial errors. In Section 4.2.1 we have defining Q and R to simulate the real system and the Kalman filters have achieved a satisfactory robust behavior. For R, it can be found directly as the covariance of measurement noise. But for Q, it must either be tuned using a combination of model knowledge and trial and error or via adaptive methods that estimate the covariance while performing the actual estimation. In this case, a trial and error method can be used. The same tuning method will be used in this actual application. Kalman filters with the same initial values and tuning parameters are given in Section 4.2.1.

5.3 Results and discussions

Figure 5.3-5.5 shows the state estimators together with real data from online sensor and laboratory analysis over a time interval of 110 days. Figure 5.3 shows lab analysis data and estimates of the state $S_{bvs} = x_1$ and $S_{vfa} = x_2$.

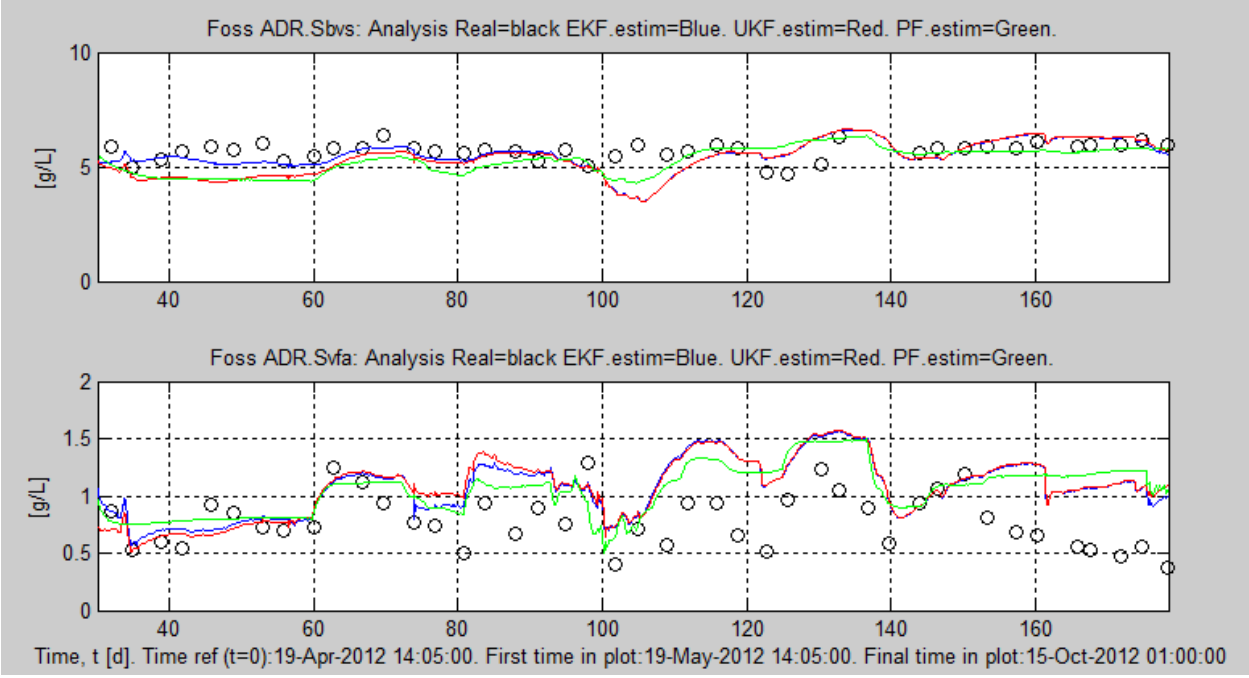

Figure 5.3: Lab analysis data and estimates of the state S_bvs=x_1 and S_vfa=x_2.

Figure 5.4 shows estimates of the state $X_{acid} = x_3$ and $S_{meth} = x_4$. We don't have lab analysis data for these two variables. The only way to judge the estimation is to see if these data are stable and around the steady point or not.

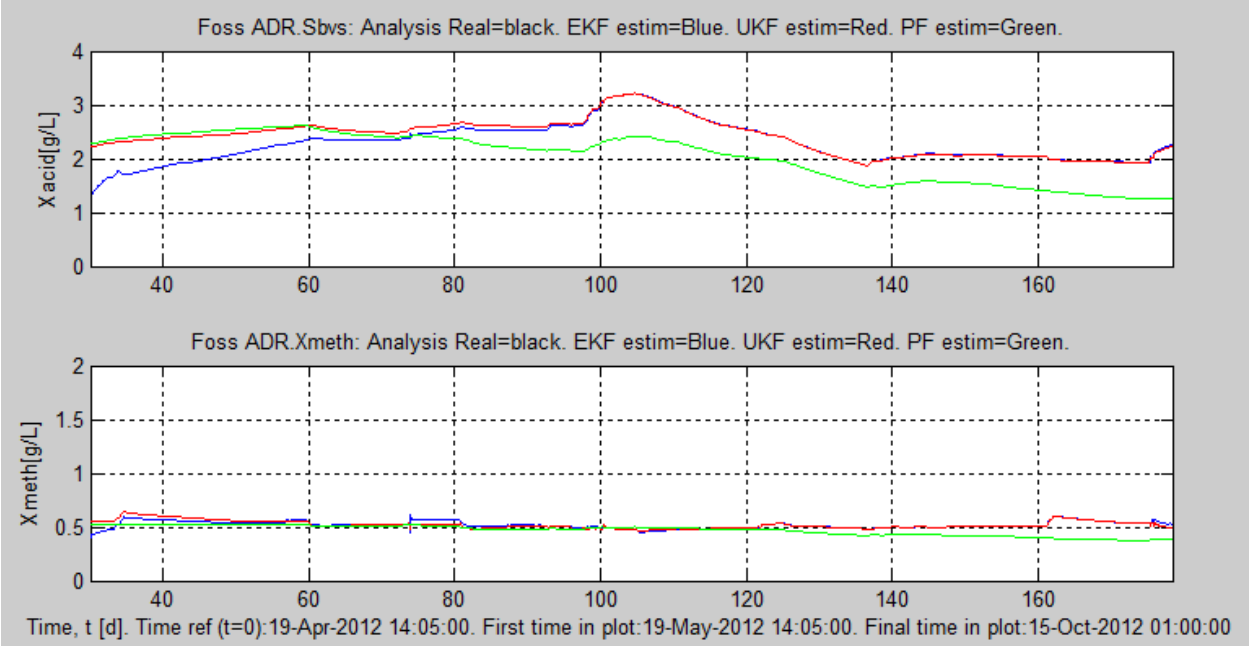

Figure 5.4: Estimate of the states X_acid=x_3 and X_meth=x_4.

Figure 5.5 shows lab analysis data and estimates of the augmentative state $S_{vs,in} = x_5$.

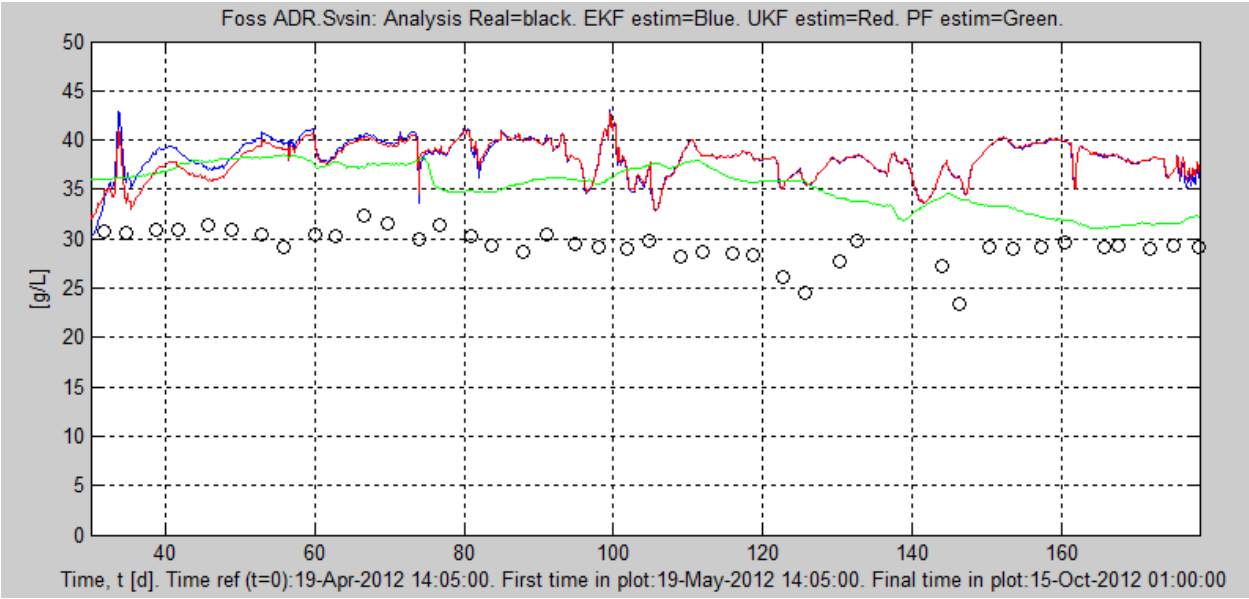

Figure 5.5: Lab analysis data and estimate of the state S_vs,in=x_5 .

Another point is deserved to be discussed is sensor failure. With the rapid development of sensor technology, an increasing number of sensors are installed in various applications for monitoring and control. Due to high employment, the frequency of occurrence of sensor faults is increasing(Kullaa, 2013). As the monitoring and control applications rely on the sensor data, it is of importance that the sensors give accurate and reliable information. A faulty sensor cannot

function properly but instead provide false data, which can result in fatal damage of the system. Therefore, it is necessary to detect such faults and adapt to the new situations with right actions.

Sensor failure can happen due to reasons like battery failure or improper service procedures, and a controller like PID based on this sensor will make a wrong action. And this can put the real process into dangerous situation. But with a Kalman filter, we can avoid the consequence of such a sensor failure may bring to our system. The reason is that the Kalman filters rely on the model predict even without measurement update. So it's vital for the algorithm to recognize a sensor failure. To simulate such an abnormal scenario, we can easily set the sensor measurement value to zero. And when this abnormal scenario happens, the estimators should detect it and react by deleting the innovation process. That means the Kalman filters now completely depends on the process model.

Deleting the innovation process and the Kalman algorithms changed at formula (3.14) and (3.38)

$$x_e(t_k) = x_{apost}(t_k) = x_{apri}(t_k) \tag{5.1}$$

$$x_{apost}(t_k) = x_{apri}(t_k) \tag{5.2}$$

In this simulation test, we compare the result of keeping the innovation process and deleting it. By doing this, we actually make the Kalman filter algorithms more intelligent. Figure 5.6 shows F_{meth} and relative states S_{bvs} and S_{vfa} when we start a sensor failure scenario without deleting the innovation process.

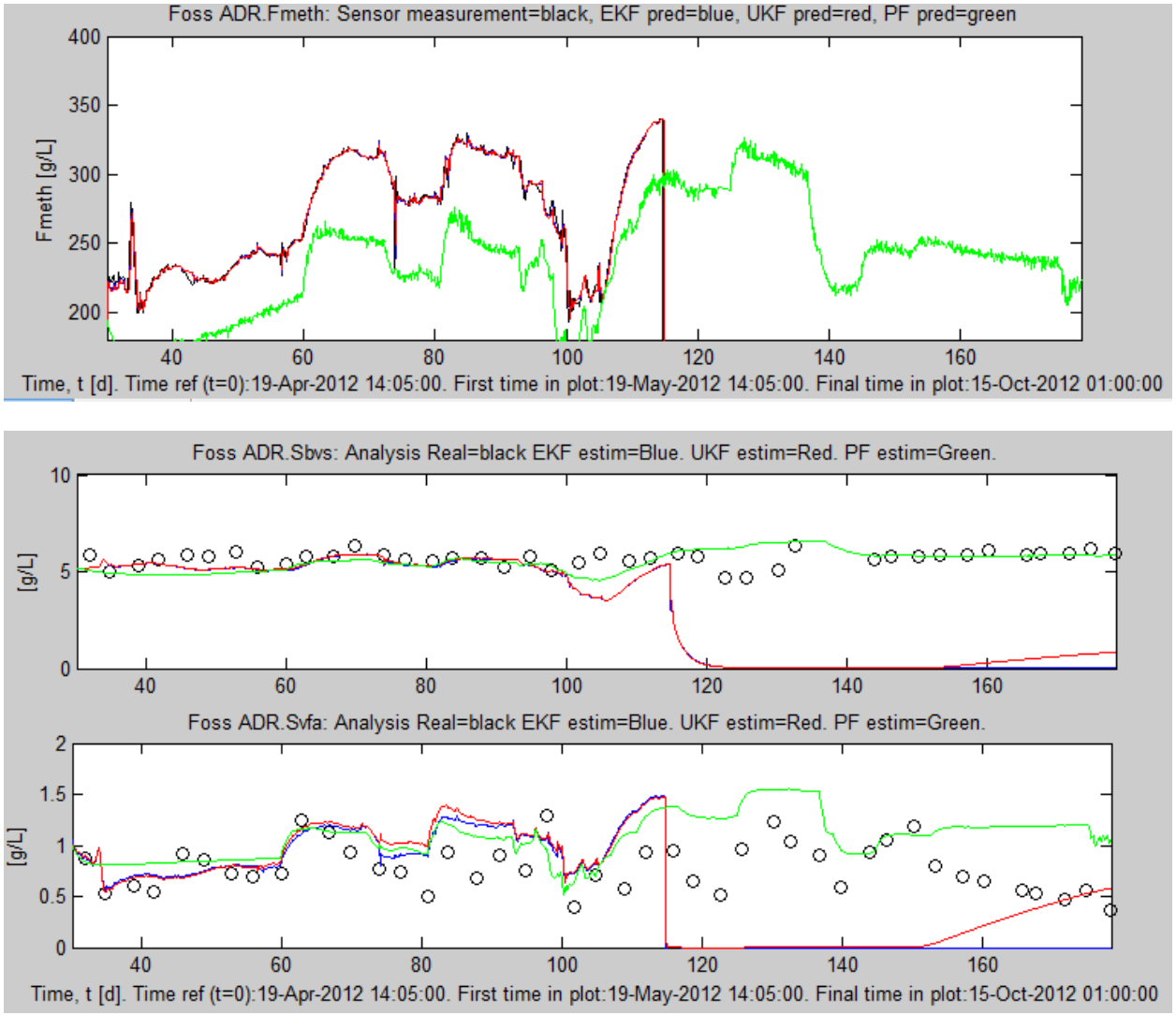

Figure 5.6: Result without deleting the innovation process. The upper is F_meth& the lower are S_bvs and S_vfa.

When sensor measurement fails, the states estimate and measurement estimate F_{meth}also lose control. Based on the theory, EKF and UKF should react similarly as the measurement update of the sate estimate using the normal Kalman filter equations. Instead of innovation process, PF algorithm is using kind of process called resampling. It's interesting to see that the PF perform well when sensor measurement becomes to zero.

Figure 5.7 shows result after deleting the innovation process. Different from Figure 5.6, the states estimate and measurement estimate F_{meth}keep running and have a good performance as normal situation.

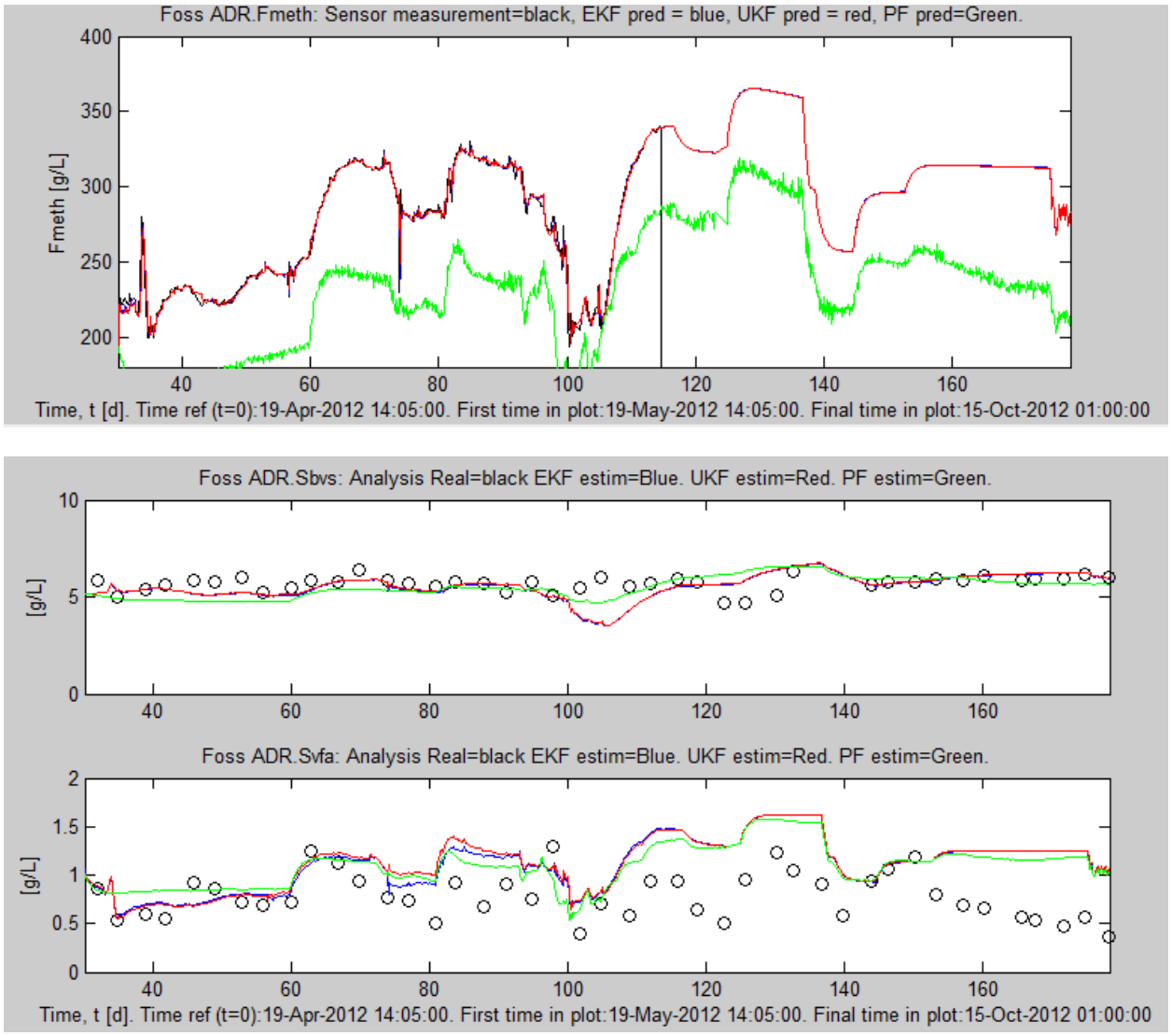

Figure 5.7: Result after deleting the innovation process. The upper is F_meth& the lower are S_bvs and S_vfa.

6 Conclusion and future work

6.1 Conclusion

This report has compared three nonlinear Kalman filtering approaches for the estimation of the states of an AD reactor. The tests have been made by simulating real working conditions, including the failure of biogas measurement sensor. The tests results show that the three filters have good estimation and similar performances.

The simulation results show that the performance of the three filters is similar in case of good knowledge of the initial state. On the contrary, if the initial estimation error covariance is large, then PF shows a faster convergence compared with UKF and EKF. A large enough error can even bring the UKF and EKF to instability. The difference depends on the nature of the three filters. The EKF obtains the posterior by exploiting the linear approximation of the system equations; whereas the UKF, instead of approximating the nonlinear equations, approximates the prior by a limited number of sigma points. The posterior is then obtained by propagating these points through the original nonlinear function. In another side, PF uses a number of independent random variables called particles, sampled directly from the state space, to represent the posterior probability, and update the posterior by involving the new observations. The three approaches give similar results if the initial uncertainty is very close to the mean, so that the sigma points are very close to each other as well as the particles.

In general, the performances of the UKF and of the EKF may differ more or less according to the specific application considered. The main factors that can make the UKF perform better than the EKF are the type of non-linearity and the level of uncertainty that characterize the considered application.

Some considerations must be done on the computational burden of the filters. In this case, the UKF is presented as less demanding compared with EKF, from a computational point of view, because it doesn’t require the computation of Jacobian matrices at each time step. Compared to other methods, the PF is easy to implement and applicable to a wider range of problems.

6.2 Future work

There are several aspects that should be considered for future work.

- Auto-tuning method can be developed for the filters.

- Different models for the AD reactor can be deployed to test one specific Kalman filter. It's meaningful to explore how a Kalman filter reacts to different mathematical models.

- Automatic control method can be developed for this AD reactor. A feed-forward controller or an advanced model-based controller can be used to cooperate with the Kalman filter.

- Filtering with non-Gaussian nonlinear state space model. With the expansion of applications - for example, in the detection of structural changes of time series models, in the analysis of time series with outliers, and in nonlinear time series modeling - the necessity of non-Gaussian state space modeling has become apparent(Kitagawa, 1996).

Reference

CHAI, Q., FURENES, B. & LIE, B. 2007. Comparison of state estimation techniques, applied to a biological wastewater treatment process. 10th International IFAC symposium on computer applications in biotechnology, Mexico, 2007.

FISCHER, J. R., IANNOTTI, E. L. & PORTER, J. H. 1984. Anaerobic digestion of swine manure at various influent solids concentrations. *Agricultural Wastes,* 11**,** 157-166.

FUMING, S., GUANGLIN, L. & JINGLI, W. 2009. Unscented Kalman Filter Using Augmented State in the Presence of Additive Noise. Control, Automation and Systems Engineering. CASE 2009. IITA International Conference on, 11-12 July 2009. 379-382.

GORDON, N. J., SALMOND, D. J. & SMITH, A. F. 1993a. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proceedings F (Radar and Signal Processing). IET, 107-113.

GORDON, N. J., SALMOND, D. J. & SMITH, A. F. M. 1993b. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F,* 140**,** 107-113.

GROSS, J. N., YU, G., RHUDY, M. B., GURURAJAN, S. & NAPOLITANO, M. R. 2012. Flight-Test Evaluation of Sensor Fusion Algorithms for Attitude Estimation. *Aerospace and Electronic Systems, IEEE Transactions on,* 48**,** 2128-2139.

GUSTAFSSON, F. 1997. Slip-based tire-road friction estimation. *Automatica,* 33**,** 1087-1099.

HAUGEN, F., BAKKE, R. & LIE, B. 2012b. State Estimation of a Pilot Anaerobic Digestion Reactor. *Nordic Process Control Workshop 2012.*

HAUGEN, F., BAKKE, R. & LIE, B. 2013. Adapting Dynamic Mathematical Models to a Pilot Anaerobic Digestion Reactor. *MODELING IDENTIFICATION AND CONTROL,* 34**,** 35-54.

HAUGEN, F., BAKKE, R. & LIE, B. 2014. State Estimation and Model-Based Control of a Pilot Anaerobic Digestion Reactor. *Journal of Control Science and Engineering,* 2014**,** 19.

HILL, D. T. 1983. Simplified monod kinetics of methane fermentation of animal wastes. *Agricultural Wastes,* 5**,** 1-16.

HUSAIN, A. 1998. Mathematical models of the kinetics of anaerobic digestion—a selected review. *Biomass and Bioenergy,* 14**,** 561-571.

JAZWINSKI, A. H. 1970. Preface. *In:* ANDREW, H. J. (ed.) *Mathematics in Science and Engineering.* Elsevier.

JULIER, S. & UHLMANN, J. 1997. A New Extension of the Kalman Filter to Nonlinear Systems. International Symposium Aerospace/Defense Sensing, Simulation and Controls, 1997. 182-193.

JULIER, S. J., UHLMANN, J. K. & DURRANT-WHYTE, H. F. 1995. A new approach for filtering nonlinear systems. American Control Conference, Proceedings of the 1995, 21-23 Jun 1995. 1628-1632 vol.3.

KALMAN, R. E. 1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering,* 82**,** 35-45.

KITAGAWA, G. 1996. Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics,* 5**,** 1-25.

KULLAA, J. 2013. Detection, identification, and quantification of sensor fault in a sensor network. *Mechanical Systems and Signal Processing,* 40**,** 208-221.

KUPILIK, M. J. & VINCENT, T. L. 2011. Estimation of biogas composition in a catalytic reactor via an extended Kalman filter. Control Applications (CCA), 2011 IEEE International Conference on, 28-30 Sept. 2011. 768-773.

LAVIOLA JR, J. J. 2003. A comparison of unscented and extended Kalman filtering for estimating quaternion motion. American Control Conference, 2003. Proceedings of the 2003. IEEE, 2435-2440.

MAEDER, U. 2010. *Augmented models in estimation and control,* DSc Dissertation ETH Zurich.

MYERS, M. R., JORGE, A. B., MUTTON, M. J. & WALKER, D. G. 2012. A comparison of extended Kalman filter, particle filter, and least squares localization methods for a high heat flux concentrated source. *International Journal of Heat and Mass Transfer,* 55**,** 2219-2228.

RAY, L. R. 1997. Nonlinear Tire Force Estimation and Road Friction Identification: Simulation and Experiments. *Automatica,* 33**,** 1819-1833.

RISTIC, B., ARULAMPALAM, S. & GORDON, N. 2004. *Beyond the Kalman filter: Particle filters for tracking applications,* Artech house.

SIMON, D. 2006a. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches,* Wiley. com.

SIMON, D. 2006b. *Using nonlinear Kalman filtering to estimate signals* [Online]. Available: http://www.embedded.com/design/connectivity/4025693/Using-nonlinear-Kalman-filtering-to-estimate-signals 2014].

SIMON, D. 2008. A comparison of filtering approaches for aircraft engine health estimation. *Aerospace Science and Technology,* 12**,** 276-284.

SIMON, D. J. 2006c. Using nonlinear Kalman filtering to estimate signals. *Embedded Systems Design,* 19**,** 38-53.

WAN, E. A. & VAN DER MERWE, R. 2000. The unscented Kalman filter for nonlinear estimation. Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000, 2000. 153-158.

WAN, E. A. & VAN DER MERWE, R. 2002. The Unscented Kalman Filter. *Kalman Filtering and Neural Networks.* John Wiley & Sons, Inc.

Appendices

Appendix A Task description for this thesis

Telemark University College

Faculty of Technology

FMH606 Master's Thesis

Title: **Evaluation and comparison of Kalman Filter algorithms**

TUC supervisor: Associate Professor Finn Haugen

External partner: -

Task description:

The main aim of the thesis is to evaluate and compare a number of alternative Kalman Filter algorithms in applications to a real and a simulated anaerobic digestion (AD) reactor.

If possible, the results should be presented in an article to be submitted to a recognized scientific journal. This article, or a draft of an article, will then be a part of the thesis.

Task background:
State estimators in the form of an appropriate Kalman Filter are useful for monotoring AD reactors. Typically, in practical applications, biogas flow sensors are used. None of the state variables are measured, but they can be estimated with a Kalman Filter. It is of interest to compare and evaluate alternative Kalman Filter algorithms with respect to their implementation and performance. Both online sensor data and laboratory data for the real AD reactor exist.

Student category: SCE student.

Practical arrangements:

The thesis will be accomplished at TUC.

Signatures:

Students (date and signature):

Supervisor (date and signature):

Appendix B Calculation of Jacobian matrix

```
% Calculate Jacobian matrix for EKF
function [A B C] = ComputeJacobian(x,Ffeed)
%state space model of the biogas reactor
%-------------------------------------------------------------------------
% Define parameters
Af = 0.69;  %acidity constant (g VFA/L)
b  = 2.90;  %Retention time ratio d/d
B0 = 0.25;  %Biodegradability constant (g BVS/L)/ (g VS/L)
k1 = 3.89;  %Yield constant g BVS/(g acidogens/L)
k2 = 1.76;  %Yield constant g VFA/(g acidogens/L)
k3 = 31.7;  %Yield constant g VFA/(g methanogens/L)
k5 = 26.3;  %Yield constant L/g methanogens
Kd = 0.02;  %Specific death rate of acidogens d/1
Kdc = 0.02; %Specific death rate of methanogens d/1
Ksc = 3;    %Monod half-velocity constant for methanogens g VFA/L
V = 250;    %Effective reactor volume L
Treac = 35; %temperature in the reactor
um = 0.013*Treac-0.129;
umc = um;
Svsin=30.2;
%-------------------------------------------------------------------------
%Naming states
Sbvs = x(1);
Svfa = x(2);
Xacid = x(3);
Xmeth = x(4);
% Svsin = x(5);
%define matrices for the continuous time model
%Ac matrix
A11 = -(Ffeed/V)-(um*k1*Ks*Xacid)/((Ks+Sbvs)^2);
A12 = 0;
A13 = -((k1*um*Sbvs)/(Ks+Sbvs));
A14 = 0;
A21 = (um*k2*Ks*Xacid)/(Ks+Sbvs)^2;
A22 = -(Ffeed/V)-(umc*k3*Ksc*Xmeth)/(Ksc+Svfa)^2;
A23 = (k2*um*Sbvs)/(Ks+Sbvs);
A24 = -((k3*umc*Svfa)/(Ksc+Svfa));
A31 = (um*Ks*Xacid)/(Ks+Sbvs)^2;
A32 = 0;
A33 = -(Ffeed/(b*V))+(um*Sbvs)/(Ks+Sbvs)-Kd;
A34 = 0;
A41 = 0;
A42 = (umc*Ksc*Xmeth)/((Ksc+Svfa)^2);
A43 = 0;
A44 = -(Ffeed/(b*V))+(umc*Svfa)/(Ksc+Svfa)-Kdc;
Ac = [A11 A12 A13 A14;A21 A22 A23 A24;A31 A32 A33  A34;A41 A42 A43 A44];
%Bc matrix
B11 = (B0*Svsin - Sbvs)/V;
B21 = (Af*B0*Svsin - Svfa)/V;
B31 = -Xacid/(b*V);
B41 = -Xmeth/(b*V);
Bc = [B11;B21;B31;B41];
%Cc matrix
C11 = 0;
```

```
C12 = (umc*V*k5*Ksc*Xmeth)/((Ksc+Svfa)^2);
C13 = 0;
C14 = (V*k5*umc*Svfa)/(Ksc+Svfa);
Cc = [C11 C12 C13 C14];
%Dc matrix
D11 = 0;
Dc = D11;
sys = ss(Ac,Bc,Cc,Dc); %continuous time model
%-------------------------------------------------------------------------
%Transfer continuous time model to disctrete time model
Ts = 0.1; %sampling time unit:s
sysd = c2d(sys,Ts,'zoh'); %discrete time model
%-----------------#####################################------------------
%Augmented state space model(discrete time),augmented state Svsin
% A15 = B0*Ffeed/V; %after add Svsin as an augmented state variable
% A25 = B0*Af*Ffeed/V;
% A35 = 0;
% A45 = 0;
% A51 = 0; A52 = 0; A53 = 0; A54 = 0; A55 = 0;
% B51 = 0;
% C15 = 0;
% Acaug = [A11 A12 A13 A14 A15;A21 A22 A23 A24 A25;A31 A32 A33  A34
A35;A41...
%     A42 A43 A44 A45;A51 A52 A53 A54 A55];
% Bcaug = [B11;B21;B31;B41;B51];
% Ccaug = [C11 C12 C13 C14 C15];
% Dcaug = 0;
% sysaug = ss(Acaug,Bcaug,Ccaug,Dcaug); %continuous time model
% sysdaug = c2d(sysaug,Ts,'zoh'); %discrete time model
[A,B,C,D] = ssdata(sysd);
%-------------------------------------------------------------------------
%Transfer function can be deployed for a PID controller
% Had = tf(sys); %continuous transfer function
% Hadaug = tf(sysaug); %continuous transfer function of augmented model
```

Appendix C Step test of nonlinear model and linearized model

```
%Evaluation of linearized model
T_min = 0;
T_max = 2000;
T=T_min: T_max;
N=length(T_min:T_max);
dt= (T_max-T_min)/(N-1)/100;

X_data1=zeros(N,4);
Y_data1=zeros(N,1);
U_data1=zeros(N,1);
%Steady-state operating point
Xs=[5.1; 1.01 ;  1.31 ;0.364];
Us=45;
X1= [5.1; 1.01 ; 1.31 ;0.364];  %initial value[0;0;0;0];%
%parameters
```

```
k5 = 26.3;
Kd = 0.02;
Kdc = 0.02;
Ks = 15.5;
Ksc = 3;
V = 250;
Treac = 35;
um = 0.013*Treac-0.129;
umc = um;
Ys=V*(umc*Xs(2)/(Ksc+Xs(2)))*k5*Xs(4);
u=45; %give step amptitude


X= [5.1; 1.01 ; 1.31 ;0.364]; %[0;0;0;0];
X_data=zeros(N,4);
Y_data=zeros(N,1);
U_data=zeros(N,1);
%step test of nonlinear model
for k=1:N
    X= X+dt*nonlinearmodel(X,u); %Forward euler method
    X_data(k,1:4)=X'; %Output states
    Y = V*(umc*X(2)/(Ksc+X(2)))*k5*X(4);
    Y_data(1,1)=Y'; %output methane gas
end

%step test of linearized model
%to get Ac Bc Cc, need to run file'discretespacemodel'first
for l=1:N
    %d(x-xs)/dt=Ac*(x-xs)+Bc*(u-us) xs-the steady point
    X1= X1 + dt*(Ac*(X1-Xs)+Bc*(u-Us));
    X_data1(l,1:4)=X1';
    Y1 = Ys + Cc*(X1-Xs); %(y-ys)=Cc*(x-xs)
    Y_data1(l,1)=Y1';
end
%%plot result
figure(1)
plot(T,X_data1(:,1),'r:',T,X_data1(:,2),'b:',T,X_data1(:,3),'c:',...
    T,X_data1(:,4),'k:')
hold on
plot(T,X_data(:,1),'r',T,X_data(:,2),'b',T,X_data(:,3),'c',T,X_data(:,4),'k')
title('Sbvs--red  Svfa--blue  Xacid--brillantblue  Xmeth--black')
ylabel('[g/L]')
xlabel('Time step')
legend('nonlinear','linearized')
hold off
figure(2)
hold on
plot(T,Y_data1(:,1),'g:')
plot(T,Y_data(:,1),'r')
legend('Fmeth')
hold off
```

Appendix D Simulation-State estimation with Kalman filters

```
%=========================================================================
%%SCRIPT FILE COMPUTATION OF STATES OF AD REACTOR
%
% (C) Chao xi
%     Telemark University College
%     March 15, 2014
% Discrete-time EKF, UKF, PF simulation for a biogas reactor
% Estimate Sbvs, Svfa, Xacid, Xmeth and Svsin on the basis of noisy
measurements of Fmeth
%=========================================================================
clear all; clc;
T_stop = 50;
dt=0.1;       % how often measurements are obtained
t=0:dt:T_stop;
N=length(t);  % Simulation length


R = 1.44;     % Measurement noise covariance
Q = diag([0.002725 0.0001 0.000169 0.000013 0.1024]); %process noise
covariance
%Add noise to each state and measurement
w = sqrt(Q)*randn(5,N); %process noise
pn = w;
v = sqrt(R)*randn(1,N); %measurement noise
mn = v;

E=eye(5,5);
P_apost=0.01*eye(5,5);%initial covariance of estimation error ekf
Pukf = 0.01*eye(5,5);%initial covariance of estimation error ukf

x_apost=[5.2155;1.0094;1.3128;0.3635;30.2]; %initial state estimate ekf
x_apostukf = [5.2155;1.0094;1.3128;0.3635;30.2];  %initial state estimate ukf
x = [5.2155;1.0094;1.3128;0.3635;40.2];     %initial state simulator

Ffeedk=45;      %feed into the AD reactor
% Initialize arrays for plotting at the end of the program
Y=zeros(N,1);
X=zeros(N,5);
X_ekf=zeros(N,5);
X_ukf=zeros(N,5);
X_pf=zeros(N,5);
Y_m = zeros(N,1);
Y_p = zeros(N,1);

% Initialization for particle filter
Ppf = 0.01*eye(5,5);     %initial covariance of estimation error pf
n = 500;           %number of particles
% Initialize the particle filter.
for k = 1 : n
    x_apostpf(:,k) = x + sqrt(Ppf)* pn(:,k);
end
```

```
%-----------------------------EKF-----------------------------------------
for i=1:N
    %output measurement states(or calculated from model)
    X(i,:)=x';
    %simulation state variable for next step
    x=x+dt*(nonlinearmodel(x,Ffeedk))+ pn(:,i));
    %Simulate noisy measurement(or calculated from model)
    y=250*(0.326*x(2)/(3+x(2)))*26.3*x(4)+ mn(:,i);
    Y_m(i,:)=y';
   % tic;
      %%Calculate Jacobian matrix A
        [A B C] = ComputeJacobian(x_apost,Ffeedk);
    %%Time update
    %(1) Apriori estimate(predicted estimate)
          x_apri = x_apost + dt*nonlinearmodel(x_apost,Ffeedk);

    %(2) Covariance of estimation error
          P_apri = A*P_apost*A'+E*Q*E';

    %%Measurement update
    %(1) Kalman filter gain matrix
         a1 = P_apri*C';
         a2 = C*P_apri*C'+R;
         K=a1/a2;
         Kekf_gain(i,:)= K';
    %(2) Measurement estimate
         y_apri=250*(0.326*x_apri(2)/(3+x_apri(2)))*26.3*x_apri(4);
         Y_p(i,:)=y_apri';
    %(3) State estimate or corrected estimate
         x_apost=x_apri+K*(y-y_apri);
         X_ekf(i,:)=x_apost';
    %(4) Update Covariance of estimation error
         I=eye(5,5);
         P_apost=(I-K*C)*P_apri*(I-K*C)'+K*R*K';

%---------------------------UKF-----------------------------------------

   W = ones(10,1) / 10; % UKF weights (2*5 states)
   % Generate the UKF sigma points.
   [root,p] = chol(5*Pukf);%
   for j = 1 : 5
       sigma(:,j) = x_apostukf + root(j,:)'; % the first 5 sigma points
       sigma(:,j+5) = x_apostukf - root(j,:)'; % the last 5 sigma points
   end
   for j = 1 : 10
       xbreve(:,j) = sigma(:,j); %10 sigma points in vector xbreve
   end

 % UKF time update
   for j = 1 : 10
         xbreve(:,j) = xbreve(:,j) + dt*nonlinearmodel(xbreve(:,j),Ffeedk);
   end
      x_apriukf = zeros(5,1);
  for j = 1 : 10
```

```
      x_apriukf = x_apriukf + W(j) * xbreve(:,j);
  end
      Pukf = zeros(5,5);
  for j = 1 : 10
      Pukf = Pukf + W(j) * (xbreve(:,j) - x_apriukf) * (xbreve(:,j) -
x_apriukf)';
  end
      Pukf = Pukf + Q;

   % UKF measurement update
   %(a)Generating new sigma points
   [root,p] = chol(5*Pukf);% number 5 means 5 states
   for j = 1 : 5
       sigma(:,j) = x_apriukf + root(j,:)'; % the first 5 sigma points
       sigma(:,j+5) = x_apriukf - root(j,:)'; % the last 5 sigma points
   end
   for j = 1 : 10
       xbreve(:,j) = sigma(:,j); %10 sigma points in vector xbreve
   end
   %(b)substitute the sigma points into nonlinear equation
  for j = 1 : 10
      zukf(:,j) = 250*(0.326*xbreve(2,j)/(3+xbreve(2,j)))*26.3*xbreve(4,j);
  end
   %(c)Caiculate the predicted measurement
      zhat = 0;
  for j = 1 : 10
      zhat = zhat + W(j) * zukf(:,j);
  end
  %(d)Estimate the covariance of the predicted measurement and cross
covariance
  Py = 0;
  Pxy = zeros(5,1);
  for j = 1 : 10
      Py = Py + W(j) * (zukf(:,j) - zhat) * (zukf(:,j) - zhat)';
      Pxy = Pxy + W(j) * (xbreve(:,j) - x_apriukf) * (zukf(:,j) - zhat)';
  end
  Py = Py + R;
  Kukf = Pxy /(Py);
  Kukf_gain(i,:) = Kukf';
  Y_ukf(i,:)=zhat';
  %(e)measurement update of the state estimate
  x_apostukf = x_apriukf + Kukf * (y - zhat);
  X_ukf(i,:)=x_apostukf';
  Pukf = Pukf - Kukf * Py * Kukf';

  %-------------------PF(Particle filter)---------------------------------
  % initial sate value see xhat,state corvariance Ppf, noise as Q,R
  % Simulate the continuous-time part of the particle filter (time update).
     x_apripf = x_apostpf;
  for k = 1:n
     x_apripf(:,k) = x_apripf(:,k)+dt*(nonlinearmodel(x_apripf(:,k),Ffeedk)+
pn(:,k));
     yhat = 250*(0.326*x_apripf(2,k)/(3+x_apripf(2,k)))*26.3*x_apripf(4,k);
     vhat(k) = y - yhat;
  end
     Y_pf(i,:)=yhat';
```

```
    % Note that we need to scale all of the q(k) probabilities in a way
    % that does not change their relative magnitudes.
    % Otherwise all of the q(k) elements will be zero because of the
    % large value of the exponential.
    vhatscale = max(abs(vhat)) / 4;
    qsum = 0;
    for k = 1 : n
        q(k) = exp(-(vhat(k)/vhatscale)^2);
        qsum = qsum + q(k);
    end
   % Normalize the likelihood of each a priori estimate.
    for k = 1 : n
        q(k) = q(k) / qsum;
    end
   % Resample.
    for k = 1 : n
        u = rand; % uniform random number between 0 and 1
        qtempsum = 0;
        for m = 1 : n
            qtempsum = qtempsum + q(m);
            if qtempsum >= u
                x_apostpf(:,k) = x_apripf(:,m);
                % Use roughening to prevent sample impoverishment.
                E_pf = max(x_apripf')' - min(x_apripf')';
                sigma_pf = 0.2 * E_pf * n^(-1/length(x));
                x_apostpf(:,k) = x_apostpf(:,k) + sigma_pf .* pn(:,k);
%                x_apostpf(3,k) = max(0,x_apostpf(3,i)); % xacid cannot be
negative
                break;
            end
        end
    end

    % The particle filter estimate is the mean of the particles.
    xhat = 0;
    for k = 1 : n
        xhat = xhat + x_apostpf(:,k);
    end
    xhat = xhat / n;
    X_pf(i,:)= xhat';
end

%---------------------------Plot results---------------------------------
close all;
figure(1)
 subplot(2,1,1)
plot(t,X(:,1),'k',t,X_ekf(:,1),':b',t,X_ukf(:,1),'-.r',t,X_pf(:,1),'--
g','LineWidth',1.3)
ylabel('Sbvs[g/L]')
title('Sbvs: Real=black  AEKF.est=blue&dotted  AUKF.est=red&dash-dot
APF.est=green&dashed')
% legend('Real','EKF','UKF','PF')
% xlabel('Time(days)')
% grid on
% figure(2)
 subplot(2,1,2)
```

```
% figure(2)
plot(t,X(:,2),'k',t,X_ekf(:,2),':b',t,X_ukf(:,2),'-.r',t,X_pf(:,2),'--
g','LineWidth',1.3)
ylabel('Svfa[g/L]')
xlabel('Time(days)')
title('Svfa: Real=black  AEKF.est=blue&dotted  AUKF.est=red&dash-dot
APF.est=green&dashed')
% legend('Real','EKF','UKF','PF')
% grid on
hold on
figure(3)
subplot(2,1,1)
plot(t,X(:,3),'k',t,X_ekf(:,3),':b',t,X_ukf(:,3),'-.r',t,X_pf(:,3),'--
g','LineWidth',1.3)
ylabel('Xacid[g/L]')
title('Xacid: Real=black  AEKF.est=blue&dotted  AUKF.est=red&dash-dot
APF.est=green&dashed')
% xlabel('Time(days)')
% legend('Real','EKF','UKF','PF')
% grid on

subplot(2,1,2)
% figure(4)
plot(t,X(:,4),'k',t,X_ekf(:,4),':b',t,X_ukf(:,4),'-.r',t,X_pf(:,4),'--
g','LineWidth',1.3)
ylabel('Xmeth[g/L]')
xlabel('Time(days)')
title('Xmeth: Real=black  AEKF.est=blue&dotted  AUKF.est=red&dash-dot
APF.est=green&dashed')
hold on
% legend('Real','EKF','UKF','PF')
% grid on

figure(4)
plot(t,X(:,5),'k',t,X_ekf(:,5),':b',t,X_ukf(:,5),'-.r',t,X_pf(:,5),'--
g','LineWidth',1.3)
ylabel('Svs,in[g/L]')
xlabel('Time(days)')
title('Svs,in: Real=black  AEKF.est=blue&dotted  AUKF.est=red&dash-dot
APF.est=green&dashed')
legend('Real','AEKF','AUKF','APF')
grid on

%  figure(4);
%  subplot(2,3,1)
%  plot(t,Kekf_gain(:,1),'b',t,Kukf_gain(:,1),'r','LineWidth',1.3)
%  title('K1: EKF=blue UKF=red')
%  xlabel('Time(days)')
%  subplot(2,3,2)
%  plot(t,Kekf_gain(:,2),'b',t,Kukf_gain(:,2),'r','LineWidth',1.3)
%  title('K2: EKF=blue UKF=red')
%  xlabel('Time(days)')
%  subplot(2,3,3)
%  plot(t,Kekf_gain(:,3),'b',t,Kukf_gain(:,3),'r','LineWidth',1.3)
%  title('K3: EKF=blue UKF=red')
%  xlabel('Time(days)')
```

```
%  subplot(2,3,4)
%  plot(t,Kekf_gain(:,4),'b',t,Kukf_gain(:,4),'r','LineWidth',1.3)
%  title('K4: EKF=blue UKF=red')
%  xlabel('Time(days)')
%  subplot(2,3,5)
%  plot(t,Kekf_gain(:,5),'b',t,Kukf_gain(:,5),'r','LineWidth',1.3)
%  title('K5: EKF=blue UKF=red')
%  xlabel('Time(days)')

%  figure(6);
%  plot(t,Y_m,'k',t,Y_p,':b',t,Y_ukf,'--r',t,Y_pf,'-.g')
%  title('R=0.0144 Sensor.output-blue Priori.Est:EKF-blue UKF-red PF-green')
%  legend('measurement','EKF','UKF','PF')
%
%plot error
RMSErr_ekf = zeros(1,4);
RMSErr_ukf = zeros(1,4);
RMSErr_pf = zeros(1,4);
for i = 1 : 4
    RMSErr_ekf(i) = sqrt((norm(X(i,:) - X_ekf(i,:)))^2 / N / dt);
    RMSErr_ukf(i) = sqrt((norm(X(i,:) - X_ukf(i,:)))^2 / N / dt);
    RMSErr_pf(i)  = sqrt((norm(X(i,:) - X_pf(i,:)))^2 / N / dt);
end
disp(['extend filter RMS error = ', num2str(RMSErr_ekf)]);
disp(['unscented filter RMS error = ', num2str(RMSErr_ukf)]);
disp(['particle filter RMS error = ', num2str(RMSErr_pf)]);
% %%=========================================================================%%
```

Appendix E Outlier removing and data preprocessing

```
%%=========================================================================
load('foss_data_from_19apr2012_to_15oct2013')
%%remove outlier
% Calculate the mean and the standard deviation
mu = mean(biogasflow_bronkhorst_filt)
sigma = std(biogasflow_bronkhorst_filt)
[n,p] = size(biogasflow_bronkhorst_filt);
% Create a matrix of mean values by
% replicating the mu vector for n rows
MeanMat = repmat(mu,n,1);
% Create a matrix of standard deviation values by
% replicating the sigma vector for n rows
SigmaMat = repmat(sigma,n,1);
% Create a matrix of zeros and ones, where ones indicate
% the location of outliers
outliers = abs(biogasflow_bronkhorst_filt - MeanMat) > 3*SigmaMat;
% Calculate the number of outliers in each column
nout = sum(outliers)
count(any(outliers),:) = [];

%%interpolation for analysis data
x = inf_vs;
t = [1:length(x)];
i = find(isnan(x)==1) %find the location of element with value of NaN
t_nan = t(~isnan(x));
```

```
x_nan = x(~isnan(x)); %delete the value of NaN

for j = 1:length(i)
    x(i(j)) = interp1(t_nan,x_nan,i(j),'linear','extrap');%Intopolation
end
figure;
plot(1:length(inf_vs),inf_vs,'r',1:length(x),x,'go')
ylabel('[g/L]')
title('Interpolation_ Analysis data Svsin:total cod in influent')
%---------Data preprocessing-------------------------
 % Save variables, where FILENAME is a variable:
    savefile = 'test.mat';
    %save log data,pick one every 10 values
    t_an = t_an_rel_ref;
    t_log = t_log_rel_ref(1:10:length(t_log_rel_ref));
    Fmeth = Fmeth_raw(1:10:length(Fmeth_raw));
    Ffeed = Ffeed_raw(1:10:length(Ffeed_raw));
    Treac = Treac_raw(1:10:length(Treac_raw));

    save(savefile, 'xlabel_txt','t_init','t_final',...
't_log','Ffeed','Treac', 'Fmeth','t_an','Sbvs_an','Svfa_an','Svsin_an');
%-----------------------------------
```